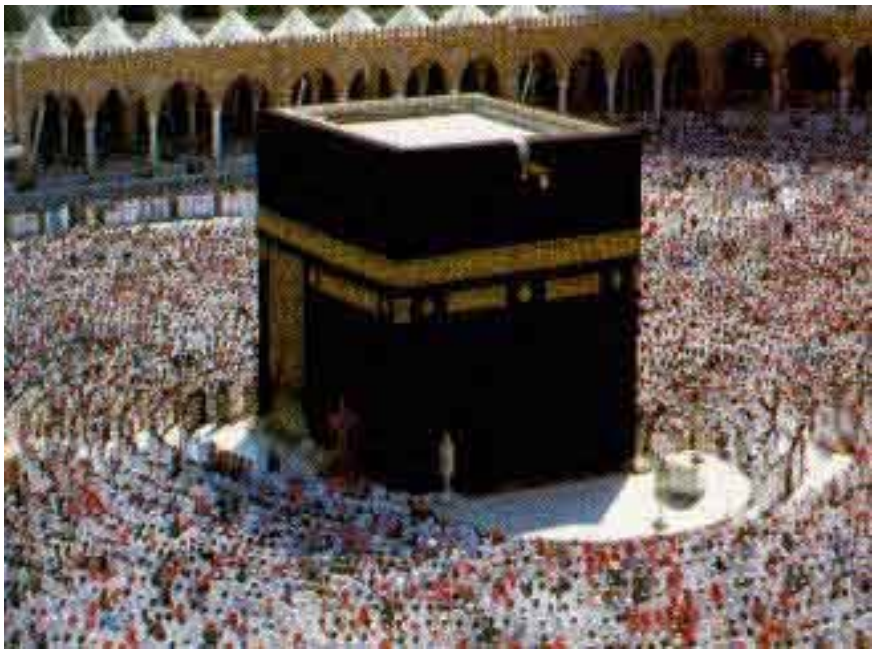
The Road to Makka

Arif Zaman

Lahore University of Management Sciences

Email: arifz@lums.edu.pk

Abstract



Abstract: All muslims are expected to pray *facing the house of Allah*, that is located in the city of Makka. As long as one is in the nearby vicinity, the direction is obvious, but as distances grow, it becomes a difficult problem to determine the proper direction. Here are a few different kinds of maps that attempt to illustrate the problem and solutions.
Note: The Black square building is called the *Kaaba*, while the direction toward it is called the *Qibla*. The use of a letter *Q* without a following *u* represents a *k*-like sound which is made further back in the throat, and is probably unique to Arabic and languages derived from it.

Background (optional)

Muslim scholars over the ages developed a strong science of astronomy, and spherical geometry, because of the need to understand the lunations of the moon (from which dates are reckoned), and the location of the sun (from which times of the five daily prayers are decided).
The problem of finding the Qibla direction is a much simpler problem than the finding the location of the sun or the moon. Some of the interesting solutions that are already in use are given below.

Sun over Makka

Since the latitude of Makka is in the tropics, there is a time of the year when the sun is directly over Makkah. At that time, anyone in the hemisphere of Makka can simply sight the sun, and find the direction in which to pray. The people on the other side of the hemisphere can wait for the time when the sun is directly above the point on the globe opposite to Makkah. At that point they can determine the direction away from the sun, or the direction of the shadows.

Angle from the Sun

Since the study of the sun was already quite detailed, it is possible to find the time of day when the Makka is a particular angle from the sun. Usually one can find charts indicating the time on any day when the sun is 0, 90, 180 or 270 degrees from Makkah. The advantage of this method over the previous is that one does not have to wait an entire year. Over the following method, the advantage is that one doesn't have to solve the addition problem of locating the North.

Angle from North

Finally, the approach that we will follow here is the one used most commonly, which is to compute the angle of the direction to Makkah, expressed in terms of degrees East of North. I have modified that to express the answer as degrees North of East (to reflect the usual mathematical convention of measuring angles). The result is a well known spherical geometry formula, with the angle (North of East) given by

$$(x, y) \to \arctan(\cos(y)\tan(y_0) - \sin(y)\cos(x_0 - x), \sin(x_0 - x))$$

where (x, y) is the location of the person and (x_0, y_0) is the location of Makkah, which in degrees is nearly (39.823333, 21.423333). This formula uses the two valued arctan function of Maple (which gives an answer between -Pi and Pi).

The direction towards Makka is fairly intuitive in most of the world, except in North America where it is initially quite a surprise to many people that they must face roughly in a northeast direction, in order to be facing Makkah. An Azimuthal projection centered at Makka makes the problem quite clear, showing the polar route to the the shortest "great circle" path.

Even then it is quite interesting that many people repeatedly got confused about this issue, it was hotly discussed by lay people who insisted (based on the Mercator Projections that are seen far more than the globe) that since the US is north of Makkah, one should face toward the southeast! This debate continually resurfaces, despite a number of very well written articles and even books explaining the issue in detail.

The Problem Statement

To every point on the globe, one can assign an angle which indicates the direction from that point toward Makka, as measured in degrees north of east.

What does this look like?

Draw maps depicting this function.

Some Basic Definitions [Required before everything else]

A basic convention is that lower case letters are for radians, and upper case for degrees.

The location of Makka

```
> X0 := 39.823333: Y0:= 21.423333:
x0 := X0*Pi/180: y0:= Y0*Pi/180:
```

This draws lines radiating from the origin, of length r

```
> polargrid := r -> PLOT(CURVES(evalf( r*
seq([[cos(15*x*Pi/180),sin(15*x*Pi/180)],
    -[cos(15*x*Pi/180),sin(15*x*Pi/180)]],x=1..24)
))):
```

This function draws similar lines, but centered at [x0,y0] and bounded by a rectangle. Also the lines are drawn using small steps, so that they can be mapped later to a sphere.

```
> boxpolar := proc(d)
  local lines,pts,p0,p,t,dp;
  lines := NULL;
  p0 := evalf([X0,Y0]);
  for t from 15 to 360 by 15 do
    p  := p0;
    pts:= p0;
    dp := evalf(d*[cos(t*Pi/180), sin(t*Pi/180)]);
    while abs((p+dp)[1])<180 and abs((p+dp)[2])<90 do
      p := evalf(p+dp);
      pts := pts, (p+dp);
    end do;
    lines := lines, [pts];
  end do;
  return PLOT(CURVES(lines),COLOR(RGB,0,0,0));
end proc:
```

This reads the world map data, and also some basic functions

All the world map pictures are courtesy of Dr. Ross Taylor (taylor@clarkson.edu) who has done extensive work on mapping the world using Maple. The data used by him comes from

http://www.ngdc.noaa.gov/mgg/shorelines/data/gshhs/gshhs_shp/

Global Self-consistant Hierarchical High-resolution Shorelines

Version 1.2 May 18, 1999

Made programs POSIX.1 compliant and added binary open for DOS.

Version 1.1, April 30, 1996

Paul Wessel, G&G, SOEST, U of Hawaii (wessel@soest.hawaii.edu)
Walter H. F. Smith, NOAA Geosciences Lab (walter@amos.grdl.noaa.gov)

Ref: Wessel, P., and W. H. F. Smith, 1996, A global self-consistent, hierarchical, high-resolution shoreline database, J. Geophys. Res., 101, 8741-8743.

```
> read `c:/maple/earth/world.m`:
read `c:/maple/earth/Source/createMAP.mpl`:
> globe := proc(map)
  local s;
  s := plottools[transform]((x,y) -> [1,x*Pi/180,(90-
y)*Pi/180]):
  plots[display]({plots[changecoords](s(map), spherical),
                  plottools[sphere]([0,0,0], 0.99)},
shading=zgreyscale);
end proc:
>
```

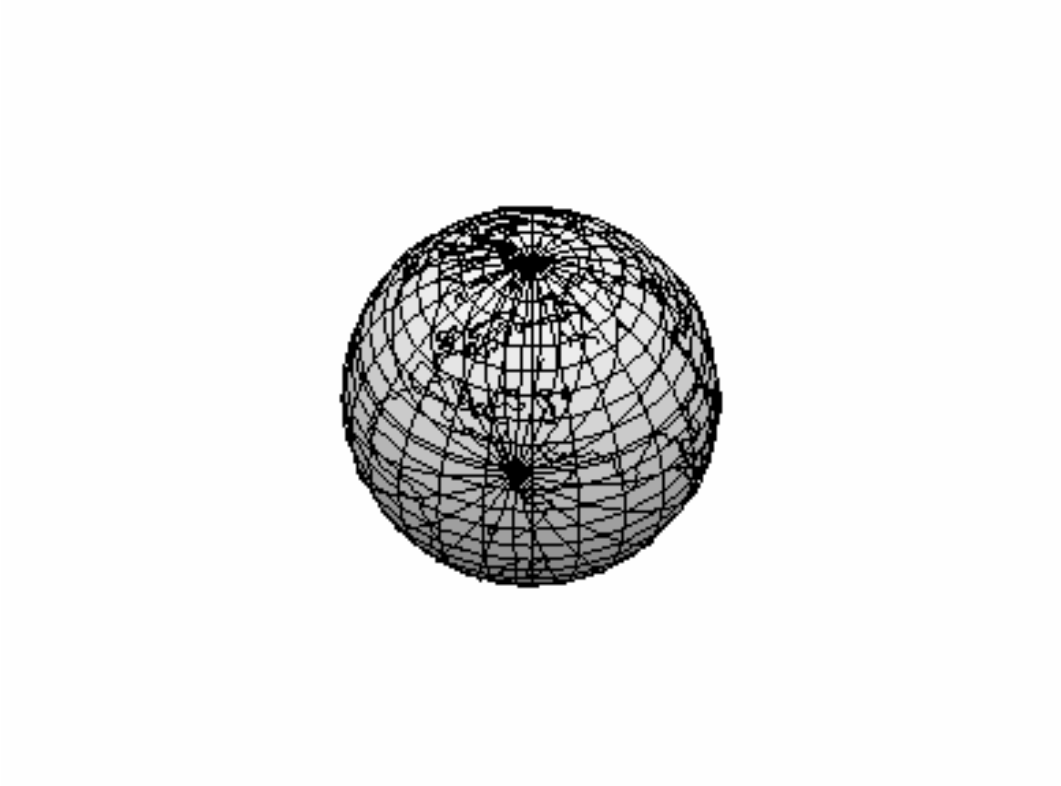
Rectangular Projections (a simple misconception)

Many people make the simple mistake of thinking about directions using a map of the world based on a rectangular grid like the one shown below.
(X0, Y0) is the location of Makka in degrees, and (x0,y0) is in radians.

```
> plots[display]({boxpolar(1),world[50]});
```

The problem is that the world is not flat, and in fact *can not* be mapped onto a flat surface without introducing distortion in angles and lines. Thus the angles as seen in this map are deceptive. The same lines are shown on a globe below.

```
> globe(plots[display]({boxpolar(1),world[50]}));
```



While near Makka these line seem reasonable, near the poles, and in the pacific ocean (New Zealand), and especially near Alaska, we see some very odd behavior.
>

Azimuthal Projections (a complex mistake)

An azimuthal projection centered at Makka is best described by just showing it, see below.

Computations

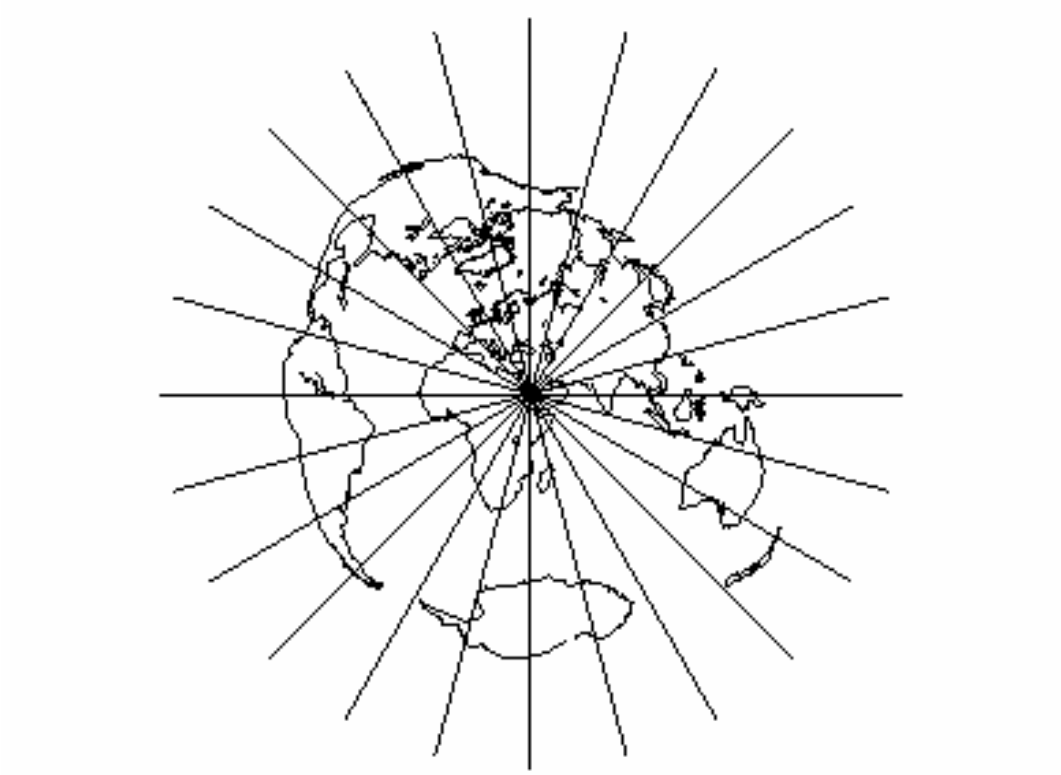
QMd is the direction from Makka to a point (not used directly, but defined here for completeness), and Qr (defined later) is its distance from Makka.
The function we use is QMp, which is equivalent to Qr * [cos(QMd), sin(QMd)], but is defined directly in a simplified form.

```
> QMd := (x,y) -> arctan(cos(y0)*tan(y)-sin(y0)*cos(x-
x0),sin(x-x0)):
QMp := (x,y) -> arccos(sin(y0)*sin(y) +
cos(y0)*cos(y)*cos(x-x0))
   / sqrt( (cos(y0)*tan(y)-sin(y0)*cos(x-x0))^2 + sin(x-
x0)^2 )
   * [ sin(x-x0), cos(y0)*tan(y)-sin(y0)*cos(x-x0) ]:
QMD := (x,y) -> QMd(x*Pi/180, y*Pi/180) * 180/Pi:
QMP := (x,y) -> 180/Pi*QMp(x*Pi/180,y*Pi/180):
>
```

Plain Azimuthal map

```
>
plots[display](polargrid(180),plottools[transform](QMP)(wor
```

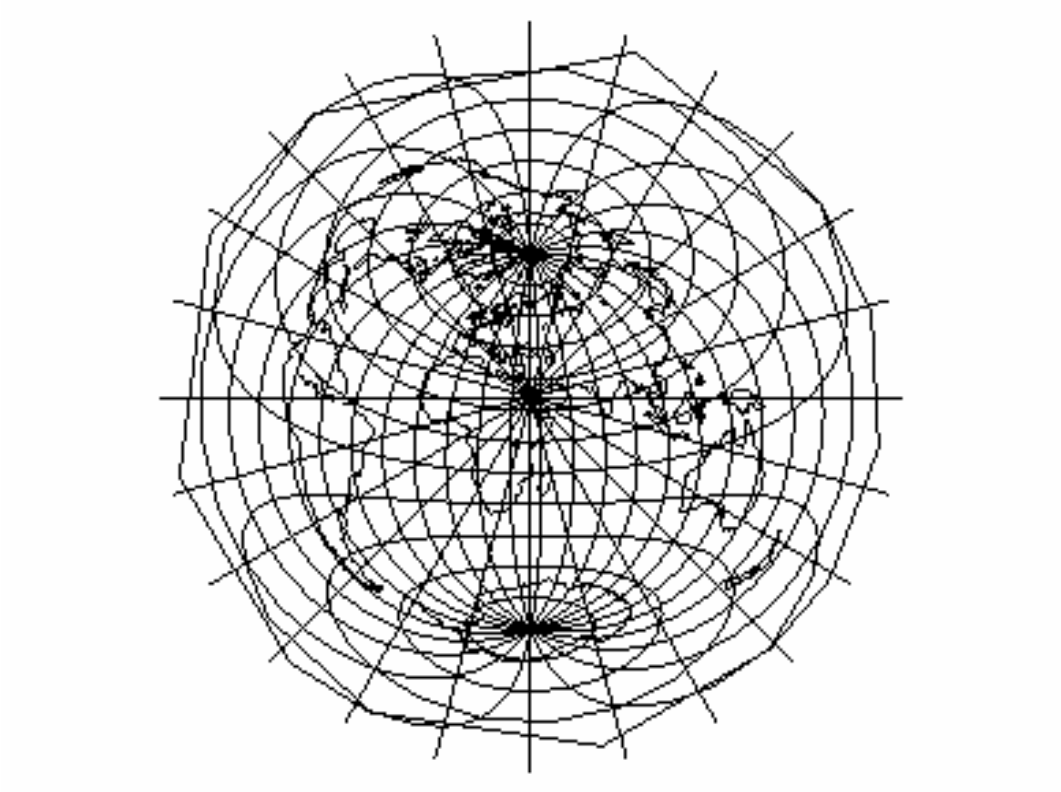
```
ld[50]));
```



What we have done in the map is to place Mekka in the center of a polar co-cordinate map. Every point is mapped with its distance being exactly the distance from Mekka (measured in degrees from 0 to 180), and its angle being the direction of that point from Mekka.
The problem is that an azimuthal projection gives the correct direction *from* Mekka to every point on the globe, not the direction from any point *to* Mekka. That does not seem to be such a large problem, because we could just turn that direction around, until we realize that if a location is directly East of Mekka, that does not mean that Mekka is directly West of that location!
This map is useful, in that the lines from Makka are the lines that one would line up on, if one were to face Makka. While one can orient oneself in any location with regards to other global landmarks (like coastlines), this map is not very helpful in giving the directions in term of compass directions.
If we superimpose the latitude and longitude lines on the same map, we can deduce directions better, although the map looks quite crowded.
>

Azimuthal map with latitude and longitude lines

```
> QMP := (x,y) -> 180/Pi*QMp(x*Pi/180,min(89.9,max(-
89.9,y))*Pi/180):
plots[display](polargrid(180),plottools[transform](QMP)(wor
ld[50,15]));
```



>
So an azimuthal map aligns all the points which are in the same direction *from Makka.* What we really want is a map that aligns all the points where Makka is the same direction from that point..
>

A Naive Kaaba Centered Map

Another possible depiction is to draw a map with the Kaaba at the center. All the points where one must face north, to face Kaaba would be drawn directly to the south of Kaabah. Similarly every point would be placed at a location such that the angle of the straight line joining that point to the center would be the Qibla angle.
Mathematically, this corresponds to mapping a point with (long, lat) = (x, y) to polar coordinates (r,theta) with theta given as 180 + Qibla Direction(x,y), and r(x,y) being any function.
If we let r(x,y) be the distance from Makka, we get an interesting graph, which takes a while to interpret.

Computations

We use the convention that functions with lower case letters refer to functions that take arguments and return values in radians,while capital latters will refer to functions that take arguments and return values in degrees.
Qd(x,y) = direction from [x,y] to Kaaba (this function is just for illustration)
Qr(x,y) = distance from [x,y] to Kaaba (also for illustration)
Qp(x,y) = Qr * [cos Qd, sin Qd] is the polar point form of the (distance, angle) to Kaaba. The actual formula used here does some simplifications, and so is independent of the

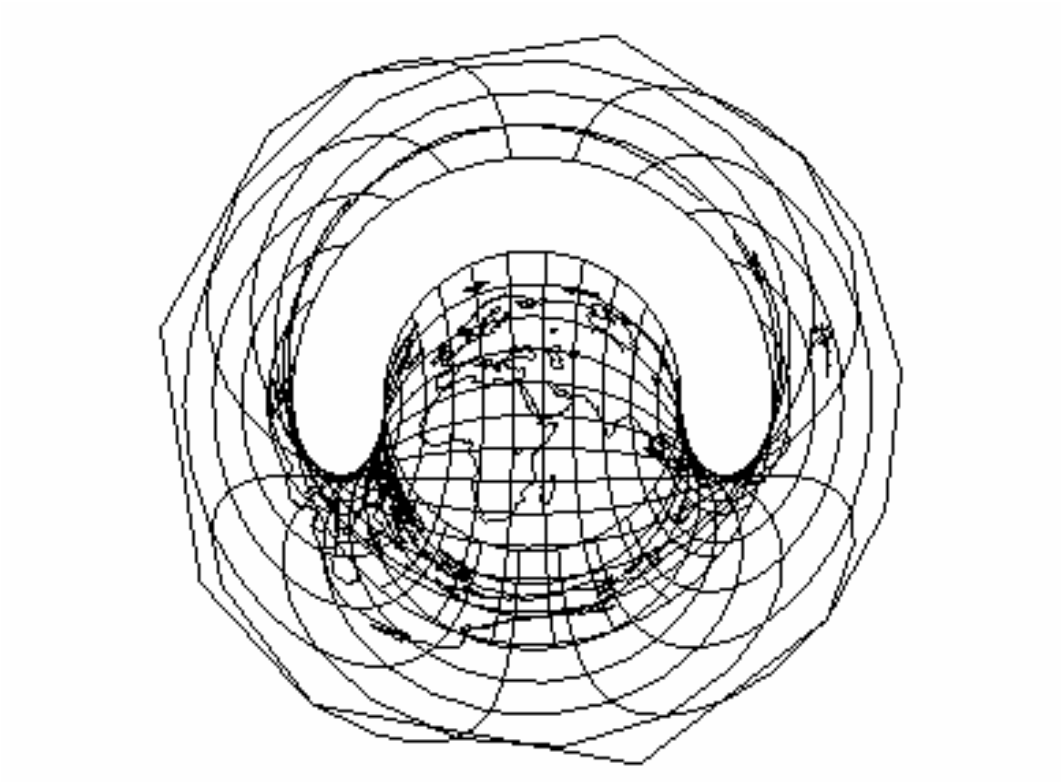
above formulas.

```
> Qd := (x,y) -> arctan(cos(y)*tan(y0)-sin(y)*cos(x0-
x),sin(x0-x)):
Qr := (x,y) -> arccos(sin(y)*sin(y0) +
cos(y)*cos(y0)*cos(x0-x)):
Qp :=  (x,y) -> arccos(sin(y)*sin(y0) +
cos(y)*cos(y0)*cos(x0-x))
   / sqrt( (cos(y)*tan(y0)-sin(y)*cos(x0-x))^2 + sin(x0-
x)^2 )
   * [ sin(x0-x), cos(y)*tan(y0)-sin(y)*cos(x0-x) ]:
QD := (x,y) -> Qd(x*Pi/180, y*Pi/180) * 180/Pi:
QR := (x,y) -> Qr(x*Pi/180, y*Pi/180) * 180/Pi:
QP := (x,y) -> 180/Pi*Qp(x*Pi/180,y*Pi/180):
>
```

A first Polar Map

The idea is to draw a similar polar-coordinates map, with (r,theta) being the distance from Makka and the opposite of (i.e. 180 plus) the angle to Makka. That way, points where one faces south to face Makka, will be drawn north of Makka, as they should.

```
> QP0 := (x,y) -> -Qp(x*Pi/180, y*Pi/180) * 180/Pi:
> plottools[transform](QP0)(world[50,15]);
```



While the round central part of the the map looks quite nice and simple, the rest of it seems like a mess.

The map also contains many surprizes. Trying to decipher what happened to the Americas is difficult.

What is the white sausage shape doing in the upper half of the map? (Pork is forbidden in Islam, and I have no intention to poke fun at Islam, being a practising Muslim myself, but the shape does resemble most closely a sausage!)
We clarify this map in the next section.

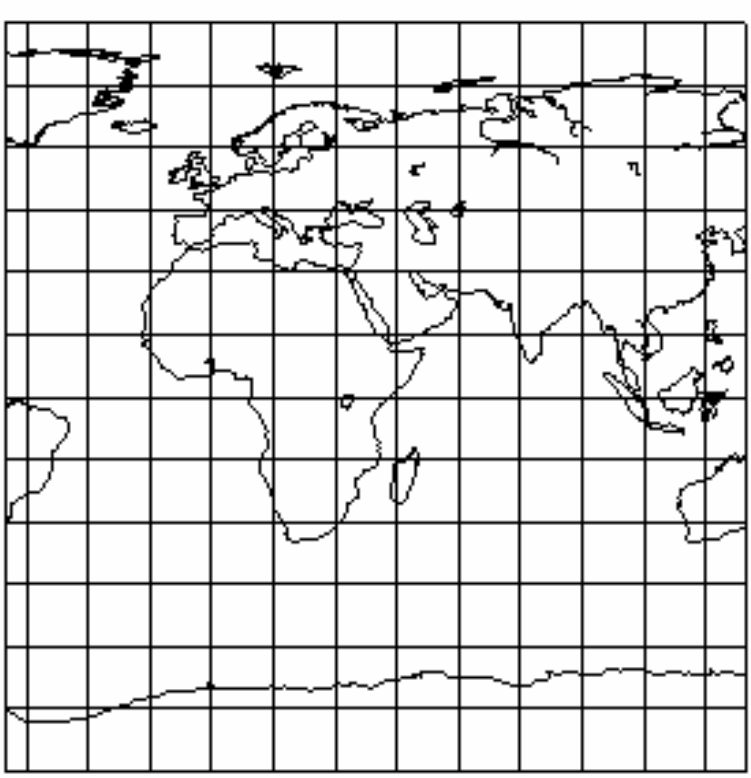
Sorting out the mess

Imagine a person standing at the north pole, facing Makkah. Imagine four other people, one in front of him, one behind him, one to the left and one to the right, all facing the same way. The person in front of him is facing due south. But the person behind him is facing the north pole, ie facing due north! Also the person to his right, has the north pole to their left, hence the are facing east. The last person, by a similar argument is facing west! And yet they are all facing the same way.
The implication of this is that both the poles are represented as circles. They are places where one can face in any direction, and are a fixed distance from Makka.
The north pole is closer to Makka, and is represented by the complete inner circle, which shares a boundary with the inner boundary of the "sausage".
The south pole is the larger circle that shares the outer boundary of the "sausage".
So the meaning of the "sausage" being blank is that there is no place in the world further than the North Pole from Makka, and closer than the South Pole, where people would face in any southern direction to face Makka!
In fact, because of this, the land within this belt appears confused on the map. The confusion is due to the fact that within this belt, there are two points on the globe which are the same distance from Makka, and which pray facing the same direction. Thus the area of the world that should have been in the "sausage" has actually been overlaid on the lower half of the circle that would complete the "sausage".
To remove this confusion, we separate the world into two hemispheres.
Q1 keeps only the points that are within 90° longitude of Makka. Q3 rotates the world by 180°, and then keeps the same points, so it the other half of the world.
QP1 is the same as QP0, restricted to the Makka hemisphere, and
QP2 is the same as QP0 restricted to the Anti-Makka hemisphere.
QP3 rotates the world by 180°, and then keeps what is now the Anti-Makka hemisphere.

```
> Q1 := (x,y) -> [min(X0+90,max(X0-90,x)),y]:
Q3 := (x,y) -> [`if`(x>0,max(X0-90,x-
180),min(X0+90,x+180)),y]:
QP1 := (x,y) -> -180/Pi*Qp(min(X0+90,max(X0-
90,x))*Pi/180,y*Pi/180):
QP2 := (x,y) -> -180/Pi*Qp(`if`(x>0,max(X0+90,x),min(X0-
90,x))*Pi/180,y*Pi/180):
QP3 := (x,y) -> -180/Pi*Qp(`if`(x>0,max(X0-90,x-
180),min(X0+90,x+180))*Pi/180,y*Pi/180):
>
```

The Macca hemisphere

```
> plottools[transform](Q1)(world[50,15]);
```



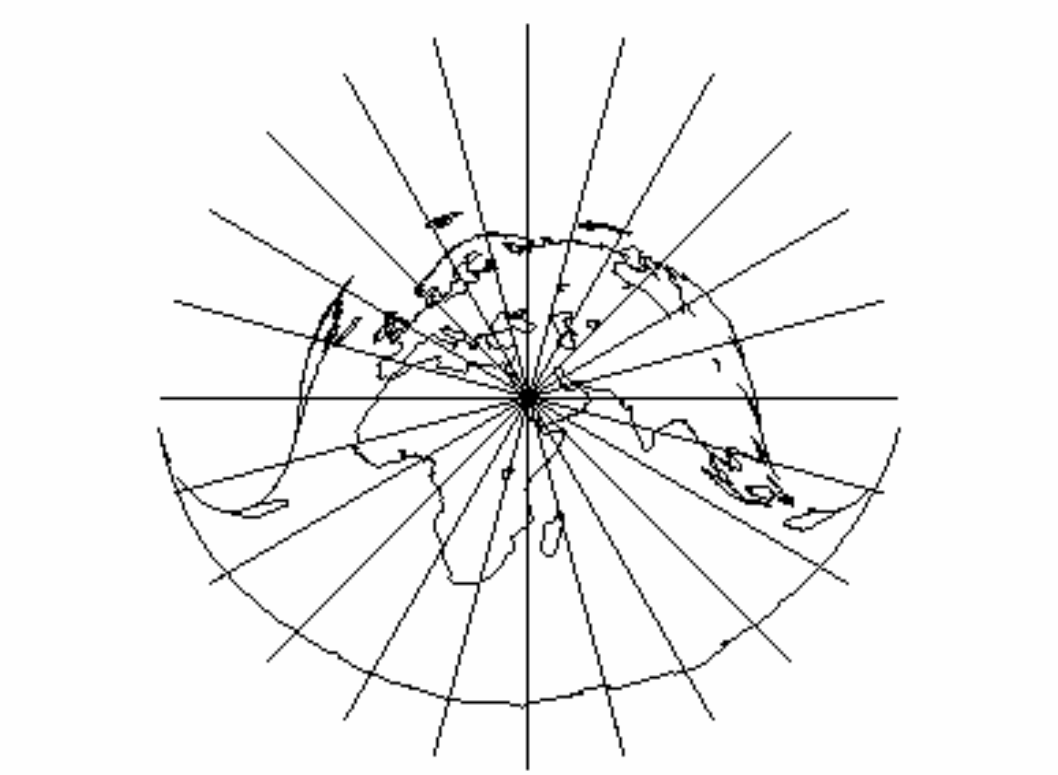
This next step takes some time, because the transformations are slow to compute.

```
> evalf(QR(0,-90));
```

111.4233330

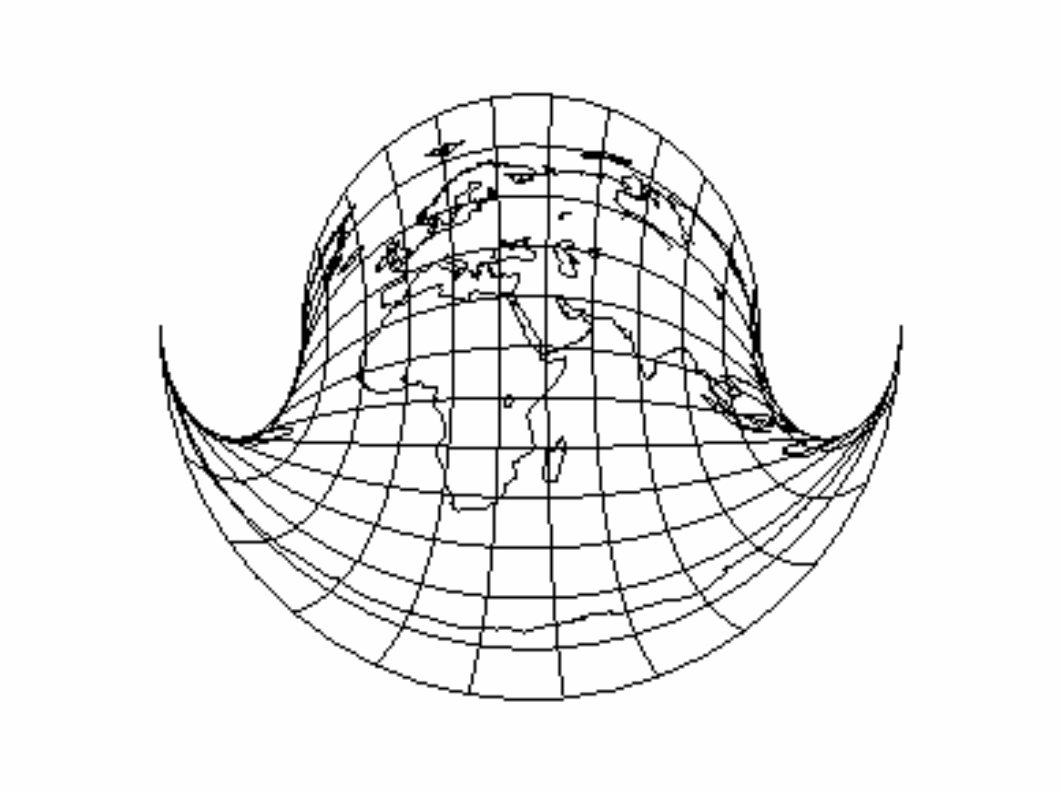
>

```
plots[display](polargrid(110),plottools[transform](QP1)(wor
ld[50]));
```



The line at the bottom of this map is half of the boundary of the continent of Antarctica. We repeat this map with the latitude longitude grid thrown in, to show the amount of distortion that is happening near the edges of the graph. In fact, the upper semicircle represents the north pole, the lower semicircle is the south pole, and the two small semi-ciruclar arcs connecting these on either side are the two longitude lines between the poles on either side of the map.

```
> plottools[transform](QP1)(world[50,15]);
```



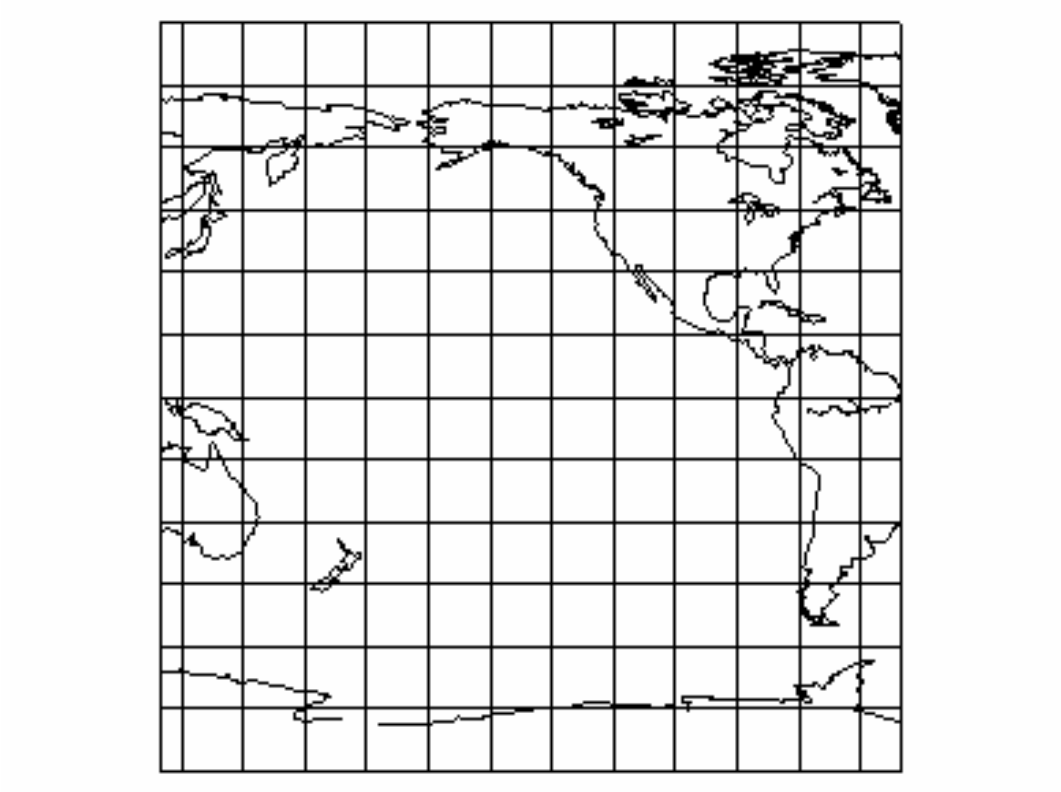
>

The Anti-Makka Hemisphere

For the other half of the world, besides the Americas, there is a lot of water.

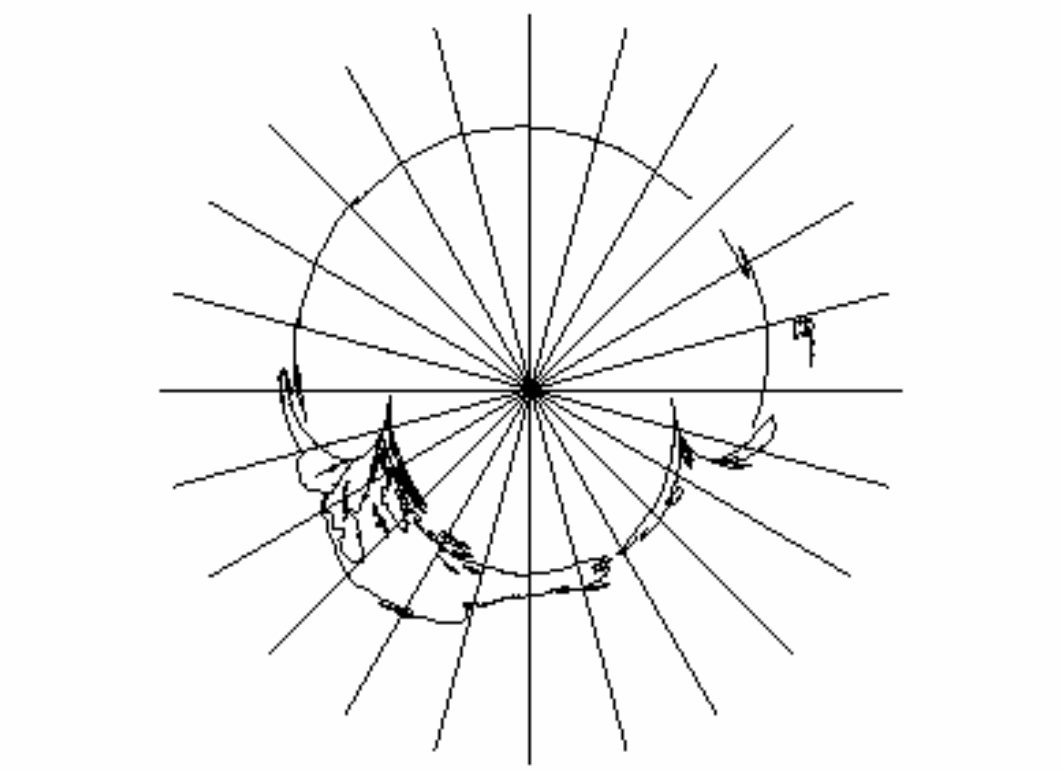
>

```
`Maps/removelines`(plottools[transform](Q3)(world[50,15]));
```



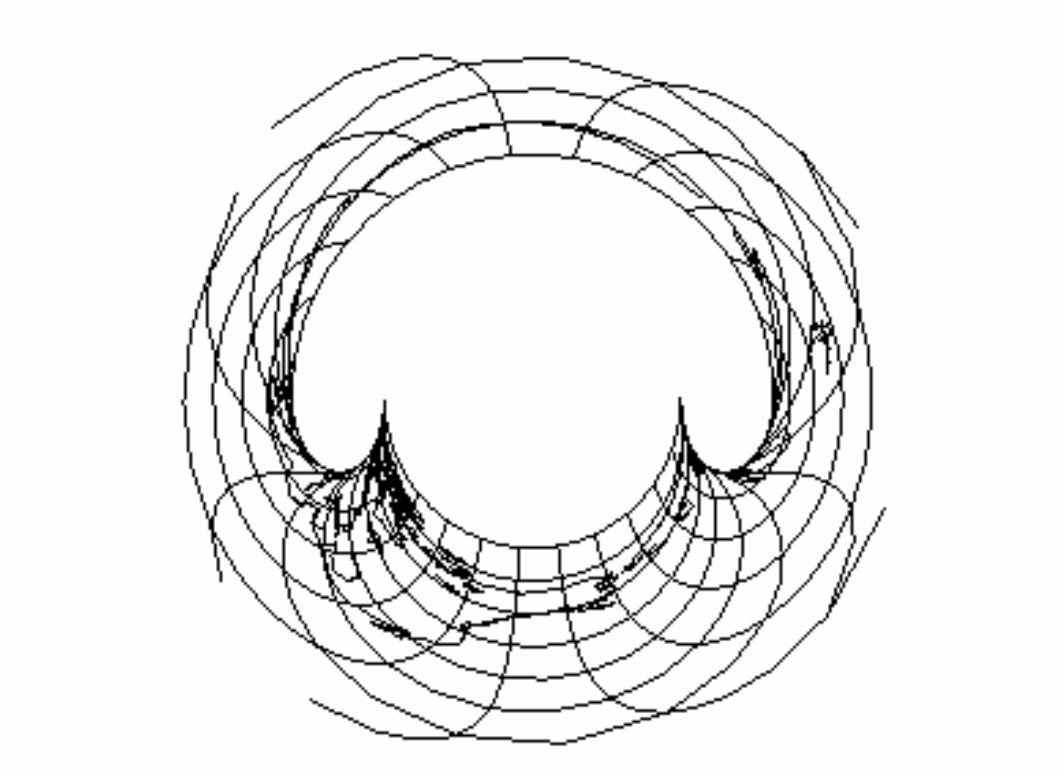
>

```
plots[display](polargrid(180),`Maps/removelines`(plottools[
transform](QP2)(world[50])));
```



The line at the top this map is the other half of the boundary of Antarctica.
Same map once again, this time with latitude and longitude lines
>

```
`Maps/removelines`(plottools[transform](QP2)(world[50,15]))
;
```

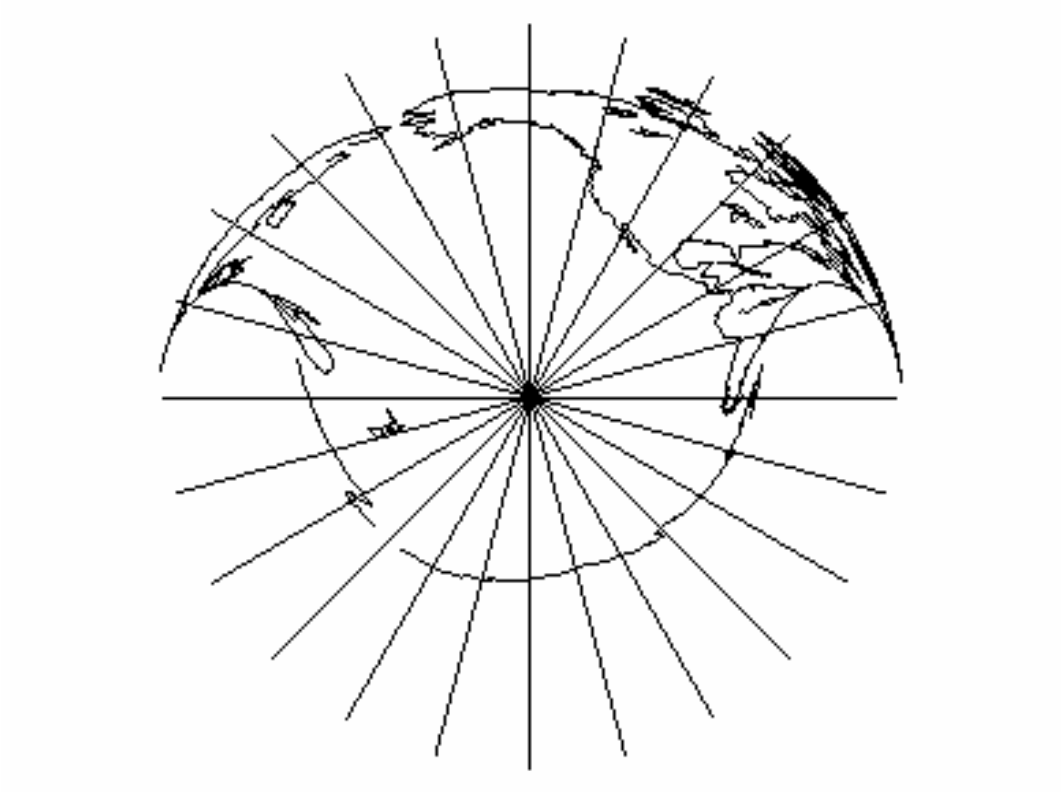


While the first plot (Makka hemisphere) is comprehensible to some degree, the second plot (Anti-Makka) is still quite strange. One thing to notice is that it is a mirror image map! Alaska is to the right, while California is to the left.
>

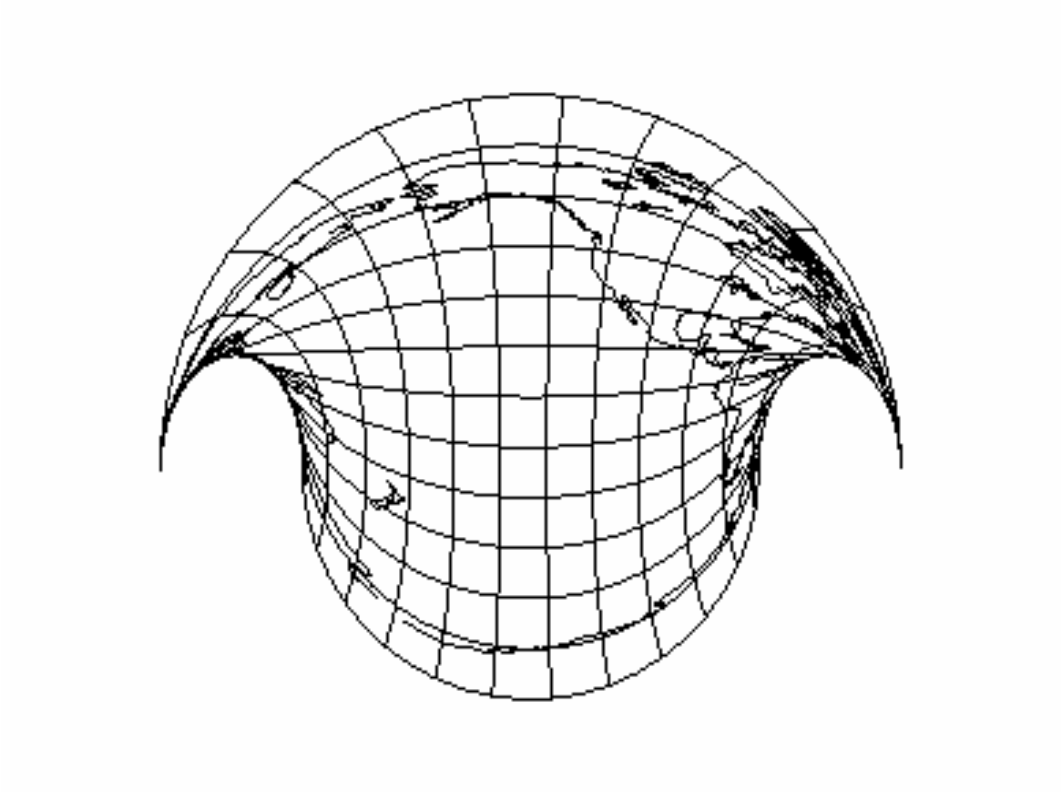
The mirror-Anti-Makka

One way to fix the problem faced in the Anit-Makka hemisphere map is to simply draw the same map, inside-out, so that the Anti-Makka point is at the center, and the direction lines are radiating out of it.

```
> Y0 := -Y0: y0 := -y0:
plots[display](polargrid(110),`Maps/removelines`(plottools[
transform](QP3)(world[50])));
Y0 := -Y0: y0 := -y0:
```



```
> Y0 := -Y0: y0 := -y0:
`Maps/removelines`(plottools[transform](QP3)(world[50,15]))
;
Y0 := -Y0: y0 := -y0:
```



>
In this map, the direction to Makka is given by following the radial line to the outside.

IsoDirectional Lines

Trying to straighten out the lines of direction by distorting the globe turned out to create an ugly map. An better looking map results from leaving the globe alone, and bending the directional lines instead.
Our goal then is to find the curve that joins together all the points on the globe where one faces due east to point to Makka. Similarly for all the other directions.

A verbal description of singularities

Kaaba (and anti-Kaabah)
It is obvious that all IsoDirectional lines must come together at the Kaaba (where the prayer lines are in fact circles, as shown in the front picture). This is because people near the Kaaba pray in all possible directions.
It is probably harder to understand that at the point directly opposite Kaabah, people would stand in a circle facing outward! Fortunately, this problem is not encountered, because this point is located deep within the Pacific Ocean.
The North Pole (and the South Pole)
As mentioned before, all IsoDirectional lines must also come together at the poles. Unlike the previous points, the fact that all the IsoDirectional lines come together at the poles is simply a problem with the form of measurement that we are using, and not a discontinuity in the "actual" direction that people are facing.

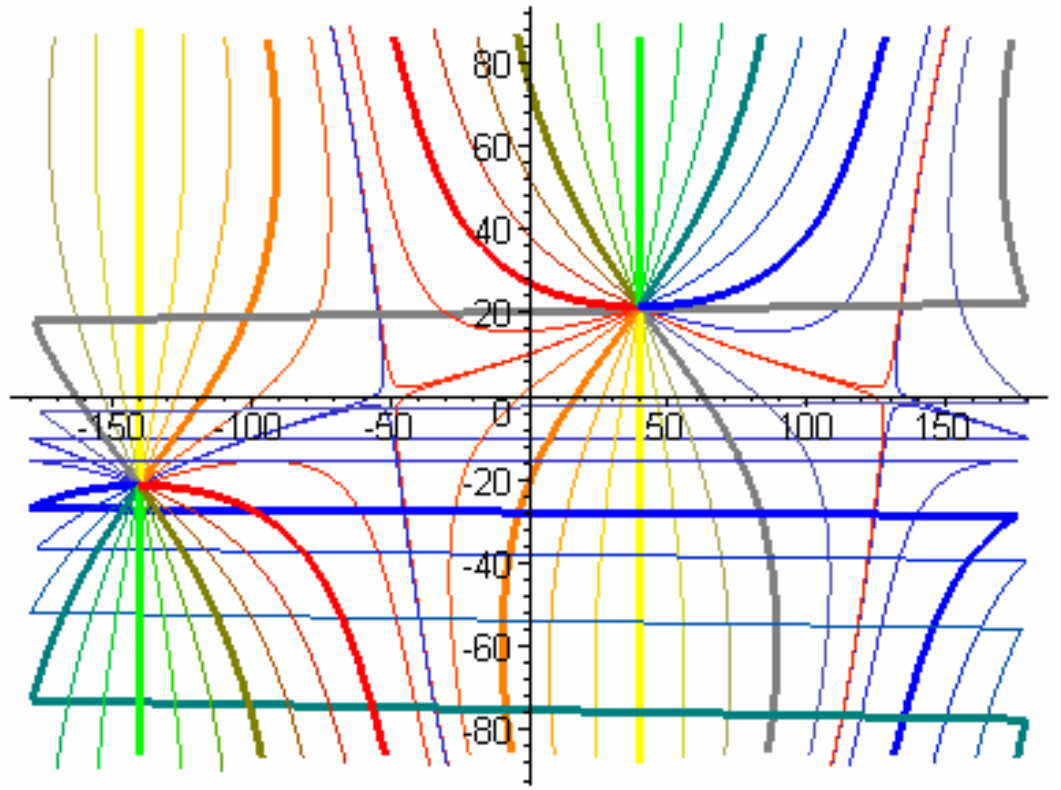
Iso Directional Computations

Contour lines

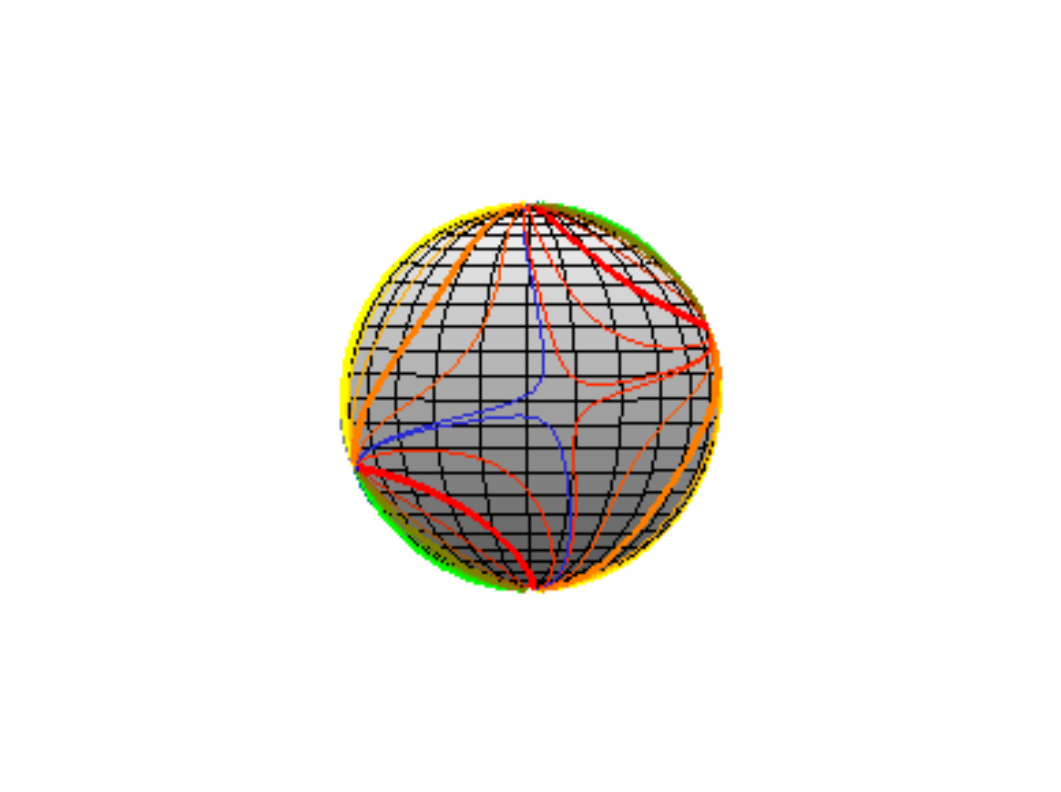
While Maple has built-in contour plots, I did not like the results, so I am doing them myself.

```
> Contour := proc(T,D,C)
local t, d, xt, yt, ut, lst;
t := T*Pi/180: d:=D*Pi/180:
xt:=x0; yt:=y0; ut:=t+Pi; lst:=evalf([x0,y0]*180/Pi);
while evalf(abs(yt)<Pi/2-d) do
  ut := op(2,op(
    fsolve({Qd(xt+d*cos(u),yt+d*sin(u))=t},{u=ut},(ut-
Pi+1)..(ut+Pi-1))
  ));
  xt := xt + d*cos(ut);
  yt := yt + d*sin(ut);
  lst := lst , evalf([xt,yt]*180/Pi);
end do:
return CURVES([lst],map(x->[`if`(x[1]>0,x[1]-180,180+x[1]),
-x[2]],[lst]),C):
end proc:
> c := PLOT():
rgb:=[
[0,0,6],[0,1,5],[0,2,4],[0,3,3],[0,4,2],[0,5,1],
[0,6,0],[1,5,0],[2,4,0],[3,3,0],[4,2,0],[5,1,0],
[6,0,0],[6,1,0],[6,2,0],[6,3,0],[6,4,0],[6,5,0],
[6,6,0],[5,5,1],[4,4,2],[3,3,3],[2,2,4],[1,1,5],
[0,0,6],[0,1,5],[0,2,4],[0,3,3],[0,4,2],[0,5,1],
[0,6,0],[1,5,0],[2,4,0],[3,3,0],[4,2,0],[5,1,0],[6,0,0]
]/6:
for i from -12 to 11 do
  c:=plots[display](c,PLOT(Contour(i*15,5,
    COLOR(RGB,rgb[13+i][1],rgb[13+i][2],rgb[13+i][3],
              rgb[25-i][1],rgb[25-i][2],rgb[25-i][3])),
    THICKNESS(`if`(irem(i,3)=0,3,0))
  )):
end do:
  c:=plots[display](c,PLOT(Contour(180-Y0-0.1,1,
    COLOR(RGB,rgb[13+1][1],rgb[13+1][2],rgb[13+1][3],
              rgb[25-1][1],rgb[25-1][2],rgb[25-1][3])),
    THICKNESS(`if`(irem(1,3)=0,3,0))
  )):
  c:=plots[display](c,PLOT(Contour(180-Y0+0.1,1,
    COLOR(RGB,rgb[13+1][1],rgb[13+1][2],rgb[13+1][3],
              rgb[25-1][1],rgb[25-1][2],rgb[25-1][3])),
    THICKNESS(`if`(irem(1,3)=0,3,0))
  )):
  c:=plots[display](c,PLOT(Contour(Y0-0.1,1,
    COLOR(RGB,rgb[13+1][1],rgb[13+1][2],rgb[13+1][3],
```

```
                rgb[25-1][1],rgb[25-1][2],rgb[25-1][3])),
    THICKNESS(`if`(irem(1,3)=0,3,0))
  )):
  c:=plots[display](c,PLOT(Contour(Y0+0.1,1,
    COLOR(RGB,rgb[13+1][1],rgb[13+1][2],rgb[13+1][3],
                rgb[25-1][1],rgb[25-1][2],rgb[25-1][3])),
    THICKNESS(`if`(irem(1,3)=0,3,0))
  )):
> plots[display](c);
```




```
> globe(plots[display](c));
```



>

Exact computations

First we try to find the exact equation for the line which splits north-pole lines from the south pole ones. It appears to be the line which is +y0 or pi - y0 degrees. From some trig computations we can solve this to get

Proof of equation

Here is the proof of the equation

```
> subs(tan(y0)=t0,tan(Qd(x+x0,y))=tan(y0));
```

$$-\frac{\cos(y)\,t0-\sin(y)\cos(x)}{\sin(x)}=t0$$

You may optionally do this step...

```
> op(1,%) = Pi - op(2,%);
```

$$-\frac{\cos(y)\,t0-\sin(y)\cos(x)}{\sin(x)}=\pi+t0$$

```
> subs((sin(y)*cos(x))/sin(x)=sin(y)/tan(x),expand(%));
```

$$-\frac{\cos(y)\,t0}{\sin(x)}+\frac{\sin(y)}{\tan(x)}=t0$$

Here we have two alternate paths... Either choice works out to the same final answer.

```
> solve(sin(x)^2/(1-sin(x)^2)=tan(x)^2,sin(x));
```

$$\frac{\tan(x)}{\sqrt{\tan(x)^2+1}},-\frac{\tan(x)}{\sqrt{\tan(x)^2+1}}$$

```
> subs(sin(x)=-tan(x)/sqrt(1+tan(x)^2),%%);
```

$$\frac{\sqrt{\tan(x)^2+1}\,\cos(y)\,t0}{\tan(x)}+\frac{\sin(y)}{\tan(x)}=t0$$

```
> expand(map(u->u*tan(x),%));
```

$$\sqrt{\tan(x)^2+1}\,\cos(y)\,t0+\sin(y)=t0\,\tan(x)$$

```
> map(u->u-sin(y),%);
```

$$\sqrt{\tan(x)^2+1}\,\cos(y)\,t0=t0\,\tan(x)-\sin(y)$$

```
> expand(map(u->u^2,%));
```

$$\cos(y)^2\,t0^2\,\tan(x)^2+\cos(y)^2\,t0^2=t0^2\,\tan(x)^2-2\,t0\,\tan(x)\,\sin(y)+\sin(y)^2$$

```
> collect(op(1,%)-op(2,%),tan(x));
```

$$(-t0^2+\cos(y)^2\,t0^2)\,\tan(x)^2+2\,t0\,\tan(x)\,\sin(y)-\sin(y)^2+\cos(y)^2\,t0^2$$

```
> solve(%=0,x);
```

$$\arctan\left(\frac{1+\sqrt{1-\sin(y)^2+t0^2-t0^2\,\sin(y)^2}}{\sin(y)\,t0}\right),\quad \arctan\left(\frac{1-\sqrt{1-\sin(y)^2+t0^2-t0^2\,\sin(y)^2}}{\sin(y)\,t0}\right)$$

```
> x[10]:=simplify(%[1]); x[11]:=simplify(%%[2]);
```

$$x_{10}:=\arctan\left(\frac{1+\sqrt{\cos(y)^2\,(1+t0^2)}}{\sin(y)\,t0}\right)$$

$$x_{11}:=-\arctan\left(\frac{-1+\sqrt{\cos(y)^2\,(1+t0^2)}}{t0\,\sin(y)}\right)$$

```
> assume(cos(y)>0); assume(cos(t0)>0); assume(cos(y[0])>0);
x[10] := simplify(subs(t0=tan(y[0]),x[10])):
x[11] := simplify(subs(t0=tan(y[0]),x[11])):
y:='y': t0:='t0': y[0]:='y[0]':
x[10]; x[11];
```

$$\arctan\left(\frac{\cos(y\sim_0)+\cos(y\sim)}{\sin(y\sim)\,\sin(y\sim_0)}\right)$$

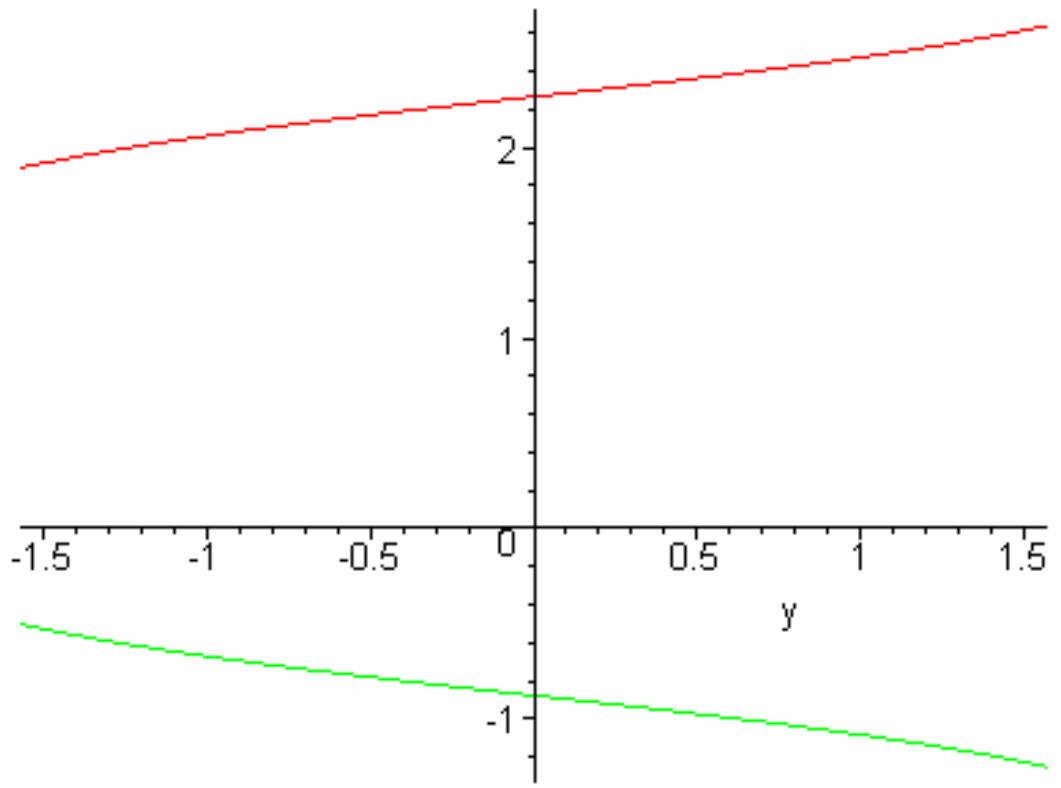
$$-\arctan\left(\frac{-\cos(y\sim_0)+\cos(y\sim)}{\sin(y\sim_0)\,\sin(y\sim)}\right)$$

```
> iso[1] := y -> x0 + `if`(y<0,0,Pi) -
arctan((cos(y0)+cos(y))/(sin(y)*sin(y0)));
iso[2] := y -> x0 - `if`(y<0,0,Pi) +
arctan((cos(y0)+cos(y))/(sin(y)*sin(y0)));
```

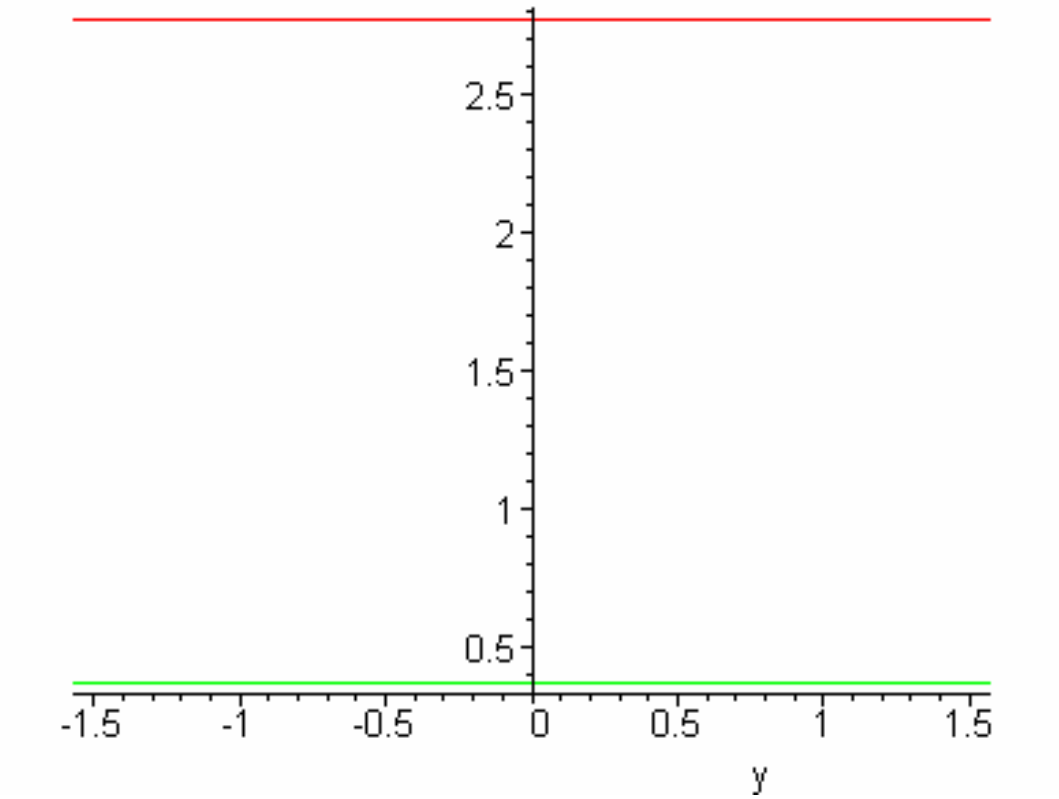
$$iso_1:=y\to x0+\text{`if`}(y<0,0,\pi)-\arctan\left(\frac{\cos(y0)+\cos(y)}{\sin(y)\,\sin(y0)}\right)$$

$$iso_2 := y \to x0 - \text{`if`}(y < 0, 0, \pi) + \arctan\left(\frac{\cos(y0) + \cos(y)}{\sin(y)\,\sin(y0)}\right)$$

```
> plot([iso[1](y),iso[2](y)],y=-Pi/2..Pi/2);
```



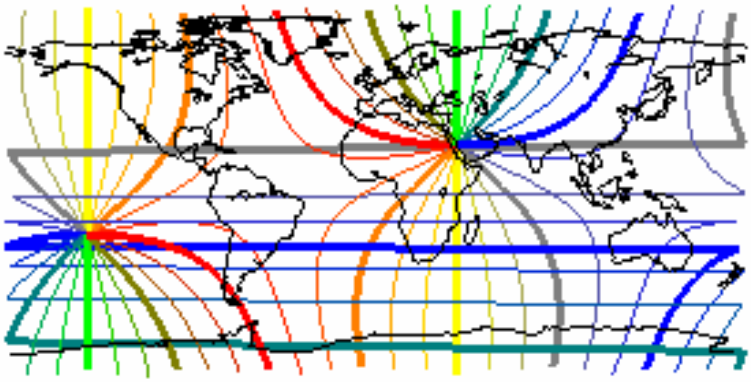

```
> plot([Qd(iso[1](y),y),Qd(iso[2](y),y)],y=-Pi/2..Pi/2);
```



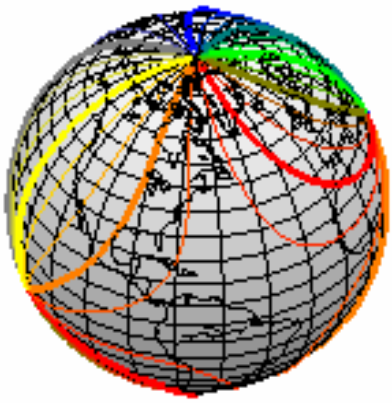
>
>

Contour Maps

```
> plots[display]({world[50],c});
```



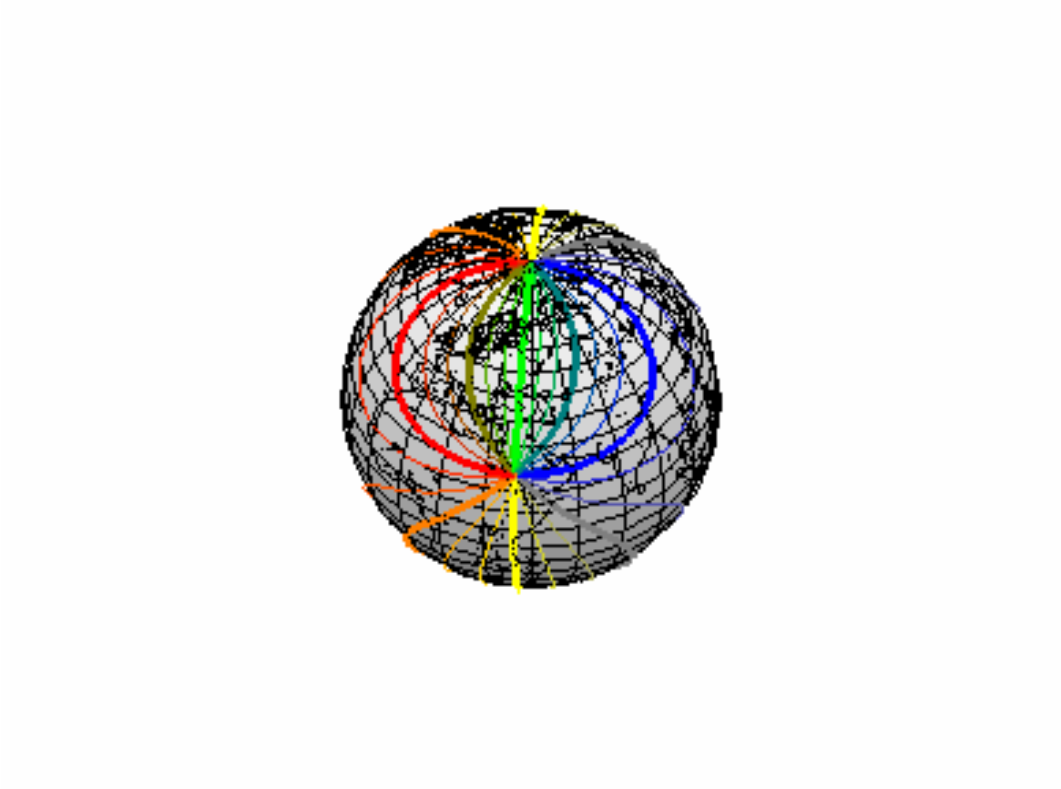
```
> globe(plots[display]({world[50],c}));
```



>

An explanation of the Sausage

The globe clearly explains the origin of the sausage shaped white area observer in the earlier maps. Most of the lines where one prays in a southerly direction go from the Kaaba, and eventually meet at the North Pole. There they disappear, only to reappear at the South Pole. Thus for distances that are further than the North Pole and closer than the South Pole, there are no locations where the Qibla direction has any southern component. Similarly, while the lines with a northern component all continue until they meet at the South Pole, lines with northern components of direction also appear at the North Pole.This causes the map of Northern Canada to be overlaid on the map of Antarctica.
>
High detail contour map
> `globe(plots[display]({c,world[1867]}));`



>
>
>
This shows the strangness of the isodirectional lines. If we naively try to straighten them out, we run into problems, as we shall see in the next section.

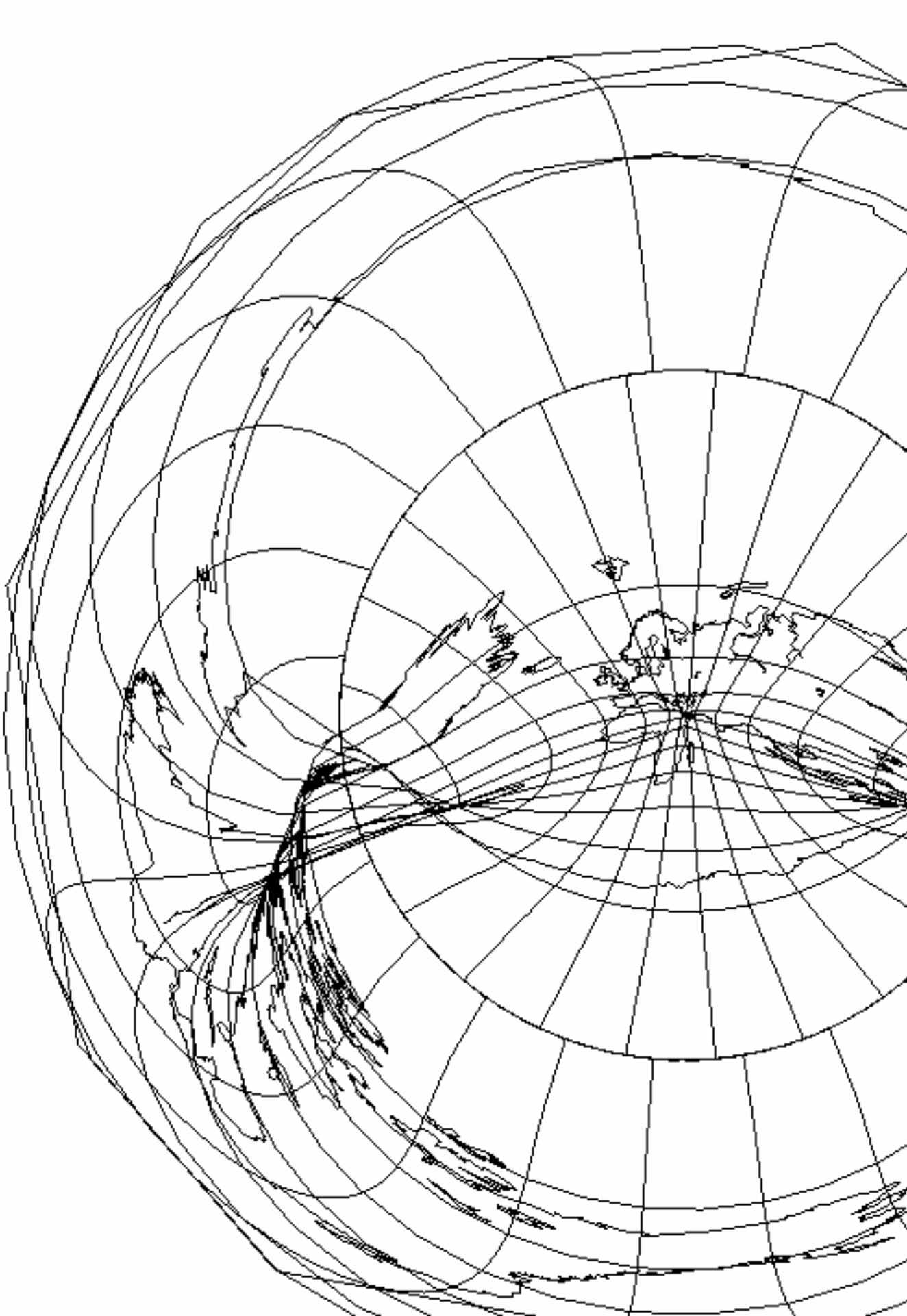
Snippets that didn't make it

Mapping both poles to the same "distance" An intetesting but futile excercise.

Draw the world with the center at the origin, the poles on the x-axis, and Kaaba (with max z) on the x-z plane.
The plane tangent to Kaaba intersects the x axis at a point. Draw a line parallel to the y

axis at that point.
For any point (x,y), define QD to be the angle between plane that goes through this point and the parallel line, and the plane that is tangent to Makka.
This puts both poles at the same "distance" from Makka.
It is useful to put a "warp" in this distance, to magnify some and shrink other distances.

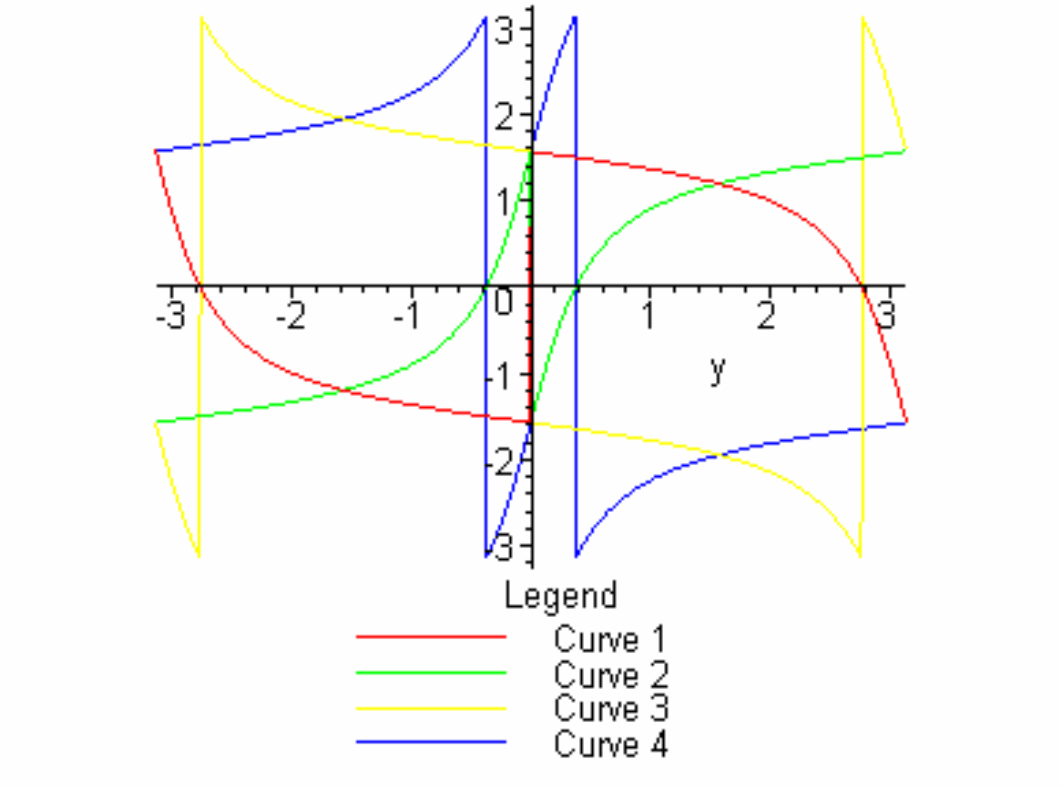
```
> A0:=X0-90: A1:=A0+180:
warp0 := x -> sqrt(y0)-sqrt(x):
warp1 := x -> sqrt(y0)+sqrt(-x):
QD0 := (x,y) -> warp0(arctan( cos(y) * cos(x-x0) / (
1/sin(y0) - sin(y) ))):
QD1 := (x,y) -> warp1(arctan( cos(y) * cos(x-x0) / (
1/sin(y0) + sin(y) ))):
Qpx := (x,y) -> [ sin(x0-x), cos(y)*tan(y0)-sin(y)*cos(x0-
x) ]
   / sqrt( (cos(y)*tan(y0)-sin(y)*cos(x0-x))^2 + sin(x0-
x)^2 ):
QDx := (x,y) -> `if`(x>A0 and x<A1,QD0(x*Pi/180, y*Pi/180),
QD1(x*Pi/180, y*Pi/180)):
QPx := (x,y) -> -`if`(x>A0 and x<A1,QD0(x*Pi/180,
y*Pi/180), QD1(x*Pi/180, y*Pi/180))
   * Qpx(x*Pi/180, y*Pi/180):
evalf(QPx(40,20));
                    [.00001140576212, -.00009870781289]

> plottools[transform](QPx)(world[50,15]);
```



Clearly one can do better by choosing a better warp function, but really this map is too twisted to be useful.
Antarctica is on the inside bottom and the outside top of the polar circle. The boundary lines are continuous, even though the upper half of it is inside out! The lands outside the polar circles are mirror images. There is still a doubled map near NW Canada and Australia.
I give up trying to straighten out this mess.
>

Some more proof

```
> plot([arctan((cos(y0)+cos(y))/(sin(y)*sin(y0))),
       arctan((cos(y0)-cos(y))/(sin(y)*sin(y0))),
       arctan(-(cos(y0)+cos(y))/(sin(y)*sin(y0)),-1),
       arctan(-(cos(y0)-cos(y))/(sin(y)*sin(y0)),-1)],y=-
Pi..Pi);
```



>
>

References and Definitions

I have tried to give references that were available on the web.

Qibla Direction

Kamal Abdali (http://www.patriot.net/users/abdali/ftp/qibla.pdf) gives the formula for Qibla direction as:

Q(x,y) = atan2[sin (-dx) , (cos y)(tan y0) - (sin y)(cos dx)]
Where:
y = Latitude of Desired Location (North is positive)
x = Longitude of Desired Location (East is positive)
y0 = Latitude of Mecca (+21.423333° north of Equator)
x0 = Longitude of Mecca (+39.823333° east of Prime Meridian)
dx = x0 - x
dy = y0 - y

Note atan2(y,x) = atan(y/x), with the angle chosen so that the sin and cos of the chosen angle have the same signs as the numerator and denominator of the fraction.

The answer will be an angle between -180 and +180 indicating the degrees North of East.

```
> Qd := (x,y) -> arctan((cos(y)*tan(y0)-sin(y)*cos(x0-
x)),sin(x0-x));
```

$$Qd := (x, y) \rightarrow \arctan(\cos(y)\tan(y0) - \sin(y)\cos(x0 - x), \sin(x0 - x))$$

Location of Kaaba

From the previous reference we also get

```
> X0 := 39.823333; Y0:= 21.423333;
```

$$X0 := 39.823333$$

$$Y0 := 21.423333$$

```
> x0 := X0*Pi/180; y0:= Y0*Pi/180;
```

$$x0 := .2212407389\ \pi$$

$$y0 := .1190185167\ \pi$$

Distance from Mecca

Quoted from: http://www.fcaglp.unlp.edu.ar/~esuarez/gmt/1997/0148.html
Other links:
http://wegener.mechanik.tu-darmstadt.de/GMT-Help/Archiv/0143.html
http://c2.com/cgi/wiki?SphericalTrigonometry

Haversine Formula (accurate)

Haversine Formula (from R.W. Sinnott, "Virtues of the Haversine",
Sky and Telescope, vol. 68, no. 2, 1984, p. 159):

a = sin^2(dy/2) + cos(y0) cos(y) sin^2(dx/2)
c = 2 * arcsin(min(1,sqrt(a)))
d = R * c

will give mathematically and computationally exact results. The intermediate result c is the great circle distance in radians.
The great circle distance d will be in the same units as R.

The min() function protects against possible roundoff errors that could sabotage computation of the arcsine if the two points are very nearly antipodal (that is, on opposide sides of the Earth). Under these conditions, the Haversine Formula is ill-conditioned (see the discussion below), but the error, perhaps as large as 2 km (1 mi), is in the context of a distance near 20,000 km (12,000 mi).

Shape of the earth

http://www.synergos-tech.com/topic011.htm states:

Earth mean (avg.) radius = 3959.740 miles
1 degree on a great circle = 69.1105 miles
1 degree on the equator = 69.186 miles
1 geographical mile (1/15 equatorial degree) = 4.613 miles

Law of Cosines for Sperical Trigonometry (ill-conditioned)

An UNRELIABLE way to calculate distance on a spherical Earth is the

Law of Cosines for Spherical Trigonometry
** NOT RECOMMENDED **

a = sin(lat1) * sin(lat2)
b = cos(lat1) * cos(lat2) * cos(lon2 - lon1)
c = arccos(a + b)
d = R * c

Although this formula is mathematically exact, it is unreliable for small distances because the inverse cosine is ill-conditioned...
In Maple, this should not be a problem.

```
> Qr := (x,y) -> arccos(sin(y)*sin(y0) +
cos(y)*cos(y0)*cos(x0-x));
```

$$Qr := (x, y) \to \arccos(\sin(y)\sin(y0) + \cos(y)\cos(y0)\cos(x0 - x))$$

The above formula is exact but numerically unstable. Qr2 is better computationally.

```
> Qr2 := (x,y) -> 2*arcsin(sqrt(sin((y-y0)/2)^2 +
cos(y0)*cos(y)*sin((x0-x)/2)^2));
```

$$Qr2 := (x, y) \to 2\arcsin\left(\sqrt{\sin\left(\frac{1}{2}y - \frac{1}{2}y0\right)^2 + \cos(y0)\cos(y)\sin\left(\frac{1}{2}x0 - \frac{1}{2}x\right)^2}\right)$$

World Maps

All the world map pictures are courtesy of Dr. Ross Taylor (taylor@clarkson.edu) who has done extensive work on mapping the world using Maple. The data used by him comes from

http://www.ngdc.noaa.gov/mgg/shorelines/data/gshhs/gshhs_shp/

Global Self-consistant Hierarchical High-resolution Shorelines

Version 1.2 May 18, 1999

Made programs POSIX.1 compliant and added binary open for DOS.

Version 1.1, April 30, 1996

Paul Wessel, G&G, SOEST, U of Hawaii (wessel@soest.hawaii.edu)
Walter H. F. Smith, NOAA Geosciences Lab (walter@amos.grdl.noaa.gov)

Ref: Wessel, P., and W. H. F. Smith, 1996, A global self-consistent, hierarchical, high-resolution shoreline database, J. Geophys. Res., 101, 8741-8743
>
>