10 GHz and Up Contest Scoring Spreadsheet
with hints on the calculations

Courtney Duncan, N5BF
San Bernardino Microwave Society
Microwave Update, San Diego, October 2015

Motivation

Scoring in the American Radio Relay League (ARRL) 10 GHz and Up Contest http://www.arrl.org/10-ghz-up involves counting QSOs, unique callsigns worked, and great circle distance of each contact. There are many software tools and smartphone apps out there to perform or aid with these calculations and logs but the math and the logic are straightforward, well suited to a spreadsheet, and I get nearly the same thrill from typing a formula into a spreadsheet cell and having it instantly give me a correct result as I do from hearing my call come back from a potential contest QSO partner. So in 2013 I took out a sheet of scratch paper, opened a spreadsheet, and rolled my own. This paper describes the result in its 2015 incarnation. It is not a polished consumer product providing foolproof data entry checks. The intent here, rather, is to utilize and explain the calculations as an aid to anyone wanting to roll their own version too.

The Spreadsheet Approach

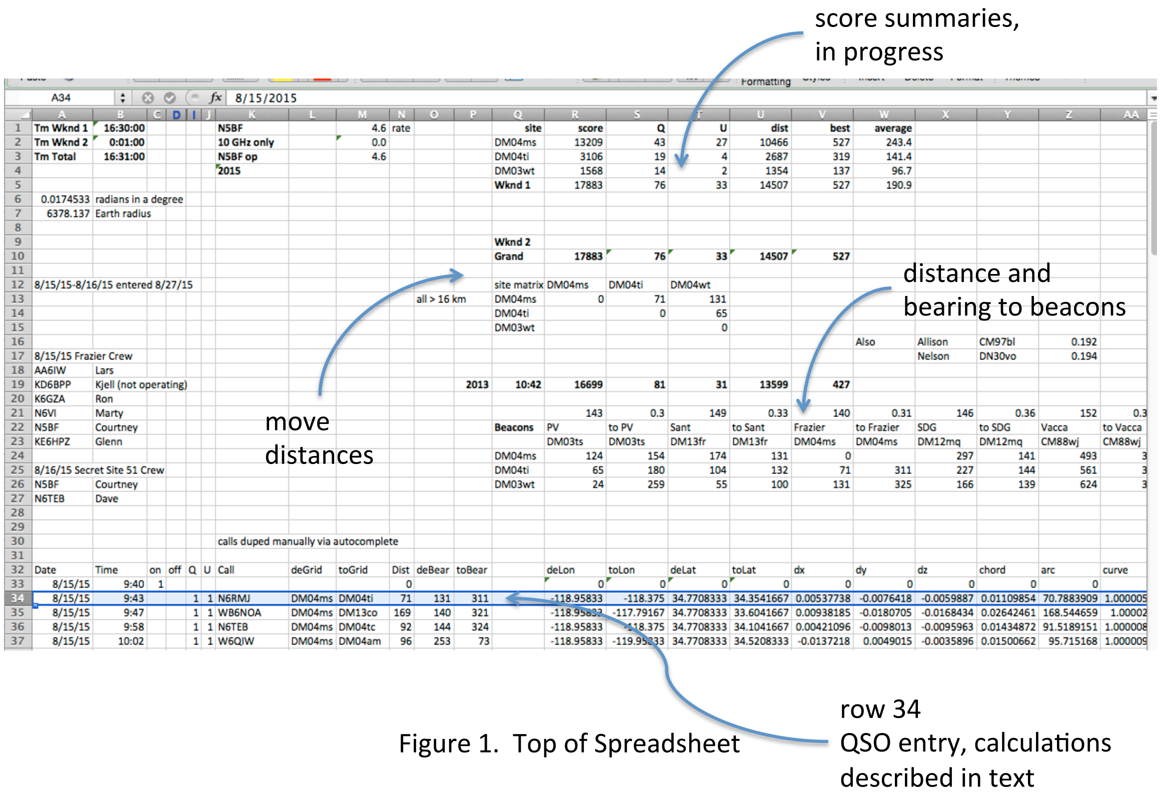
In the sample spreadsheet, the top of which is shown in Figure 1., I have used the lines above row 30 as a general collection area for information that might be operationally useful, like running scores per site roved and beacon headings, time on the air, and rates. It is all ad hoc; I just do calculations down below and link to them up here. The idea is to make something of which that I can print out the first page for field reference or as a contest summary.

The Microsoft Excel spreadsheet itself can be downloaded at

http://www.ham-radio.com/sbms/presentations/N5BF/2015_n5bf_10GHz_log.xlsx

and is filled out with my actual QSOs from the August weekend of the contest as examples. Space is left for the September weekend.

The calculations discussed here underlie the log-entry rows beginning with row 33. Below line 130 is a work area where the “useful information” calculations for the top are actually performed. These include beacon headings and distances and roving move distances. I’ve also just jotted down reference information, like the calls of the other stations at some sites where I spent some time.



A34 | *fx* 8/15/2015

	A	B	C	D	I	J	K	L	M	N	O	P
1	Tm Wknd 1	16:30:00					N5BF		4.6	rate		
2	Tm Wknd 2	0:01:00					10 GHz only		0.0			
3	Tm Total	16:31:00					N5BF op		4.6			
4							2015					
5												
6	0.0174533	radians in a degree										
7	6378.137	Earth radius										
8												
9												
10												
11												
12	8/15/15-8/16/15 entered 8/27/15											
13											all > 16 km	
14												
15												
16												
17	8/15/15 Frazier Crew											
18	AA6IW	Lars										
19	KD6BPP	Kjell (not operating)										2013
20	K6GZA	Ron										
21	N6VI	Marty										
22	N5BF	Courtney										
23	KE6HPZ	Glenn										
24												
25	8/16/15 Secret Site 51 Crew											
26	N5BF	Courtney										
27	N6TEB	Dave										
28												
29												
30							calls duped manually via autocomplete					
31												
32	Date	Time	on	off	Q	U	Call	deGrid	toGrid	Dist	deBear	toBear
33	8/15/15	9:40	1							0		
34	8/15/15	9:43			1	1	N6RMJ	DM04ms	DM04ti	71	131	311
35	8/15/15	9:47			1	1	WB6NOA	DM04ms	DM13co	169	140	321
36	8/15/15	9:58			1	1	N6TEB	DM04ms	DM04tc	92	144	324
37	8/15/15	10:02			1	1	W6QIW	DM04ms	DM04am	96	253	73

	Q	R	S	T	U	V	W	X	Y	Z	AA
1	site	score	Q	U	dist	best	average				
2	DM04ms	13209	43	27	10466	527	243.4				
3	DM04ti	3106	19	4	2687	319	141.4				
4	DM03wt	1568	14	2	1354	137	96.7				
5	Wknd 1	17883	76	33	14507	527	190.9				
9	Wknd 2										
10	Grand	17883	76	33	14507	527					
12	site matrix	DM04ms	DM04ti	DM04wt							
13	DM04ms	0	71	131							
14	DM04ti		0	65							
15	DM03wt			0							
16							Also	Allison	CM97bl	0.192	
17								Nelson	DN30vo	0.194	
19	10:42	16699	81	31	13599	427					
21		143	0.3	149	0.33	140	0.31	146	0.36	152	0.3
22	Beacons	PV	to PV	Sant	to Sant	Frazier	to Frazier	SDG	to SDG	Vacca	to Vacca
23		DM03ts	DM03ts	DM13fr	DM13fr	DM04ms	DM04ms	DM12mq	DM12mq	CM88wj	CM88wj
24	DM04ms	124	154	174	131	0		297	141	493	3
25	DM04ti	65	180	104	132	71	311	227	144	561	3
26	DM03wt	24	259	55	100	131	325	166	139	624	3
32		deLon	toLon	deLat	toLat	dx	dy	dz	chord	arc	curve
33		0	0	0	0	0	0	0	0	0	
34		-118.95833	-118.375	34.7708333	34.3541667	0.00537738	-0.0076418	-0.0059887	0.01109854	70.7883909	1.000005
35		-118.95833	-117.79167	34.7708333	33.6041667	0.00938185	-0.0180705	-0.0168434	0.02642461	168.544659	1.00002
36		-118.95833	-118.375	34.7708333	34.1041667	0.00421096	-0.0098013	-0.0095963	0.01434872	91.5189151	1.000008
37		-118.95833	-119.95833	34.7708333	34.5208333	-0.0137218	0.0049015	-0.0035896	0.01500662	95.715168	1.000009

Figure 1. Top of Spreadsheet

Contest rules limit participation to 24 hours within the period 0600 Saturday to 2400 Sunday local time on each of the two weekends. Columns C, D, E, F, and G are used to calculate time "on the air" per weekend, August being "weekend 1" and September being "weekend 2". The results appear in cells B1, B2, and B3. Row 33 is the beginning of a session where a "1" (for weekend 1) appears in column C indicating that I am QRV for the beginning of the contest (or segment) at 8/15/15 0940 (A33 and B33). This first segment (Frazier Mountain DM04ms) ends at line 77 with a "1" (for weekend 1) in the "off" (or QRT) column D. All the lines between QRV and QRT are QSO log entries where column I (labeled "Q" for "QSO") contains a "1" for a valid QSO and column J (labeled "U" for "unique") contains a "1" for a new unique callsign (an "initial") or a "0" for a callsign worked before, as from a different location. These columns are summed up in the usual spreadsheet ways for the scoring cells at the top of the spreadsheet.

As I enter callsigns, the tooltip in Excel will give me completion options for entries previously made in that column. If I see the callsign I am entering in those options, I know the call is not unique and has already been counted as unique previously and so I enter "0" in the "unique" column. "Search" can also be used to make sure the callsign (assuming it is typed correctly) has or has not appeared before. There is no other auto-checking or guaranteeing that you are not cheating in these fields as there might be in a real software product. These columns are just done in this way to enable easy spreadsheet summing.

Once the rows are set up to make the distance and bearing calculations to be described next, I enjoy entering all the QSO into the spreadsheet from my field note "logs". It is like reliving the contest!

Calculating Latitude and Longitude from Grid Squares

Now for the interesting mathematics: the calculation of QSO distances and bearings from six digit grid squares. Again, there is no validity checking in this code (it is all worked up for the "happy" or "correct" data case). An entry error will result in erroneous (likely entertainingly so) results.

Column L is the "from grid" labeled "deGrid" which is where I'm operating, and is the report I gave in the QSO. Column M is the "to grid" labeled "toGrid" which is the location I receive in the other station's report. For correct calculations hereafter, the first two characters must be upper case letters, the next two must be numbers, and the final two must be lower case letters. The system, as it needs only 24 letters, does not use the letters "y" and "z" but, again, there is no checking here. A typo of "DM04tz" gives an equivalent grid to DM05tb due to wraparound of the calculation. Put another way, "error in, error out."

For viewing convenience, columns N, O, and P, the final results of the calculations to follow to the right: distance, "bearing from me" and "bearing to me" respectively, are placed next. These will be discussed at the end of the paper. Column Q is blank to demark the work area to the right. A programmer interested in appearance might hide all these calculation columns, but I enjoy looking at them and keeping my eye on them for erroneous looking stuff, so I don't.

While this paper is not a tutorial on Microsoft Excel (for which one can use the online help features and documentation), I will explain keywords relevant to the calculations. (Translation to another spreadsheet or even a C program should be straightforward.) For example, in converting from the alphanumeric grid square designators to numerical latitudes and longitudes, it is necessary to get a numerical interpretation of letters represented in ASCII. This is done with the "CODE()" keyword where

CODE("A") = 65
 (decimal, I will not digress here into a discussion of the beauties of hexadecimal)
CODE("B") = 66
CODE("C") = 67
.
.
CODE("a") = 97
CODE("b") = 98
.
.

and so forth.

The design of the Maidenhead Grid Square system is as follows:

The first letter represents longitude beginning from the dateline (180 degrees) and proceeding eastward in 20-degree steps:

A represents -180 to -160,
B represents -160 to -140,

and so forth, where the convention is that west longitude is negative and east longitude is positive.

The second letter represents latitude beginning from the South Pole in 10-degree steps:

A represents -90 to -80,
B represents -80 to -70,

and so forth, where the convention is that south latitude is negative and north latitude is positive.

The third character is a number representing the number of 2-degree steps into the 20-degree segment represented by the first letter. For example "D_1_ "(where "_" means, "don't care" in this example) represents the segment 118 to 116 west longitude.

The fourth character is a number that goes similarly with the second letter in one-degree steps of latitude. For example "_M_4" represents the segment 34 to 35 degrees north latitude.

So, DM14 is bounded by -118 to -116 longitude and +34 to +35 latitude and the DM14 in the calculation so far means the west and south edges of the grid: -118 / +34.

Finally, the fifth and sixth characters divide each 2 x 1 degree segment into 24 increments of longitude (5 minutes of arc each) and latitude (2.5 minutes of arc each) respectively. Only the 24 letters "a" through "x" are used. The number 24 is chosen because it is more commensurate with the 60 minutes of arc that degrees are conventionally divided into than, for example, the 26 letters of the alphabet would be.

Note that, by definition, a minute of arc in latitude is a nautical mile, so 6x6 grids are, worldwide, 2.5 nautical miles (4.63 kilometers) tall and 4x4 grid squares are 60 nautical miles (111.111 kilometers) tall. By contrast, the size of a minute (or degree) of arc in longitude is proportional to the cosine of the latitude. At the equator (cosine(latitude=0) = 1), 6x6 grid squares are 5 nautical miles wide (9.26 kilometers) and have an aspect ratio of 2. At 60 north or south (cosine(latitude = 60 degrees) = .5) grid squares are 2.5 nautical mile (4.63 kilometer) literal squares, and at higher latitudes, they are actually taller than wide! This is one reason why the grids cover twice as much longitude as latitude. Also, the system must cover 360 degrees of longitude worldwide while there are only 180 degrees of latitude to be represented.

This provides the background for understanding the calculation in column R ("deLon") which decodes the "deGrid" (entered by user) into longitude in decimal degrees.

The calculations about to be described locate not the edge but the center of the indicated 6x6 grid square, which is the convention for determining positions, distances, and bearings from grid squares. Clearly, the position found can and will be a few kilometers off from the true position depending on the station's location in the grid. The grid square system can be extended to additional digits for greater precision but that is not done here or for this contest nor is greater precision generally needed except for very close in contacts (a few kilometers) that might occur within the same grid square or adjacent ones. These should be handled separately and by other measurement means.

In the following, I break the single cell calculation out into its constituent parts, explaining each one with comments, following the "//" markers to the right. (This is a standard programming usage.) The calculation for row 34, the first actual logbook entry is:

```
=IF(LEN(L34)=6,   // OK, so there is a little "fat finger" checking.
                  // This makes sure there are 6 characters in deGrid.
                  // If there are…

-180+(CODE(MID(L34,1,1))-CODE("A"))*20
                  // This takes the first letter "D" in this case and
                  // determines which 20-degree segment, starting
                  // at -180 degrees, it is. In this case
                  // "D" - "A" is 3, so the west edge of the grid is
                  // -180 + 3*20 = -120

+VALUE(MID(L34,3,1))*2
                  // This takes the third character and just uses it as
                  // as the number of two-degree steps further east.
                  // In this case it's "0" so we're still at -120 degrees
                  // for the west edge.
+(CODE(MID(L34,5,1))-CODE("a")+0.5)/12
                  // Finally, the fifth character, an "m" in this case is
                  // used to find the distance east of -120.
                  // In this case "m" - "a" is 12, the "+0.5" puts it in
                  // the middle of the grid  (rather than the west edge)
                  // at 12.5 / 12 = 1.0416667 for a total of
                  // -120 + 1.0416667 = -118.958333 degrees.
                  // The "/12" divides two degrees into 24 parts,
                  // "a" to "x".

,0)
                  // The final ",0" refers to the length check at the
                  // very beginning.  If there were not six characters
                  // just put a 0 here to keep the spreadsheet looking
                  // "clean" and indicate that this is not a log entry line.
```

Column S calculates the "toLon" from the "toGrid" in column M similarly.

Column T calculates "deLat" from the "deGrid" in column L similarly to the longitude calculation:

```
=IF(LEN(L34)=6,     // check the grid designator length, if OK…

-90+(CODE(MID(L34,2,1))-CODE("A"))*10
                    // Take the second character and find the number
                    // of 10-degree segments north of the South Pole
                    // (-90) to the south edge of the grid.  In this case,
                    // "M" - "A" = 12 so we're at -90 + 120 = 30 north.

+VALUE(MID(L34,4,1))*1
                    // Plus the number of 1-degree segments north of that
                    // south edge.  In this case it's 4, so the south edge is
                    // 34 north.

+(CODE(MID(L34,6,1))-CODE("a")+0.5)/24
                    // Finally, find the number of 24ths of a degree north
                    // of the south edge (60 minutes / 24 = 2.5 minutes)
                    // plus a half grid +0.5.  In this case, "s" - "a" is 18
                    // so it's 18.5 / 24 = 0.7708333 degrees for a total
                    // of 34.7708333 degrees.

,0)
                    // And finally the ",0" which answers the "fat finger"
                    // or "not a log entry" check at the beginning of the
                    // line.
```

Similarly, column U calculates the "toLat" from the "toGrid" in column M.

Calculating Distance from Latitudes and Longitudes

Now we have all we need to calculate distances and bearings. We could just go to spherical trigonometry at this point (discussed more below) but when I first set this up, I decided to use unit vectors, as follows.

Although the earth is an ellipsoid, being about 21 km smaller in radius at the poles than at the equator (due to the hydrodynamic balancing of centripetal force due to its 24 hour rotation), this fact is negligible for the kilometer-class calculations being performed here. The earth is treated as a perfect sphere of radius one. Every position on the surface can then be represented by a vector from the center to the surface, with length one, whose Cartesian coordinates (X, Y, and Z) are trigonometric functions of the latitude and longitude angles. All calculations and distances are then in radians on this "unit sphere" and are converted, at the end, to kilometers by multiplying by the earth radius.

For these calculations, I use the convention that longitude east of Greenwich is positive and west is negative, preserving a right handed X/Y/Z coordinate system for the earth. (+X is 0 latitude/0 longitude in the Atlantic off Gabon, +Y is at the equator south of India, and +Z is the North Pole.) This matches the right-handed convention of west longitude being negative and north latitude being positive.

Column V calculates dx, the distance from me "de" to the other station "to" in the X direction.

```
=COS(S34*Radians)*COS(U34*Radians)
                        // cos(toLon) * cos(toLat) = the X component of "to"

-COS(R34*Radians)*COS(T34*Radians)
                        // cos(deLon) * cos(deLat) = the X component of "de"
                        // Noting that computers work in radians
                        // Cell A6 is the degrees to radians constant.
```

Similarly column W calculates dy.

```
=SIN(S34*Radians)*COS(U34*Radians)
                        // sin(toLon) * cos( toLat ) = the Y component of "to"

-SIN(R34*Radians)*COS(T34*Radians)
                        // sin(deLon) * cos(deLat) = the Y component of "de"
```

And column X calculates dz.

```
=SIN(U34*Radians)
                        // sin(toLat) = the Z component of “to”

-SIN(T34*Radians)
                        // sin(deLat) = the Z component of “de”
```

We now have the direction from my station to the other station in X, Y, Z coordinates in units of earth radius. The length of this vector, calculated in column Y, is just the Pythagorean:

```
=SQRT(V34*V34+W34*W34+X34*X34)
                        // That is, sqrt( dx^2 + dy^2 + dz^2 )
```

But this is the “chord” or straight line vector distance between the two stations that passes through the inside of the earth, not the great circle distance along the surface on which scoring and “as the crow flies” distances are based. Column Z calculates this surface distance in radians and multiplies it by the radius of the earth to finally give kilometers.

```
=2*ASIN(Y34/2)*Rearth
                        // That is, 2 * arcSin( half of the chord )
                        // (This works because the radius=hypotenuse = 1
                        // then times the radius of the earth.
```

This is described pictorially in Figure 2.

Earth radius, cell A7, is 6378.137 kilometers, from the WGS-84 coordinate system used by the Global Positioning System (GPS). This could easily be changed another earth radius if needed, or to Mars or another planet in this one cell, though the need for this (that is, microwave contesting on Mars) is, sadly, not near term.

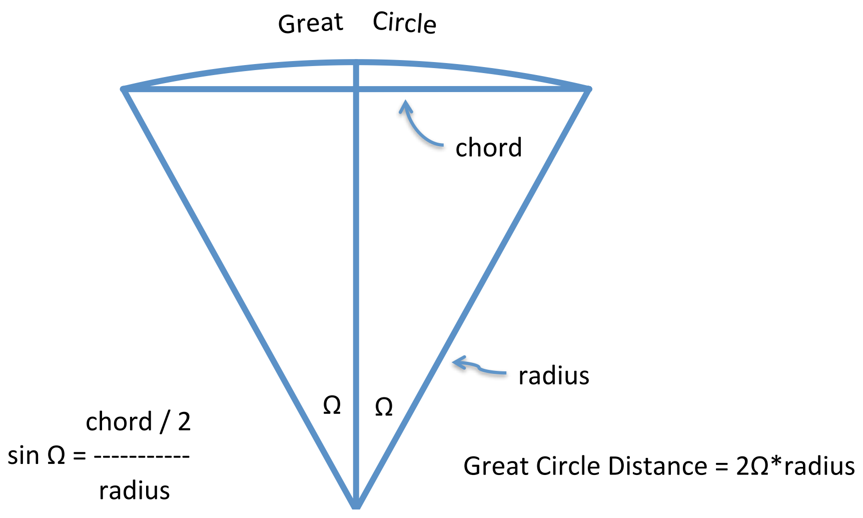


Figure 2. Chord to Great Circle

Calculate Bearings from "de" to "to" and from "to" to "de"

Now we calculate heading from "de" to "to" (my heading) and from "to" to "de" (the other station's reciprocal heading) using the well known Law of Sines for spherical trigonometry:

$$\frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c}$$

That is, the ratio of sines of corners of the spherical triangle with the sines of the opposite sides are equal for all three corners and sides.

We use the North Pole for corner C (Figure 3) where the angle is just the difference of longitudes. We have just calculated opposite side c, the arc distance between stations so we have the ratio 'sin C / sin c' which allows us to solve for the unknowns 'sin A' and 'sin B' which will give us the angles with the lines of longitude and allow us to ultimately determine bearings.

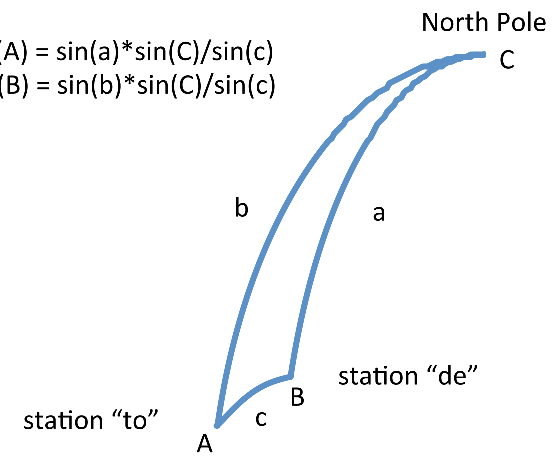
(Side c, the arc distance between stations, could also have been calculated with spherical trigonometry using the other well known "cosine rule" cos c = cos a*cos b + sin a*sin b * cos C

but I had already done it the other way by the time I realized this, both methods get the same answer, of course, and I didn't deem it worth going to the trouble to change. See

https://en.wikipedia.org/wiki/Spherical_trigonometry

and numerous other references.)

sin(A) = sin(a)*sin(C)/sin(c)
Sin(B) = sin(b)*sin(C)/sin(c)



$a = 90 - \text{latitude}_{de}$
$b = 90 - \text{latitude}_{to}$
c = Great Circle between stations
A = bearing at "to"*
B = bearing at "de"*
$C = \text{longitude}_{to} - \text{longitude}_{de}$

*Quadrant of sine unresolved
See Figure 4.

Figure 3. Great Circle Bearings

So, column AB finds the intermediate result

```
=SIN((S34–R34)*Radians)/SIN(Z34/Rearth)
                        // That is sin(toLon – deLon) / sin( arc distance )
                        // which is sin(C) / sin(c)
                        // with 'arc distance' converted back to radians.
```

Column AC finds the angle from "de" to "to" with the longitude (north-south) line.

```
=(ASIN(COS(U34*Radians)*AB34)/Radians)
                    // That is sin(B) = sin(b) * sin(C)/sin(c)
                    // converted to degrees.
                    // (COS() is used since sin(b) = cos(90-b)
                    // and 90-b is the latitude value we already have.)
```

And column AD finds the angle from “to” to “de” with the longitude (north-south) line

```
=(ASIN(COS(T34*Radians)*-AB34)/Radians)
                    // That is sin(A) = sin(a) * sin(C)/sin(c)
                    // converted to degrees.
                    // (COS() is used since sin(a) = cos(90-a)
                    // and 90-a is the latitude value we already have.)
```

In all trigonometric arc functions there is an ambiguity of direction depending on what quadrant the actual angle is in. In this case, if the “to” location is further north than the “de” location, the angle B we have just calculated is with the meridian going north (Figure 4). If this number is greater than zero, it is the true bearing. If it is less than zero, we just add 360 degrees (one circle) to make it into an equivalent positive number

If, on the other hand, the “to” station is further south than the “de” station, the angle B is with the meridian going south and the true bearing is 180 degrees minus that angle. This is handled for the “deBearing” in column AE

```
=IF(X34<0,
                    // if dz < 0, “to” is south of “de”

180-AC34,
                    // bearing = 180 - B
                    // if B < 0, this just makes bearing > 180
                    // which is fine (e.g. 225 degrees)

                    // else dz > 0, “to” is north of “de”
IF(AC34<0,360+AC34,
                    // if B < 0, report 360 + B (e.g. 315 degrees)
AC34))
                    // if B >= 0, the angle with the north going meridian
                    // is the bearing, just report B.
```

Similarly, the bearing from “to” do “de” is resolved in “toBearing” in column AF

```
=IF(X34>0,
                        // if dz > 0, "de" is south of "to"

180-AD34,
                        // bearing = 180 - A
                        // if A < 0, this just makes bearing > 180
                        // which is fine (e.g. 225 degrees)

                        // else dz < 0, "de" is north of "to"
IF(AD34<0,360+AD34,
                        // if A < 0, report 360 + A (e.g. 315 degrees)
AD34))
                        // if A >= 0, the angle with the north going meridian
                        // is the bearing, just report A.
```

For stations due north and south, or for those close enough to neglect the curvature of the earth and consider it as a plane surface, the difference between these headings should be 180 degrees. For regional or adjacent state distances the difference should be 180 degrees within a degree or two. This is checked in column AG (just as a programmers sanity check).

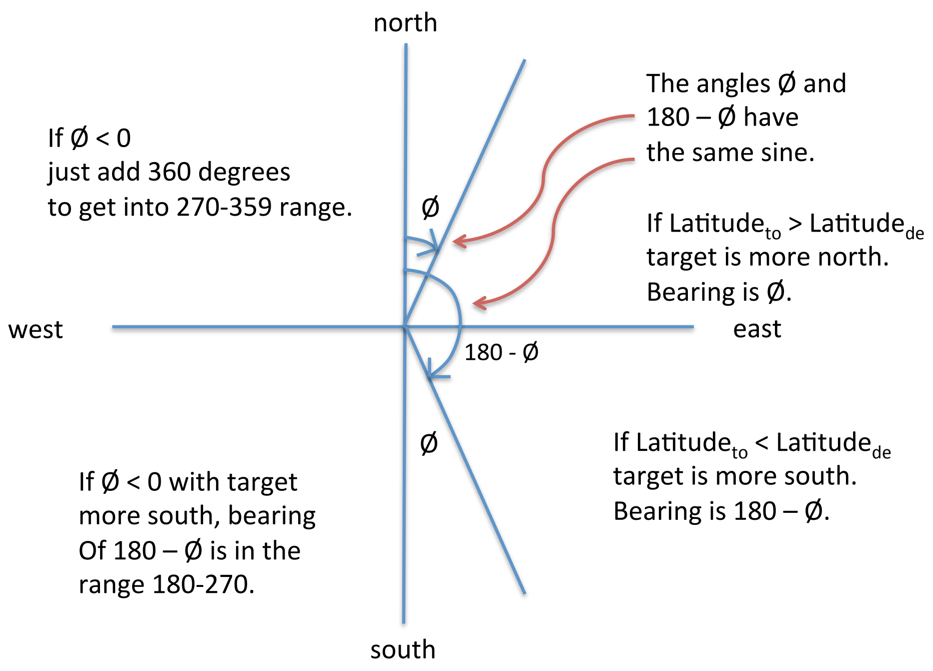


Figure 4. Arcsine Resolution

Rounding and Reporting

This completes the calculations and the results are now rounded into integer kilometers and degrees and placed back in columns N, O, and P for distance, bearing from me, and bearing from other station to me, respectively. The rounding is done like this

```
=INT(Z34+0.5)
                    // Add half a kilometer and truncate to the next
                    // integer down.  This makes
                    // 234.4 km -> INT(234.9) = 234
                    // 234.5 km -> INT(235.0) = 235
                    // 234.6 km -> INT(235.1) = 235
                    // which is the appropriate convention
                    // for rounding and scoring.
```

General Use

This row can be copied as many times as desired for log entries or for helper calculations. Simply copy the row and enter the desired “de” and “to” grid designators in columns L and M.

While filling out the August 2015 portion of my log into this spreadsheet, I checked all of these distance and bearing calculations with online grid square calculators and all results agreed within the integer round-off used, including those that were “close”, i.e. 249.49 degrees rounding to 249.

This gives an integer number of kilometers, which is used for scoring, and an integer number of degrees for headings, which is about as good as you can do in a typical roving setup and plenty to get within “peak up” range. Summing the integer values in column N for total distance points avoids summing errors that might occur if fractional kilometers were carried in that column.

The kilometer distances are used in the score summaries up in the convenience area at the top. Also, I’ve calculated beacon bearings and distances and roving move distances to check against the 16 km move rule. I’ve used other cells as a scratch pad for useful operational information, either as a computer tool in the field or printout of the top part of the spreadsheet.

Summary

This paper has presented the calculations and example spreadsheet for calculating points, scores, and times for the ARRL 10 GHz and Up Contest held on one weekend in August and one weekend in September each year. The calculations performed on first log entry in the spreadsheet, row 34, are described in detail. This row is then replicated throughout the spreadsheet for other QSO log entries or for helpful results like the distance and bearing to a beacon, or the distance of a rover move.

Partial columns can then be summed for scoring and otherwise analyzed for timekeeping purposes.

The spreadsheet is designed to be useful for this particular event but not foolproof. There is little checking for typos or erroneous data entry. The operator is expected to know that a QSO distance of a million kilometers or a callsign with three numbers in it is an error by inspection and look for entry errors manually, not by user interface robustness.