Journal of Navigation (2008), 61: 355-365 Cambridge University Press
doi:10.1017/S0373463307004602
Published online by Cambridge University Press 25Mar2008

Journal of Navigation (2008), 61:355-365 Cambridge University Press

doi:10.1017/S0373463307004602

Research Article

Vector Solution for the Intersection of Two Circles of Equal Altitude

Andrés Ruiz González[a1] [c1]

[a1] (Navigational Algorithms. San Sebastián.)

Abstract

A direct method for obtaining the two possible positions derived from two sights using vector analysis instead of spherical trigonometry is presented. The geometry of the circle of equal altitude and of the two body fixes is analyzed, and the vector equation for simultaneous sights is constructed. The running fix problem is also treated.

Key Words: Circle of Equal Altitude; Celestial Navigation; Sight Reduction; Vector Analysis

Correspondence:

[c1] (Email: aruiz4@euskalnet.net)

back to top

Vector Solution for the Intersection of two Circles of Equal Altitude

Andrés Ruiz González
http://www.geocities.com/andresruizgonzalez
Navigational Algorithms. San Sebastián.

A direct method for obtaining the two possible positions derived from two sights using the vector analysis instead the spherical trigonometry is presented. The geometry of the circle of equal altitude and of the two body fixes is analyzed, and then the vector equation for simultaneous sights is constructed. Also the running fix problem is treated.

Finally the C++ source code for the algorithm is provided in an easy implementation, susceptible for being translated to other common programming languages.

KEY WORDS

1. Circle of Equal Altitude. 2. Celestial Navigation. 3. Sight Reduction. 4. Vector Analysis.

Paper published in The Journal of Navigation,
The Royal Institute of Navigation

- Submitted: 2007/05/14
- Accepted: 2007/10/09
- Published: Volume 61 - Issue 02, April 2008.

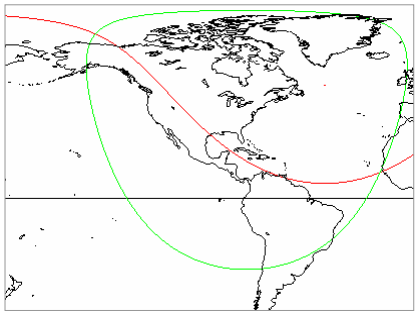
A3. EXAMPLES

Two simultaneous sights

Date	UT1	Body	GHA	Dec	Ho
10/10/1990	19:32:36	Eltanin	43.195708	51.49344	45.50248
10/10/1990	19:33:03	Alphecca	78.832391	26.74654	31.17998

results:

GP1 (xyz)	0.453890851	-0.4261679	0.78253691	
GP2 (xyz)	0.172957212	-0.87609689	0.450044483	
Alpha	36.48192299			
k1	0.840152453			
k2	-0.15779189			
OIc (xyz)	0.354046266	-0.21980502	0.586436933	
L1	0.694570888			
L2	-0.69457089			
GP1xGP2_unit	0.830489865	-0.11592541	-0.54483748	
I1 (xyz)	0.930880349	-0.30032343	0.208008679	
I2 (xyz)	-0.22278782	-0.1392866	0.964865188	
I1 (BL)	12.00568121	-17.8808959	12º 0.3' N	17º 52.9' W
I2 (BL)	74.76697018	-147.98644	74º 46.01'N	147º 59.18'W



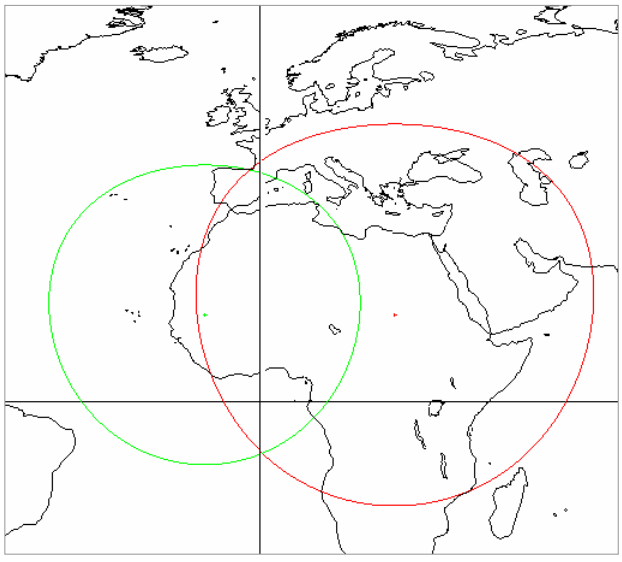
Running fix

Date	UT1	Body	GHA	Dec	Ho
05/05/2007	10:15:28	Sun LL	334.687032	16.205374	54.314509
05/05/2007	12:38:25	Sun LL	10.426691	16.233739	61.982844
C = 20º	S = 10 kt				

results:

iter	error	Be	Le	B1	L1	B2	L2	GHA1 (t2)	dec1 (t2)
0	9.117738163	43.3166	-2	-9.466394	0.845506	43.361622	-2.199485	334.450283	16.53628
1	0.049268529	43.361622	-2.199485	-9.46644	0.8454	43.361382	-2.198405	334.449808	16.53567
2	0.000266894	43.361382	-2.198405	-9.466439	0.845401	43.361383	-2.19841	334.449811	16.53568
3	1.44577E-06	43.361383	-2.19841	-9.466439	0.845401	43.361383	-2.19841	334.449811	16.53568
4	7.82877E-09	**43.361383**	**-2.19841**	-9.466439	0.845401	**43.361383**	**-2.19841**	334.449811	16.53568

05/05/2007 12:38:25 Fix: 43º 21.7'N 002º 11.9'W



A4. C++ Source code

```
#include <math.h>

#define PI ((double)3.14159265358979)
#define DegRad(ang) ((double)((ang)*PI/180.0))
#define RadDeg(ang) ((double)((ang)*180.0/PI))
#define SIN(x) (sin(DegRad(x)))
#define COS(x) (cos(DegRad(x)))
#define ACOS(x) (RadDeg(acos(x)))
#define ATAN2(x,y) (RadDeg(atan2(x,y)))
#define SQ(x) ((double)((x)*(x)))
```

// General vector algebra functions

```
double Mod( double *x )
{
	return( sqrt(x[0]*x[0]+x[1]*x[1]+x[2]*x[2]) );
}


double* Add( double *x, double *y )
{
	double *z = new double[3];

	z[0] = x[0]+y[0];
	z[1] = x[1]+y[1];
	z[2] = x[2]+y[2];

	return( z );
}


double* aVector( double a, double *x )
{
	double *z = new double[3];

	z[0] = a*x[0];
	z[1] = a*x[1];
	z[2] = a*x[2];

	return( z );
}


double* Unit( double *x ) { return( aVector( 1.0/Mod(x), x ) ); }


double Dot( double *x, double *y )
{
	return( x[0]*y[0]+x[1]*y[1]+x[2]*y[2] );
}


double* Cross( double *x, double *y )
{
	double *z = new double[3];

	z[0] = x[1]*y[2]-x[2]*y[1];
	z[1] = x[2]*y[0]-x[0]*y[2];
	z[2] = x[0]*y[1]-x[1]*y[0];

	return( z );
}
```

```
// Coordinate transformation functions

double* VectorEquatorial2Cartesian( double Dec, double GHA )
{
	double *v = new double[3];
	// unit vector
	v[0] =  COS( Dec )*COS( GHA );
	v[1] = -COS( Dec )*SIN( GHA );
	v[2] =  SIN( Dec );

	return( v );
}


void Cartesian2Geographical( double x, double y, double z, double* B, double* L )
{
	*B = ATAN2( z, sqrt(x*x+y*y) );
	*L = ATAN2( y, x );
}

// Vector Solution

void Fix2CoP(  double GHA1, double dec1, double HO1,
               double GHA2, double dec2, double HO2,
               double* B1, double* L1, double* B2, double* L2 )
{
	double *GP1, *GP2;
	double *OI1, *OI2;
	double alpha, k1, k2, l1, l2;
	double *GP1l, *GP2l, *OIc;
	double *IcI1, *IcI2, *GP1xGP2_unit;


	GP1 = VectorEquatorial2Cartesian( dec1, GHA1 );
	GP2 = VectorEquatorial2Cartesian( dec2, GHA2 );

	alpha = ACOS( Dot( GP1, GP2 ) );

	k1 = ( SIN(HO1) - SIN(HO2)*COS(alpha) )/SQ(SIN(alpha));
	k2 = ( SIN(HO2) - SIN(HO1)*COS(alpha) )/SQ(SIN(alpha));
	GP1l = aVector( k1, GP1 );
	GP2l = aVector( k2, GP2 );

	OIc = Add( GP1l, GP2l );
	l1 = +sqrt( 1.0-Dot(OIc,OIc) );
	l2 = -l1;
	GP1xGP2_unit = Unit( Cross( GP1, GP2 ) );
	IcI1 = aVector( l1, GP1xGP2_unit );
	IcI2 = aVector( l2, GP1xGP2_unit );

	OI1 = Add( IcI1, OIc );
	Cartesian2Geographical( OI1[0], OI1[1], OI1[2], B1, L1 );
	OI2 = Add( IcI2, OIc );
	Cartesian2Geographical( OI2[0], OI2[1], OI2[2], B2, L2 );


	delete[] GP1;		delete[] GP2;
	delete[] GP1l;		delete[] GP2l;
	delete[] OIc;		delete[] GP1xGP2_unit;
	delete[] IcI1;		delete[] IcI2;
	delete[] OI1;		delete[] OI2;
  }
```