Elliptic Integrals: the Landen Transformation and Carlson Duplication

**Acknowledgements**: I would like to thank Rod Deakin for suggesting this project and for his 2014 paper which formed the material for this project. Also I would like to thank Don Grant for direction and helpful comments in doing this project.

**Abstract**: **The aim of this project was to show that the recursive processes of the Landen transformation and Carlson forms can give arbitrary precision in problems involving elliptic integrals. Programs were written in MATLAB© to find Elliptic Integrals of the First and Second Kind using a Landen transformation and Elliptic Integrals of the Third Kind using Carlson Duplication. These were used to calculate meridian distances and to solve the direct problem of geodesy. These techniques give mathematical precision up to machine precision and were checked with MATLAB©’s ellipticF, ellipticE and ellipticPi inbuilt functions. These MATLAB© functions were shown to give 15 decimal place precision and as such can be used with confidence to solve geodetic problems. It was found that Carlson Duplication gives a straightforward technique to solve elliptic integrals.**

Contents

[2. Introduction 6](#_Toc402209323)

[3. What is a Landen transformation? 9](#_Toc402209324)

[3.1 History of the Landen transformation and its use in geodesy. 10](#_Toc402209325)

[4. MATLAB© First Kind program results 11](#_Toc402209326)

[5. Elliptic Integrals of the Second Kind 15](#_Toc402209327)

[6. MATLAB© Second Kind program results 16](#_Toc402209328)

[7. Meridian Arc Length Formula (Derivation given in Appendix L) 17](#_Toc402209329)

[8. Results of Calculations of Arc Lengths along Meridians 17](#_Toc402209330)

[9. Ascending and Descending First Kind transformation 18](#_Toc402209331)

[10. Landen’s descending transformation for Second Kind integrals 20](#_Toc402209332)

[11. Ascending and Descending Second Kind transformation 20](#_Toc402209333)

[12. Inverting Elliptic Integrals of the First Kind 22](#_Toc402209334)

[13. MATLAB© First Kind Inversion program results 23](#_Toc402209335)

[14. Evaluating General Elliptic Integrals of the Third Kind using Carlson Symmetric Forms 24](#_Toc402209336)

[14.1 What are functions (brief introduction of terms)? 25](#_Toc402209337)

[14.2 Elliptic Integral of the Third Kind with parameters useful in geodesy 26](#_Toc402209338)

[15. Inverting Elliptic Integrals of the Third Kind using Carlson form and a binary technique 28](#_Toc402209339)

[16. The Direct Geodetic Problem 29](#_Toc402209340)

[17. Geodesics (Rollin’s Method) 30](#_Toc402209341)

[18. My Direct.m results compared with Karney’s GEODRECKON 34](#_Toc402209342)

[19. Safeguards against the computer not preserving precision when implementing recursion 37](#_Toc402209343)

[20. Conclusions: 37](#_Toc402209344)

[21. Appendices 41](#_Toc402209345)

**Appendices**

[Appendix A MATLAB© code for determining elliptic integrals of the First Kind using Landen’s ascending transformation 41](#_Toc402209369)

[Appendix B MATLAB© code for determining elliptic integral of the Second Kind 42](#_Toc402209370)

[Appendix B.1 Theory summary 42](#_Toc402209371)

[Appendix B.2 Code 42](#_Toc402209372)

[Appendix B.3 Results 43](#_Toc402209373)

[Appendix C MATLAB© code for determining meridian distance M. 44](#_Toc402209374)

[Appendix D MATLAB© code First Kind integrals using Landen’s descending transformation 45](#_Toc402209375)

[Appendix E MATLAB© code for finding the amplitude of a First Kind elliptic integral given the result of the integral (inverting the First Kind integral). 46](#_Toc402209376)

[Appendix F MATLAB© code for determining elliptic integral of the Third Kind using Carlson 47](#_Toc402209377)

[Appendix G MATLAB© code for determining the inverse of an elliptic integral of the Third Kind using Carlson and a binary search technique. 48](#_Toc402209378)

[Appendix H MATLAB© code for solving the direct problem 51](#_Toc402209379)

[Appendix I MATLAB© code for testing the direct problem 55](#_Toc402209380)

[Appendix J Results of running the General Elliptic Integrals of the Third Kind program ThirdCarlson and the inbuilt MATLAB© program ellipticPI. 56](#_Toc402209381)

[Appendix K Arc Length of a Curve 59](#_Toc402209382)

[Appendix K.1 Arc length given a polar equation 60](#_Toc402209383)

[Appendix K.2 Arc Length of an Ellipse 60](#_Toc402209384)

[Appendix K.3 Parametric equations of the ellipse in terms of the latitude parameter 61](#_Toc402209385)

[Appendix K.4 Arc length as a function of Latitude 64](#_Toc402209386)

[Appendix K.5 Arc length as a function of Latitude using radius of curvature 65](#_Toc402209387)

[Appendix K.6 General Equation for the Radius of Curvature 66](#_Toc402209388)

[Appendix K.7 The Radius of Curvature of the Ellipse 68](#_Toc402209389)

[Appendix L Meridian Distance 70](#_Toc402209390)

[Appendix M Evaluating Elliptic Integrals of the First Kind. 74](#_Toc402209391)

[Appendix M.1 The modulus q is increasing 79](#_Toc402209392)

[Appendix M.2 Determining the new amplitude 79](#_Toc402209393)

[Appendix M.3 The chain of transformed elliptic integrals 80](#_Toc402209394)

[Appendix N Landen’s descending transformation for First Kind integrals 81](#_Toc402209395)

[Appendix O Evaluating Elliptic Integrals of the Second Kind 86](#_Toc402209396)

[Appendix P Abandoned working to find the amplitude given the result of the integral: Second Kind. 93](#_Toc402209397)

[Appendix P.1 Finding the latitude given meridian distance using the elliptic integral of the second kind. 93](#_Toc402209398)

[Appendix P.2 Inverting Elliptic Integrals of the Second Kind 93](#_Toc402209399)

[Appendix Q Abandoned: Inverting Elliptic Integrals of the Third Kind using Carlson form and Newton-Raphson 97](#_Toc402209400)

[Appendix R Abandoned: Inverting Elliptic Integrals of the Third Kind using Newton-Raphson 105](#_Toc402209401)

[Appendix S Abandoned working to evaluate Elliptic Integrals of the Third Kind using a Landen transformation 108](#_Toc402209402)

[Appendix T Khan’s endpoint formula 112](#_Toc402209403)

[Appendix U MATLAB code to test Khan’s formula 114](#_Toc402209404)

[Appendix V MATLAB code to change DD.MMSS into decimal degrees 116](#_Toc402209405)

**Figures**

[Figure 1 Successive amplitudes tolerance 1E-3 13](#_Toc402098621)

[Figure 2 Successive amplitudes beginning with 45° 14](#_Toc402098622)

[Figure 3 Successive amplitudes beginning with 90° 14](#_Toc402098623)

[Figure 4 Successive amplitudes beginning with 135° 15](#_Toc402098624)

[Figure 5 The latitudes which result from equally spaced distances on the ellipse 29](#_Toc402098625)

[Figure 6 30](#_Toc402098626)

[Figure 7 1000 successive 100km geodesics from a point near Chile 36](#_Toc402098627)

[Figure 8 1000 successive 100km geodesics from a point near Chile 37](#_Toc402098628)

# Introduction

This thesis is concerned with calculating the First Kind, Second Kind and Third Kind of elliptic integral using a recursive technique. The formula for distance along a meridian is an elliptic integral of the Third Kind which can be changed into a Second Kind integral plus a constant. The Second Kind integral can be calculated using a Landen transformation.

The Third Kind integral will be calculated using the Carlson Symmetric Forms. Carlson provides a fast method of calculation of general Third Kind elliptic integrals – that is those with a characteristic and modulus which are unrelated. The Carlson forms have as special cases the Landen transformation and Gauss arithmetic-geometric mean transformation.

The particular Carlson method of solution of the general Third Kind integral used here is the duplication method. Carlson in his paper (Carlson B. C., 1979) adds an extra step of a Taylor’s series after convergence (p2) but the program presented here omits this step as speed has not been made a priority.

The calculation of meridian distances on the ellipsoid is a purely mathematical problem and distances between two points will depend on the ellipsoid chosen. The GRS80 ellipsoid is the one currently used in Australia and it has a semi-major axis of 6,378,137.0 metres and an inverse flattening of 298.257222101 (Geocentric Datum of Australia, 2014).

As the semi-major axis and the flattening are given on the Geosciences Australia website the eccentricity will be calculated from these two given facts using the following formula:

This gives a value of as 0.08181919104281579 to sixteen significant digits.

When using MATLAB© the function ‘eps’ gives the gap between one number and the next higher number which can be stored. Running the ‘eps’ function on the value of above gives the following result:

>> eps(0.08181919104281579)

ans =

1.387778780781446e-17

This means the next higher number after 0.08181919104281579 is 0.08181919104281580. So when computations are performed using MATLAB© a value of eccentricity of 0.08181919104281579 will be used. Any more digits will not be able to be stored.

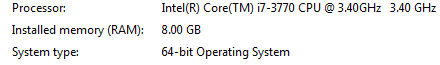
Similarly the value of that will be used is: 3.1415926535897932 as running the ‘eps’ function on this value of gives :

>> eps(3.1415926535897932)

ans =

4.440892098500626e-16

So the storage precision of is in the sixteenth decimal place with the computer used for this work. The computer details are:



In summary then the following values for the GRS-80 ellipsoid are used in this work:

All of the ellipsoids used to model the earth are ellipsoids of revolution so all meridian sections will produce the same ellipse. This ellipse with semi-major axis and semi-minor axis becomes the mathematical object of study.

To determine arc lengths on this ellipse a Third Kind integral needs to be calculated. This Third Kind integral can be written as a Second Kind integral plus a constant. The Second Kind integral can be calculated using a recursive technique that transforms the Second Kind integral into the sum of a Second Kind and a First Kind integral. An elliptic integral of the First Kind in Legendre notation is:

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

An elliptic integral of the Second Kind in Legendre notation is:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

An elliptic integral of the Third Kind in Legendre notation is:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

A special case of an elliptic integral of the Third Kind which occurs in determining the meridian arc length is:

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

The is the modulus and in geodesy problems is usually the eccentricity of the ellipse. The is the amplitude and is usually the latitude. If the limits of the integral are between and then the elliptic integral is called a complete integral. Otherwise it is called an incomplete integral.

The Carlson forms of these integrals that will be used here are the (Fukushima, 2012, p. 3)and (Carlson D. Z., 1969, p. 202) forms.

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

From these two the general elliptic integral of the Third Kind can be written as:

(7)

The usual method of solution is to use a binomial expansion to turn the integrand into an infinite series. The required precision is then achieved by integrating a sufficient number of terms. The downside is that the precision needs to be known, then the required number of terms are determined, then the integrals of these terms are determined which are programmed in and the summation achieved by computer which will take in as input the latitude. In unusual situations which call for an unusual level of precision there is some programming required unless the precision has been offered as part of the program used.

Another method of solution of the elliptic integrals is by Landen transformation when using the Legendre forms and by duplication when using the Carlson forms. These forms naturally lend themselves to recursion and so programming using recursion is straightforward. The advantage of this method is that the program to calculate the integrals remains unchanged for different precision and just a precision parameter need be entered into the program.

In addition inversion programs to find latitudes which use numerical methods can only give precision which are as good as the programs to evaluate the original function. In the programs which are developed here a simple binary search technique is used but this will also give arbitrary precision as the forward program to go from latitude to integral result is giving arbitrary precision.

# What is a Landen transformation?

A Landen transformation turns an integral into a constant times another integral. For example the First Kind integral (Eqn (1)) is turned into another First Kind integral as follows:

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Note in the above example that there is a multiplying factor in front, (, the has been changed to a and the has been changed to a It may appear that this change has made things a little bit more complicated with the appearance of the multiplying factor but if the process is continued then eventually the modulus () turns into a one.

The part to be integrated is then:

which is easily integrated as:

The formula to evaluate an elliptic integral of the First Kind is then (see Appendix M for derivation)

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

with

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

and

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

It must be known what the final amplitude is so that it can be used to specify the limits of the integration. Also the sequence of multiplying factors must be kept track of so that once the integration has been evaluated then this can be multiplied by the multiplying factors.

The circumflex on the angle ( indicates that this is the final angle at the point when the modulus has reached its limit.

## History of the Landen transformation and its use in geodesy.

The Landen transformation was discovered by John Landen in the latter part of the eighteenth century and used by Legendre (Deakin, Elliptic Integrals and Landen's Transformation: What I should've known for Geodesy, 2014, p. 1). L.V. King published a book in 1924 which gives a detailed account of this transformation and its application to calculate elliptic integrals.

The textbook *Geodesy the Concepts* by Edward Krakiwsky published in 1986 says that elliptical integrals are usually carried out with power series (p34). Landen’s transformation is not mentioned in the index.

The power series method is one in which the integrand is converted into a power series using a binomial expansion. Each term of the expansion is then integrated. The algorithm can be defined recursively as given in Klotz (1993) (Dorrer p92).

Similarly the textbook *Maths for Map Makers* by Arthur Allan and published in 1997 says that the integrand of the elliptical integral can be converted into a series and integrated term by term (p238). Another method it gives is to use a numerical method (p238). Landen’s transformation is not mentioned in the index.

Gerstl in 1984 calculates complex elliptic integrals by means of a Landen transformation in his paper. The book *Map Projections: Cartographic Information Systems* by Erik W. Grafarend and Friedrich W. Krumm uses the Landen transformation.

Dorrer in 1999 published a paper called *From Elliptic Arc Length to Gauss-Krueger Coordinates by Analytical Continuation*. Landen transformations are used to calculate elliptic integrals and in particular uses complex numbers and the programming language called APL2.

Rösch in 2011 published a paper called *The Derivation of Algorithms to Compute Elliptic Integrals of the First and Second Kind by Landen Transformation*. The computer programming language used is C. Rösch compares his results with those given by Maple V.

Deakin (2014) explicates the work of King and uses *Maxima* as the computer program. Results are compared with those achieved by Dorrer.

My report covers similar ground as Dorrer, Rösch and Deakin but uses MATLAB© as the computer programming language. In addition I have used the Carlson Duplication method to calculate the elliptic integral of the Third Kind.

# MATLAB© First Kind program results

The program written in MATLAB© to compute elliptic integrals of the First Kind is given in Appendix M. The program is called F\_LARec to indicate: First Kind, Landen Ascending and Recursive. The program takes two inputs – the latitude and the eccentricity given by Eqn (9). The function returns as an answer the value of the elliptic integral. This program also prints out the successive moduli and amplitudes as the recursion goes down to the terminating condition (k close enough to one) and then bounces back. So in the following list the numbers are a mirror of each other around the column headings. This first run is for an input of the amplitude of 90° gives:

>> F\_LARec(3.1415926535897932/2,0.08181919104281579)

- 0.0818191910428158 1.5707963267948966

- 0.5288139520469275 0.8263535410338578

- 0.9513204018612136 0.6129086911058872

- 0.9996887750618011 0.5959896142847262

- 0.9999999878886106 0.5958840752748563

- 1.0000000000000000 0.5958840711684208

==============================================================

Integral k Amplitude

==============================================================

0.6346425378629763 1.0000000000000000 0.5958840711684208

0.6347413115257022 0.9999999878886106 0.5958840752748563

0.6505762056505652 0.9996887750618011 0.5959896142847262

0.8510861701379811 0.9513204018612136 0.6129086911058872

1.5734351492093230 0.5288139520469275 0.8263535410338578

This is the how to call the inbuilt MATLAB© mupad function for the same amplitude specifying 40 digit precision in calculations:

DIGITS := 40:

ellipticF(/2, ^2).

The eccentricity is squared because this is what the MATLAB© function accepts.

The first argument of the ellipticF function is . In the following comparison rather than putting in pi()/2 as a first argument a constant was put in so the two functions – mine and MATLAB©’s – could be compared. That is, both mine and MATLAB©’s functions were fed exactly the same numbers as inputs. This methodology was followed throughout all the comparison tests which follow.

|  |  |  |
| --- | --- | --- |
| Latitude, eccentricity | MATLAB© ellipticF result | F\_LARec result  ( |
|  |  |  |
| pi()/2,0.0818191910435 | 1.573  435  149  209  323  129 | 1.573  435  149  209  323 |
| pi()/4,0.0818191910435 | 0.785  876  554  121  529  944 | 0.785  876  554  121  530 |
| 3\*pi()/4,0.0818191910435 | 2.360  993  744  297  116  313 | 2.360  993  744  297  117 |

Table 1

How does the set tolerance for amplitude difference affect the precision of the results? As can be seen from the following table reducing the tolerance to still gives precision such that the fourteenth decimal digit is the same as the MATLAB© result.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| F\_LARec result  ()  ( |  |  |  |  |  |  |
| 1.573  435  149  209  323 | 1.573  435  149  209  323 | 1.573  435  149  209  323 | 1.573  435  149  209  323 | 1.573  435  149  209  323 | 1.573  435  149  209  323 | 1.573  435  149  209  323 |
|  |  |  |  |  |  |  |
|  | 1.573  435  149  209  323 | 1.573  435  149  209  323 | 1.573  435  149  209  323 | 1.573  435  149  209  323 | 1.573  435  149  209  323 | 1.573  435  151  981  971 |

Table 2

When the tolerance between the successive amplitudes is set to be less than then the result for the integral changes as shown in table 2. The eighth decimal place changes. The plot for successive amplitudes for a tolerance of is shown:



Figure 1 Successive amplitudes tolerance 1E-3

The output trace is:

>> F\_LARec(3.1415926535897932/2,0.08181919104281579)

- 0.0818191910428158 1.5707963267948966

- 0.5288139520469275 0.8263535410338578

- 0.9513204018612136 0.6129086911058872

- 0.9996887750618011 0.5959896142847262

==============================================================

Integral k Amplitude

==============================================================

0.6505762067969861 0.9996887750618011 0.5959896142847262

0.8510861716377330 0.9513204018612136 0.6129086911058872

1.5734351519819711 0.5288139520469275 0.8263535410338578

If we compare Figure 1 with Figure 3 it may appear that there is no difference. However Figure 3 actually has six amplitudes present but only four can be distinguished. This can be seen from the output trace of amplitudes given above for the case where the beginning amplitude is 90°.

This is a plot of the successive amplitudes beginning with 45°. The circular angle indicators are scaled to the value of k. A value of k equal to one gives an angle indicator half the length of the arrow. As can be seen the initial value of k is quite small, barely larger than the marker indicating the centre of the circle.



Figure 2 Successive amplitudes beginning with 45°



Figure 3 Successive amplitudes beginning with 90°



Figure 4 Successive amplitudes beginning with 135°

# Elliptic Integrals of the Second Kind

An elliptic integral of the Second Kind in Legendre notation is:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

The Landen transformation does not give a nice neat recurrence relation like the transformation of an elliptic integral of the First Kind. However the following relation in which the elliptic integral of the Second Kind can be written as a transformed integral of the Second Kind plus a transformed integral of the First Kind take away can be established (see Appendix O).

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

with and . The changes in amplitude and modulus are given by:

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

and

|  |  |  |
| --- | --- | --- |
|  |  | (15) |

Now as the modulus converges to unity and the amplitude converges to the elliptic integral of the Second Kind in (41) will converge to

# MATLAB© Second Kind program results

The program to find the elliptic integral of the Second Kind is in Appendix B.

A comparison between the MATLAB© Second Kind inbuilt function results using 40 digit precision and the results using the Landen transformation are given. Differences are found in the fifteenth decimal place.

|  |  |  |
| --- | --- | --- |
| Latitude, eccentricity | MATLAB© ellipticE result | E\_Rec result  ( |
| Second(pi()/648000),  0.08181919104281579 | 0.000  004  848  136  811 | 0.000  004  848  136  810 |
| Minute(pi()/10800),  0.08181919104281579 | 0.000  290  888  208  638 | 0.000  290  888  208  638 |
| Degree(pi()/180),  0.08181919104281579 | 0.017  453  286  588  438 | 0.017  453  286  588  438 |
| pi()/4, 0.08181919104281579 | 0.784  920  272  732  874 | 0.784  920  272  732  875 |
| pi()/3, 0.08181919104281579 | 1.046  168  817  527  900 | 1.046  168  817  527  901 |
| pi()/2, 0.08181919104281579 | 1.568  164  140  912  962 | 1.568  164  140  912  961 |
| 3\*pi()/4, 0.08181919104281579 | 2.351  408  009  093  139 | 2.351  408  009  093  141 |

Table 3

# Meridian Arc Length Formula (Derivation given in Appendix L)

An example of the use of the elliptic integral of the second kind is the formula for an elliptic arc length M (Deakin p. 32).

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

In this formula is the semi-major axis, is the first eccentricity given by

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

is the latitude.

As Equation (16) multiplies the elliptic integral of the second kind by the semi-major axis, which for the GRS80 ellipsoid is 6,378,137.0 metres, the error in the integration of the elliptic integral should be less than (Rösch, 2011, p. 5). This will give an accuracy of 1mm. Rösch (p. 5) uses a tolerance of in his recursive processes. However more work is needed to relate tolerance in the amplitude or tolerance in the modulus to the precision of the result.

# Results of Calculations of Arc Lengths along Meridians

The Meridian distance is given by:

M\_Rec is the MATLAB© program written to evaluate meridian distance. It calls the MATLAB© program in Appendix B to calculate the elliptic integral of the Second Kind. For the purposes of testing, M\_Rec offers the option of calculating meridian distances M for latitudes 30°, 45°, 60° and 90° on Bessel’s ellipsoid ( These distances are given in Dorrer (1999, p97) and Deakin (2014, p34). Running the program with the GRS80 parameters gives the following along with output for Bessel ellipsoid latitudes.

>> M\_Rec

==========================================================

Latitude Semi-major flat Meridian distance

==========================================================

60.0000 6378137 298.257222101 6654072.819367446

----------------------------------------------------------

30.000000 Bessel -- 3319786.509543302

45.000000 Bessel -- 4984439.265470861

60.000000 Bessel -- 6653376.120611615

90.000000 Bessel -- 10000855.764435511

These differ in the eighth decimal place with the results of Dorrer (p97):

|  |  |  |  |
| --- | --- | --- | --- |
| Latitude | Ellipsoid | M\_Rec | Dorrer |
| 30° | Bessel | 3319786.50954330 | 3319786.50954331 |
| 45° | Bessel | 4984439.26547086 | 4984439.26547085 |
| 60° | Bessel | 6653376.12061161 | 6653376.12061161 |
| 90° | Bessel | 10000855.76443551 | 10000855.7644355 |

We can input the WGS-84 ellipsoid parameters – flattening of 298.257223563 and semi-major axis length 6,378,137.0 m – into M\_Rec to compare Landen transformation results with Weintrit’s (2013) table of quadrant distances. The result is 10,001,965.72931271 which differs from the best results in the table – those by Deakin, Bowring and Weintrit - only at the tens of nanometers level. The Landen transformation result is being limited by hardware and/or MATLAB© constraints.

|  |  |
| --- | --- |
| Method | Quadrant |
| AMN, 1987 | 10,001,965.72952860 |
| Bomford, 1985 | 10,001,965.72931360 |
| **M\_Rec (2014)** | **10,001,965.72931271** |
| Deakin, 2010 | 10,001,965.72931270 |
| Bowring, 1983 | 10,001,965.72931270 |
| Weintrit, 2013 | 10,001,965.72931270 |
| Veis-Torge | 10,001,965.72922300 |
| Pallikaris, 2009 | 10,001,965.72590000 |

Table 4 Weintrit's (2013) quadrant distances

Weintrit’s formula is:

The other formulae are based on the binomial expansion of the elliptic integral of the Third Kind.

# Ascending and Descending First Kind transformation

On the following diagram ascending is going down from the middle, descending is going up from the middle. The moduli and amplitudes are transformed (downwards on the diagram from the middle) according to

The moduli and amplitudes are transformed upwards on the diagram from the middle according to:

# Landen’s descending transformation for Second Kind integrals

The ascending transformation for elliptic integrals of the second kind is:

This can be rearranged as:

The new is found from the old by the previously derived formula:

|  |  |  |
| --- | --- | --- |
|  |  | (18) |

And the new amplitude is found by the previously derived formula:

|  |  |  |
| --- | --- | --- |
|  |  | (19) |

# Ascending and Descending Second Kind transformation

On this diagram ascending is going down descending is going up. The moduli and amplitudes are transformed (downwards on the diagram) according to

The moduli and amplitudes are transformed upwards on the diagram according to:

=

=sin

# Inverting Elliptic Integrals of the First Kind

An Elliptic Integral of the First Kind can be written as:

|  |  |  |
| --- | --- | --- |
|  |  | (20) |

The substitution converts this integral to the familiar form:

The inverse of the elliptic integral finds the sine of the upper limit of the integration which is which is the sine of the amplitude.

|  |  |  |
| --- | --- | --- |
|  |  | (21) |

Further the cosine of the amplitude is and the delta amplitude is

These three - - are the Jacobi elliptic functions.

Landen’s descending transformation is:

|  |  |  |
| --- | --- | --- |
|  |  | (22) |

|  |  |  |
| --- | --- | --- |
|  |  | (23) |

|  |  |  |
| --- | --- | --- |
|  |  | (24) |

Now

where

|  |  |  |
| --- | --- | --- |
|  |  | (25) |

Eventually will go to which is the amplitude when goes to zero.

|  |  |  |
| --- | --- | --- |
|  |  | (26) |

|  |  |  |
| --- | --- | --- |
|  |  | (27) |

If we just work with the amplitude then

|  |  |  |
| --- | --- | --- |
|  |  | (28) |

where from (5) and (6)

and

Also we know that

where is the amplitude when goes to zero. close to zero is the terminating condition for the recursive algorithm used in the program to calculate the amplitude.

# MATLAB© First Kind Inversion program results

The code for finding the amplitude given the result of a First Kind integral are in Appendix E. The program is called AMPF\_LDRec for Amplitude, First Kind, Landen Descending and Recursive.

>>AMPF\_LDRec(0.785876554121538,0.0818191910435,0.785876554121538)

Amplitude modulus

3.13823421358965 0.00000070494628

Amplitude modulus

3.13823421358965 0.00000070494628

Amplitude modulus

1.56911710797858 0.00167922039466

Amplitude modulus

3.13823421358965 0.00000070494628

Amplitude modulus

3.13823421358965 0.00000070494628

Amplitude modulus

1.56911710797858 0.00167922039466

Amplitude modulus

0.78539816339745 0.08181919104350

ans =

0.785398163397448

# Evaluating General Elliptic Integrals of the Third Kind using Carlson Symmetric Forms

The Incomplete Elliptic Integral of the Third Kind in Legendre notation with Amplitude , Characteristic , and Modulus is:

|  |  |  |
| --- | --- | --- |
|  |  | (29) |

The Carlson forms use a notation using , , and rather than and .

The Carlson Symmetric Form of the First Kind elliptic integral is (Zill p200):

|  |  |  |
| --- | --- | --- |
|  |  | (30) |

The following is the Carlson Symmetric Form of the Third Kind elliptic integral

|  |  |  |
| --- | --- | --- |
|  |  | (31) |

The Forms are called symmetric because the integrals are unchanged if the arguments are switched around. For example is the same as . With the Form this is true with the but not the .

Running my program R\_JRec with the arguments (1,2,3,0.5) gives the same result as running it with (3,2,1,0.5)

>> R\_JRec(1,2,3,0.5)

ans =0.793964910504615

>> R\_JRec(3,2,1,0.5)

ans =0.793964910504615

## What are functions (brief introduction of terms)?

The standard hypergeometric function is (Debnath 2012):

where is the hypergeometric function defined by:

The terms in the series above form a pattern containing with the ’s forming their own series etc.

A generalised univariate hypergeometric function

is a function which can be written in terms of a hypergeometric series. A hypergeometric series is one in which successive terms can be written as:

A generalised multivariate hypergeometric function

Can be written as:

where the Pocchammer symbol is:

and the “C” Jack function is:

The notation means:

For example:

## Elliptic Integral of the Third Kind with parameters useful in geodesy

The Elliptic Integral of the Third Kind with parameters useful in geodesy in terms of the Carlson Symmetric Forms is:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  | (32) |

|  |  |  |
| --- | --- | --- |
|  |  |  |

Now by the duplication theorem (Press p262):

|  |  |  |
| --- | --- | --- |
|  |  | (33) |

Let , , . Then iterate the series:

Now

|  |  |  |
| --- | --- | --- |
|  |  | (34) |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

Calculating

|  |  |  |
| --- | --- | --- |
|  |  | (35) |

Now

|  |  |  |
| --- | --- | --- |
|  |  | (36) |

where

When the arguments converge to then Equation (35) becomes:

|  |  |  |
| --- | --- | --- |
|  |  | (37) |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

The program to implement this I have called ThirdCarlson. The results of running this program and comparing it to the inbuilt MATLAB© program ‘ellipticPI(n,,m) can be found in Appendix J. There is no difference between my program and MATLAB©’s at the fourteenth decimal place.

Carlson’s form allows arbitrary ranges of integration (Press, 1992, p. 262).

# Inverting Elliptic Integrals of the Third Kind using Carlson form and a binary technique

Appendix G has the code for inverting an elliptic integral of the Third Kind using a simple binary technique. Work was done trying to get a Newton-Raphson technique to work but this was abandoned as there were problems with convergence (see appendix R).

The binary technique is extremely robust but takes about fifty iterations to achieve 1E-15 precision. The following is a test of this program using and and ranging from

-2\*1.92764575848523 to 2\*1.92764575848523.



Figure 5 The latitudes which result from equally spaced distances on the ellipse

# The Direct Geodetic Problem

A geodesic is the shortest path between two points on a curved surface. Consider Figure 4 (from Deakin and Hunter 2007) which shows the shortest path on an ellipsoid between A and B. The direct geodesic problem begins with a position A with latitude and longitude , an azimuth which is the direction to head off in and a distance to go and then determines the arrival point B such that the distance from A to B is the shortest distance from A to B.

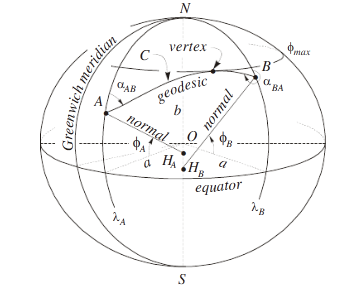


Figure 6

The characteristic equation (Clairaut’s equation) to geodesics is:

(5)

where is the radius of curvature in the prime vertical plane

is latitude

is azimuth

is the semi-major axis of the ellipsoid

is parametric latitude

C is a constant.

When the azimuth is then and is a minimum as is a constant. being a minimum means that is a maximum and this is called . The geodesic then reaches maximum latitude.

# Geodesics (Rollin’s Method)

Using a change of variable the formula for the length of a geodesic on the ellipsoid becomes the same form as an arc length on an ellipse. The program already given to find arc lengths can then be used to solve the distance part of the inverse problem.

The formula for a latitude difference on the ellipse is (Rollins, 2010):

|  |  |  |
| --- | --- | --- |
|  |  | (38) |

The formula for the geodesic length between two points of a geodesic is:

|  |  |  |
| --- | --- | --- |
|  |  | (39) |

The constant is the Clairaut constant of the geodesic and is found from the following equation:

|  |  |  |
| --- | --- | --- |
|  |  | (40) |

To find the Clairaut constant we must know the azimuth , the latitude of the starting point on the ellipsoid and the first eccentricity .

is the reduced latitude and is found from:

|  |  |  |
| --- | --- | --- |
|  |  | (41) |

Equations (1) and (2) work when the geodesic does not pass through a vertex. When the geodesic passes through a vertex then the integration is divided up into two parts: one up to the vertex and then another from the vertex to the destination (Rollins, 2010, p. 21). For example the geodesic from New York to Paris would be made up of two calculations: from New York up to the vertex and from the vertex to Paris. That is:

By using a change of variable

|  |  |  |
| --- | --- | --- |
|  |  | (42) |

with at some Equator crossing and with the constant defined by (Rollins, 2010, p. 21):

|  |  |  |
| --- | --- | --- |
|  |  | (43) |

This change of variable changes Equations (38) and (39) into (Rollins, 2010, p. 22)

|  |  |  |
| --- | --- | --- |
|  |  | (44) |

This is an elliptic integral of the Third Kind with and being given by equations (40) and (43) and being the first eccentricity. The corresponding geodesic distance is:

|  |  |  |
| --- | --- | --- |
|  |  | (45) |

which is also an elliptic integral of the Third Kind with being the semi-major axis of the ellipse.

Now this is the formula for meridian arc length on an ellipse having a semi-major axis of length (Rollins, 2010, p. 22):

and eccentricity

between latitudes and .

Now the direct problem will require finding with all the other variables known

An advantage of equation (44) and (45) over (38) and (39) is that there is no need to break up geodesics which pass through a vertex (Rollins, 2010, p. 23).

Now Equation (44) is a general elliptic integral of the Third Kind so the method which will be used is the Carlson Duplication technique.

Example (from Rollins p24):

Consider a point near Santiago Chile with ° and °. The direction to go is ° which is ° south of West (the azimuth is bearing from North). Using the GRS80 ellipsoid (6378137, ) find the location after 10,000km of travel along the geodesic.

Solution:

To find :

Equation (43) gives the value of :

Now the rules for assigning a sign to are as follows (Rollins, 2010, p. 23):

|  |  |
| --- | --- |
| quadrant |  |
| NE or NW |  |
| SE or SW |  |

So this gives a negative sign for here where is in the SW quadrant (

To find we use the change of variable equation (4b):

In Equation (45) is the only unknown quantity.

|  |  |  |
| --- | --- | --- |
|  |  |  |

The MATLAB© program ‘Direct.m’ gives the following result for which agrees with Rollins p24.

=========================================================

Latitude Longitude Geodesic Distance ThetaB

=========================================================

-34.0000 -72.0000 1E7 165.568952965

---------------------------------------------------------

Now that both and are known Equation (44) can be used to find longitude difference.

|  |  |  |
| --- | --- | --- |
|  |  |  |

The integrand is a general Third Kind integral:

Taking the upper limit of integration as the first integral:

The characteristic is and the modulus is The values of these in this current example are (with the modulus squared given):

==========================================================

Characteristic modulus ThetaB rads ThetaB degrees

==========================================================

3.335145e-01 4.725117e-02 2.889723e+00 165.568952965

----------------------------------------------------------

The extreme latitudes correspond to (Rollins 24).

# My Direct.m results compared with Karney’s GEODRECKON

Karney’s program geodreckon.m accepts arguments of latitude, longitude, the geodesic distance and the azimuth. These are in degrees or metres. There are two optional arguments of ellipsoid and arcmode.

My program accepts radians and metres. If there are not six arguments when the function is called then a window pops up to allow user input of arguments in the DD.MMSS format.

Using Rollins example of a point near Chile of 34° South and 72° West an azimuth of -100 and a geodesic distance of 10,000,000 metres my program Direct.m gives the following output which agrees with Rollins (p24). Rollins reports at most nine decimal places though the machinery and software he uses is capable of twenty digit precision (p26):

>> Direct

==========================================================

Next lat(D) Next long(D) Forward Azimuth(D)

==========================================================

-8.274933372023414 -167.428228335873230 -55.673724820079144

----------------------------------------------------------

Karney’s program GEODRECKON gives the following results:

|  |  |  |  |
| --- | --- | --- | --- |
| S = 10,000,000 | GEODRECKON.m | Direct.m (newton-raphson) | Direct.m (binary) |
| latitude | -8.27493337202321 | -8.274933372023414 | -8.274933372023227 |
| longitude | -167.428228335873 | -167.428228335873230 | -167.428228335873540 |
| Forward azimuth | -55.6737248200791 | -55.673724820079144 | -55.673724820079109 |

I decided to just use the binary search method in my program due to its robustness. Typically 50 iterations are required to achieve 1E-15 precision but this can be improved by choosing a better starting point.

Starting at the same point near Chile and entering a geodesic distance of 100,000 metres and running my program Geodesy\_Tester\_Direct for 1000 loops gives the following results (first and last few shown). R stands for Rollins’ method which my program Direct.m uses and K stands for the results gained by running Karney’s program ‘geodreckon’.

My program and Karney’s program were running independently so they were keeping their own successive latitudes and longitudes (see Appendix I for program). After the 1000 loops there are differences only at the twelfth decimal place in latitude, longitude and azimuth.

>> Geodesy\_Tester\_Direct

==========================================================

Starting Point with distance, 100000.00000 metres

lat(D) long(D) Forward Azimuth(D)

==========================================================

-34.000000000000000 -72.000000000000000 -100.000000000000000

----------------------------------------------------------

R -34.151914446601666 -73.067907701385167 -99.401647957809416

K -34.151914446601687 -73.067907701385195 -99.401647957809374

----------------------------------------------------------

R -34.294503066081532 -74.139528183321204 -98.798936391946455

K -34.294503066081539 -74.139528183321147 -98.798936391946427

----------------------------------------------------------

R -34.427684932956431 -75.214655787426381 -98.192114686882149

K -34.427684932956439 -75.214655787426352 -98.192114686882135

----------------------------------------------------------

R -34.551383759367404 -76.293077068212952 -97.581439837046787

K -34.551383759367418 -76.293077068212938 -97.581439837046773

----------------------------------------------------------

.

.

.

.

----------------------------------------------------------

R 33.655457785220477 112.668404470691270 -78.773556943984090

K 33.655457785222019 112.668404470692000 -78.773556943978988

----------------------------------------------------------

R 33.826443100681544 111.608819971748360 -79.362103702458242

K 33.826443100683157 111.608819971749140 -79.362103702453169

----------------------------------------------------------

R 33.988284453721590 110.545122269365360 -79.955501746027750

K 33.988284453723274 110.545122269366120 -79.955501746022691

----------------------------------------------------------

R 34.140891903356135 109.477499835768640 -80.553518467416453

K 34.140891903357904 109.477499835769380 -80.553518467411479

----------------------------------------------------------

The successive latitude and longitude points were plotted on a MATLAB© map. The distance of 100,000 km represents about 2.5 times around the earth.



Figure 7 1000 successive 100km geodesics from a point near Chile

Starting from Melbourne the following path resulted. After 1000 loops the comparison with Karney’s latitude, longitude and azimuth was:

----------------------------------------------------------

R 37.919331449846460 -33.604736343534313 -80.565745425055454

K 37.919331449848727 -33.604736343534199 -80.565745425055880



Figure 8 1000 successive 100km geodesics from a point near Chile

# Gauss-Krüger Coordinates

Isometric latitude is transformed into parametric latitude by means of the following equation (Dorrer p98):

This is the Lambertian of and is also called the inverse Gudermannian.

The isometric coordinate pair is defined as the complex variable

From this complex longitude

## amplitude in Landen’s descending transformation

The method used in the program in the appendix is

# Calculation of UTM and Geodetic Coordinates

A Transverse Mercator Projection is defined by the following equations (Dozier, 1980, p. 2):

where is the Jacobian elliptic function

The Gauss-Krüger projection is one kind of Transverse Mercator Projection is:

which can be written as:

is an elliptic integral of the Second Kind. This can be found using the arithmetic-geometric mean as described in section 5 above.

An expression for is:

K is the elliptic integral of the First Kind and can be found using the arithmetic-geometric mean as described in section 4 above.

q is the nome and is defined by:

# Safeguards against the computer not preserving precision when implementing recursion

The advantages of using a recursion algorithm:

1. Once the precision is specified then the recursions are performed until the precision is met.
2. The intermediate results are passed on to the next level
3. The program structure remains the same regardless of the ellipsoid used. With different constants just feed the new constants into the top level

The disadvantages of using a recursion algorithm

1. The computer and software need to be able to manage the recursion without losing precision.

# Conclusions:

Carlson’s forms can be used for all three kinds of elliptic integrals when going forward from angle to result of integral and give arbitrary precision.

The MATLAB© inbuilt functions ellipticF, ellipticE and ellipticPI are precise to machine precision and so they should be because they use a calculation method based on the arithmetic-geometric mean which is a specialized version of the Carlson Duplication technique. The MATLAB© Second Kind integral function ellipticE matches my Second Kind integral using the Landen transformation to the fourteenth decimal place (Section 6). So when writing geodetic programs in MATLAB© then we know we have at least 14 decimal place precision with the elliptic integrals.

Inversion of the First Kind integral is precise using the Landen transformation without any use of a numerical technique.

The programs which invert the Third Kind integrals have used a simple binary search as this is easy to program and reliable with the downside that about fifty iterations are required for femtometre precision. The final program here – Direct.m – which finds successive latitude and longitude points when called to calculate 1000 successive geodesics takes 29.13 seconds to run. Calling the Karney program takes 5.6 seconds to do the same job. Calling both programs takes 29.73 seconds. This means the Karney program without the calling program is taking 0.6 seconds to calculate the 1000 geodesics. My program Direct.m is taking 24 seconds. It is the binary search which takes up most of the time. So my Direct.m is an example of the precision of the Carlson method rather than a fast method of finding geodesics. Karney’s program uses the auxiliary sphere, series reversion and a Newton-Raphson step to increase precision.

Comparing my program to the precision of the Karney program ‘geodreckon.m’ it matches ‘geodreckon.m’s results to the eleventh decimal place. This shows that using femtometre precision allows the preservation of nanometre precision after a lot of accumulated small error. The practical consequence of this is that when finding UTM grid coordinates using a high precision method based on Carlson Duplication then mm errors may be found at the edges of the UTM zone. If so then this has consequences for comparing the recorded permanent mark grid coordinates with what the GPS satellite says it should be.

In addition the Carlson Duplication technique does not produce singularities at the poles and this along with precision over the whole ellipsoid rather than just in 6° widths provides a precise grid system for the entire ellipsoid.

These things of course are also offered by Karney’s methods using an auxiliary sphere but the advantage of using direct calculation is a pedagogical one; there is no need to introduce the extra concept of an auxiliary sphere. This is also an advantage of Carlson Duplication over Landen transformation as the Landen transformation deals with two types of transformed quantities (angle and modulus) while Carlson Duplication only deals with the independent variables. So using Carlson Duplication reduces the quantity of concepts to deal with.

In addition to the use of the Landen transformation and Carlson Forms these methods were implemented using recursive techniques. The advantage of the recursive technique is that it can give arbitrary accuracy up to the limits of machinery and software.

Another advantage of the recursive approach is that it can save space in documentation and programming. There is no need for lists of terms and the recursive technique means that the work of the program is done in two steps: the recursive line and the terminating condition.

So the final conclusion is that Carlson Duplication implemented using a recursive algorithm should be used for determining arc lengths for a worldwide grid coordinate system.

**Bibliography**

*Geocentric Datum of Australia*. (2014). Retrieved from Geoscience Australia: http://www.ga.gov.au/scientific-topics/positioning-navigation/geodesy/geodetic-datums/gda

Almkvist, G. (1988). Gauss, Landen, Ramanujan, the Arithmetic-Geometric Mean, Ellipses, pi, and the Ladies Diary. *Amer. Math. Monthly*, 585–607.

Blumel, R. (2011). *Advanced Quantum Mechanics: The Classical-Quantum Connection.* Connecticut: Jones and Bartlett.

Borwein, J. B. (1984). The Arithmetic Geometric Mean and Fast Computation of Elementary Functions. *Society for Industrial and Applied Mathematics*.

Carlson, B. C. (1965). On Computing Elliptic Integrals and Functions. *J. Maths and Physics*.

Carlson, B. C. (1979). Computing Elliptic Integrals by Duplication. *Numerische Mathematik*(33), 1-16.

Carlson, B. C. (1995). *Numerical Calculation of Real or Complex Elliptic Integrals.* Retrieved from arxiv.org.

Carlson, D. Z. (1969). Symmetric Elliptic Integrals of the Third Kind. *www.ams.org*.

Deakin, R. (2010). *Geometric Geodesy Part A.* RMIT.

Deakin, R. (2014). *Elliptic Integrals and Landen's Transformation: What I should've known for Geodesy.*

Debnath, L. (2012). *Non-linear Partial Differential Equations for Scientists and Engineers.* New York: Springer.

Diarmuid, M. (2008). *Integrable Systems in Celestial Mechanics.* Boston: Birkhäuser.

Dorrer, E. (1999). From Elliptic Arc Length to Gauss-Krueger Coordinates by Analytical Continuation. *Geodesy and Geoinformatics*.

Dozier, J. (1980). *Improved Algorithm for Calculation of UTM and Geodetic Coordinates.* NOAA Technical Report NESS 81, U.S. Department of Commerce.

Fukushima, T. (2012). Numerical Inversion of General Incomplete Elliptic Integral. *Journal of Computational and Applied Mathematics*.

Gerstl, M. (1984). *Die Gauss-Krügersche Abbildung des Erdellipsoides mit direkter Berechnung der elliptischen Integrale durch Landentransformation.* Munich: Verlag.

Gray. (2001). Automatic Reduction of Elliptic Integrals Using Carlson's Relations.

Hankin, R. (2006). *Introducing Elliptic, an R package for elliptic and modular functions.* Retrieved from jstatsoft: www.jstatsoft.org/v15/i07/paper

Hunter, R. D. (2007). *Geodesics on an Ellipsoid - Pottman's Method.* Retrieved from academia.edu.

Jameson, G. (2014). *Ellitpic integrals, the arithmetic-geometric mean and the Brent-Salamin algorithm for pi.* Retrieved from www.maths.lancs.ac.uk/~jameson/ellagm.pdf

Karney, C. (2011). Geodesics on an ellipsoid of revolution. *SRI International*.

Karney, C. (2012). *Geodesics on an Ellipsoid of Revolution.* Retrieved September 30, 2014, from MATLAB Central.

Khan, M. F. (2013). Arc Length of an Elliptical Curve. *International Journal of Scientific and Research Publications*, 1-5.

King, L. V. (1924). *On the Direct Numerical Calculation of Elliptic Functions and Integrals.* London: CUP.

Klotz, J. (1993). Eine Analytische Lösung der Gauss-Krüger-Abbildung. *ZfV 3/1993*, 106-116.

Kos, S. a. (2012). On the Mathematics of Navigational Calculations for Meridian Sailing. *Solstice: An Electronic Journal of Geography and Mathematics*.

Krakiwsky, E. (1986). *Geodesy the Concepts.* Amsterdam: Elsevier.

Landen, J. (1775, January). *An Investigation of a General Theorem for Finding the Length of Any Arc of Any Conic Hyperbola, by means of Two Elliptic Arcs, with Some Other new and Useful Theorems Deduced Therefrom.* Retrieved October 10, 2014, from https://archive.org/details/jstor-106197

Press, F. T. (1992). *Numerical Recipes in C: the Art of Scientific Computing.* New York: University of Cambridge.

Rollins, C. (2010, January). An Integral for Geodesic Length after Derivations by P. D. Thomas. *Survey Review*, pp. 20-26.

Rösch, N. (2011). The Derivation of Algorithms to Compute Elliptic Integrals of the First and Second Kind by Landen Transformation. *Bol. Ciênc. Geod., sec. Artigos, Curitaba, 17*, 03-22.

Sjöberg, L. E. (2012). *Solutions to the ellipsoidal Clairaut constant and the inverse geodetic problem by numerical integration.* Retrieved October 14, 2014, from Journal of Geodetic Science: http://adsabs.harvard.edu/abs/2012JGeoS...2..162S

Tkachev, V. G. (n.d.). *Elliptic Functions: Introduction Course.* Retrieved from URL: http://www.math.kth.se/˜tkatchev

Villarino, M. B. (6 July 2005). A Direct Proof of Landen’s Transformation. *math.CA*.

Weintrit, A. (2013). So, What is Actually the Distance from the Equator to the Pole? - Overview of the Meridian Distance Approximations. *TransNav, 7*(2).

Appendices

MATLAB© code for determining elliptic integrals of the First Kind using Landen’s ascending transformation

function u = F\_LARec(phi,k )

%F\_LAREC Finds the elliptic integral of the First Kind

% with arguments of

% the amplitude and the modulus.

% The Landen Transformation says: F(k,phi)=2/(1+k) F(q,phi1) where

% q=(2k^0.5)/(1+k) and

% sin(2psi-phi)=ksin(phi)

% 2psi-phi=arcsin(ksin(phi))

% 2psi=arcsin(ksin(phi))+phi

% psi=[arcsin(ksin(phi))+phi]/2

% After a while the transformation leaves k = 1 and so the

% last integration equals lntan(pi/4+phihat/2). This naturally lends

% itself to a recursion algorithm.

% Type 'clf' at the MATLAB© prompt to clear existing plots

% Test input F\_LARec(pi()/4,0.0818191910435). Note latitude is in

% radians.

% Answer should be: 0.785876554121538

format long

% q and phi1 are the new modulus and amplitude

Plot\_Angle(phi,k);

fprintf(1,'%20s %20.16f %20.16f\n','-',k,phi)

q = (2\*k^0.5)/(1+k);

phi1=(asin(k\*sin(phi))+phi)/2;

% This is the difference between the old amplitude and the new

% amplitude

diff=abs(phi-phi1);

% Set tolerance

if diff < 1E-15

% print the headers for the output report

fprintf(1,'%16s %16s %20s\n','Integral', 'k', 'amplitude')

u = (2/(1+k))\*log(tan(pi()/4 + phi1/2));

return;

end

% This calls the function recursively. u is the integral

u=(2/(1+k))\*F\_LARec(phi1,q);

fprintf(1,'%20.16f %20.16f %20.16f\n',u,q,phi1)

end

MATLAB© code for determining elliptic integral of the Second Kind

Theory summary

Equation (41) says:

(41)

Where and are given by the Landen transformations.

(42)

and

Writing (42) explicitly we have:

Eventually will go to unity. This leaves the last recursion as:

Code

function [ u,Erl] = E\_Rec( phi,k,Erl )

%E\_REC Finds an elliptic integral of the second kind recursively

% Input takes the latitude in radians and first eccentricity. The first

% time the program is called there are only two input arguments and this

% is tested in the program. This allows the program to know that the

% recursion level is zero.

% Test input E\_Rec(pi()/3,0.0818191910435). Note latitude is in radians.

% Remove percentage's from fprintf's to get working printouts.

format long

nargin; % program said error without this line

if nargin < 3

Erl = 0;

% fprintf(1,'%16s %16s %16s \n','E\_Recursion Level','Amplitude','modulus')

% fprintf(1,'%16u %20.14f %20.14f\n',Erl,phi,k)

else

Erl = Erl + 1;

% fprintf(1,'%16s %16s %16s \n','E\_Recursion Level','Amplitude','modulus')

% fprintf(1,'%16u %20.14f %20.14f\n',Erl,phi,k)

end

if Erl == 0

end

% q and phi1 are the new modulus and amplitude

q = (2\*k^0.5)/(1+k);

phi1=(asin(k\*sin(phi))+phi)/2;

% This is the difference between the old amplitude and the new amplitude

diff=abs(phi-phi1);

% Set tolerance

if diff < 1E-15

u1 = sin(phi1); % Second Kind limit

u2 = log(tan(pi()/4 + phi1/2)); % First Kind limit

u=(1+k)\*u1+(1-k)\*u2-k\*sin(phi); %Note phi is previous

return;

end

% This calls the function recursively

u=(1+k)\*E\_Rec(phi1,q,Erl) + (1-k)\*F\_LARec(phi1,q)-k\*sin(phi);

end

Results

MATLAB© result for the elliptic integral of the second kind for 60° is:

ellipticE(1.047197551196598,0.0818191910435^2)



The result for E\_Rec is:

>> E\_Rec(pi()/3,0.0818191910435)

ans =

1.046168817527883

E\_Rec matches the MATLAB© function.

MATLAB© code for determining meridian distance M.

function M\_Rec

% M\_REC This function gives meridian distance by calling the programs

% F\_Rec and E\_Rec.

format long g

% This creates a window for the user to enter latitude,

% semi-major axis and flat

prompt = {'Enter latitude DD.MMSS','Enter semi-major axis a','enter flat','Run Test?'};

dlg\_title = 'Input phi';

num\_lines = 1;

def = {'60.0000','6378137','298.257222101','Yes'};

answer = inputdlg(prompt,dlg\_title,num\_lines,def);

phi0 = abs(str2num(answer{1}));

a = str2num(answer{2});

flat = str2num(answer{3});

runtest = answer{4};

%convert input latitude to radians

%Change DD.MMSS into whole degrees, minutes, seconds

phi0deg = fix(phi0);

mins\_secs = phi0 - phi0deg;

phi0minutes = abs(fix(mins\_secs\*100));

phi0seconds = abs(fix((phi0 - phi0deg - phi0minutes/100)\*10000));

phi0r = (phi0deg+(phi0minutes/60)+(phi0seconds/3600))\*(pi()/180);

% convert flat into flattening and find first eccentricity

f=1/flat;

e = (f\*(2-f))^0.5;

% find meridian distance by using the meridian formula which

% calls the program E\_Rec

M=a\*(E\_Rec(phi0r,e)-(e^2\*sin(phi0r)\*cos(phi0r))/(((1-e^2\*(sin(phi0r))^2))^0.5));

fprintf(1,'==========================================================\n');

fprintf(1,'%12s %12s %5s %26s\n','Latitude', 'Semi-major', 'flat','Meridian distance');

fprintf(1,'==========================================================\n');

fprintf(1,'%10s %10s %16s %19.9f\n',answer{1}, answer{2},answer{3}, M);

fprintf(1,'----------------------------------------------------------\n');

if runtest == 'Yes'

lats = (pi()/180)\*[30,45,60,90];

a=6377397.155; %Bessel's ellipsoid

e = 0.08169683121517; %Bessel's ellipsoid

for n = 1:4

M=a\*(E\_Rec(lats(n),e)-(e^2\*sin(lats(n))\*cos(lats(n)))/(((1-e^2\*(sin(lats(n)))^2))^0.5));

fprintf(1,'%10f %10s %16s %19.9f\n',180/pi()\*lats(n), 'Bessel','--', M)

end

end

end

MATLAB© code First Kind integrals using Landen’s descending transformation

function u = F\_LDRec(psi,q )

%F\_LDREC Finds the value of an elliptic integral of the First Kind

% This function takes in the value of the integral (u) and the modulus

% The Descending Landen Transformation says: F(psi,q)=(1+k)/2 F(k,phi) where

% k=(1-?(1-q^2))/(1+?(1-q^2)) and

% phi=arctan(((1-k))/((1+k))tan?)+?

% Test with F\_LDRec(pi()/4,0.0818191910435)

% Answer should be 0.785876554121538

% After a while the transformation leaves k = 0 and so the

% last integration equals psihat.

% Test input F\_Rec(pi()/2,0.0818191910435). Note latitude is in radians.

% Remove percentage's from fprintf's to get working printouts.

format long

fprintf(1,'%20s\n','eps')

fprintf(1,'%20.19f\n',eps)

% k and phi are the new modulus and amplitude. This finds the modulo(180degrees)

% and doubles this for the new phi. If the previous amplitude is

% in the second quadrant then add pi() to the new amplitude. This needs

% further refinement to cater for particular borderline cases.

k = (1-(1-q^2)^0.5)/(1+(1-q^2)^0.5);

n=floor(psi/(pi()));

psito180=psi-n\*pi();

if psito180 > pi()/2 & psi < pi() % psi in second quadrant

phi=atan(((1-k)/(1+k))\*tan(psito180))+pi()+2\*n\*pi() +psi;

else

phi=atan(((1-k)/(1+k))\*tan(psito180))+2\*n\*pi()+psi;

end

% k will go to zero

% Set tolerance

if k < 1E-15

u = (1/2)\*phi;

fprintf(1,'%16s %16s %16s\n','Amplitude','Modulus','Integral')

fprintf(1,'%20.14f %20.14f %20.14f\n',psi,q,u)

return;

end

% This calls the function recursively

u=((1+k)/2)\*F\_LDRec(phi,k);

fprintf(1,'%16s %16s %16s\n','Amplitude','Modulus','Integral')

fprintf(1,'%20.14f %20.14f %20.14f\n',psi,q,u)

end

MATLAB© code for finding the amplitude of a First Kind elliptic integral given the result of the integral (inverting the First Kind integral).

function psi = AMPF\_LDRec(u,q,psihat)

%AMPF\_LDREC Finds the value of the amplitude of an elliptic integral of the First Kind

% This function takes in the value of the integral (u) and the modulus

% (q).

% To begin with the value of psihat is input as the same as u the integral.

% Test with AMPF\_LDRec(0.785876554121538,0.0818191910435,0.785876554121538)

% Answer should be pi()/4=0.785398163397448

% After a while the transformation leaves k = 0 and so the

% last integration equals psihat.

% Remove percentage's from fprintf's to get working printouts.

format long

%fprintf(1,'%20s\n','eps')

%fprintf(1,'%20.19f\n',eps)

% k and u1 are the new modulus and integral.

% psihat1 is the cumulative product which eventually gives the final

% amplitude.

k = (1-(1-q^2)^0.5)/(1+(1-q^2)^0.5);

u1=(2/(1+k))\*u;

psihatcum=psihat\*(2/(1+k));

% k will go to zero

% Set tolerance

if k < 1E-15

psi = psihatcum/(2/(1+k)); % This reverses line above for last go

% fprintf(1,'%16s %16s %16s\n','Amplitude','Modulus','Integral')

% fprintf(1,'%20.14f %20.14f %20.14f\n',psihat,k,u)

return;

end

% This calls the function recursively

psi=(asin(k\*sin(AMPF\_LDRec(u1,k,psihatcum)))+AMPF\_LDRec(u1,k,psihatcum))/2;

fprintf(1,'%16s %16s\n','Amplitude','modulus')

fprintf(1,'%20.14f %20.14f\n',psi,q)

end

MATLAB© code for determining elliptic integral of the Third Kind using Carlson

function THIRDC = ThirdCarlson(characteristic, modulus, amplitde)

%THIRDCARLSON Calculates the elliptic integral of the Third Kind

% The window input accepts DD.MMSS format. The function input accepts

% radians.

% This uses the Carlson form of the Third Kind integral

% sin?R\_F(cos^2?,1-k^2\*sin^2\*?,1)+

% +(1/3)nsin^3?R\_j(cos^2?,1-k^2sin^2?,1-nsin^2?)

format long g

% This creates a window for the user to enter latitude,

% semi-major axis and flat

if nargin == 3

n = characteristic;

k = modulus;

phi = amplitde;

else

prompt = {'Enter characteristic n','Enter modulus k','Amplitude phi'};

dlg\_title = 'Input phi';

num\_lines = 1;

def = {'0.333514524427718','0.08181919104281579','60.0000'};

answer = inputdlg(prompt,dlg\_title,num\_lines,def);

n = str2num(answer{1});

k = str2num(answer{2});

phi = DecDMStoRad(str2num(answer{3}));

end

% Plot the input angle

% Plot\_Angle(phi,k);

if phi < 0

PositiveAngle = -1;

phi = -1\*phi;

else

PositiveAngle = 1;

end

if phi > pi()/2

% Find the whole number of quadrants

whole\_quadrants = floor(2\*phi/(pi()));

if (whole\_quadrants/2-floor(whole\_quadrants/2))<0.1 %whole quads even

first\_quad\_amp = phi - whole\_quadrants\*pi()/2;

THIRD = (whole\_quadrants\*ThirdCarlson(n,k,pi()/2))+ThirdCarlson(n,k,first\_quad\_amp);

else

first\_quad\_amp = (1+whole\_quadrants)\*pi()/2-phi; %whole quads odd

THIRD = (whole\_quadrants+1)\*ThirdCarlson(n,k,pi()/2)-ThirdCarlson(n,k,first\_quad\_amp);

end

else

%fprintf(1,'==========================================================\n');

%fprintf(1,'%15s %15s %15s %15s\n', 'x','y','z','p')

%fprintf(1,'==========================================================\n');

% Plot the first quadrant angle

% Plot\_Angle(phi,k);

% Call the functions to find the Third Kind integral

THIRD=sin(phi)\*R\_F((cos(phi))^2,1-k^2\*(sin(phi))^2,1)+...

((1/3)\*n\*(sin(phi))^3)\*R\_JRec((cos(phi))^2,1-k^2\*(sin(phi))^2,1,1-n\*(sin(phi))^2);

end

THIRDC = PositiveAngle\*THIRD;

end

MATLAB© code for determining the inverse of an elliptic integral of the Third Kind using Carlson and a binary search technique.

function phi = ThirdCarlsonInv\_Binary(u,characteristic,modulus)

%THIRDCARLSONINVERSION Inverts ThirdCarlson using Newton-Raphson

% The default values in the dialog box correspond to the default values

if nargin == 3

multiplying\_factor = 1;

end

format long g

% This creates a window for the user to enter latitude,

% semi-major axis and flat

if nargin == 3

arclength = u;

n = characteristic;

k = modulus;

else

N=20; % Makes the dialog box wide enough so that the title can be read

prompt = {'Arc Length u','Enter characteristic n','Enter modulus k'};

dlg\_title = 'ThirdCarlson';

num\_lines = 1;

def = {'1.39404451505814 ','0.333514524427718','0.08181919104281579'};

%answer = inputdlg(prompt,dlg\_title,num\_lines,def);

answer = inputdlg(prompt,dlg\_title,[1, length(dlg\_title)+N],def);

%Change DD.MMSS into whole degrees, minutes, seconds

arclength = str2num(answer{1});

n = str2num(answer{2});

k = str2num(answer{3});

end

% Change arclength into quadrant arc lengths plus an arc length in the

% first quadrant

evenquads = 0;

% If in an even quadrant (2,4 ...) then work with the complement of the

% angle

whole\_quadrant\_arclength = ThirdCarlson(n,k,pi()/2);

whole\_quadrants = floor(arclength/whole\_quadrant\_arclength);

if (whole\_quadrants/2-floor(whole\_quadrants/2))<0.1 %whole quads even

evenquads = 1;

first\_quadrant\_arclength = arclength - whole\_quadrants\*whole\_quadrant\_arclength;

else

first\_quadrant\_arclength = (1+whole\_quadrants)\*whole\_quadrant\_arclength-arclength; %whole quads odd

end

% Binary Search

%fprintf(1,'=============================================================================\n');

%fprintf(1,'%20s %20s %20s %20s\n','Theta0','fencelow','fencehigh','fx0');

%fprintf(1,'=============================================================================\n');

Theta0 = pi()/4;

fencelow = 0;

fencehigh = pi()/2;

countA = 0;

diff = 1;

while diff > 1E-12

fx0=sin(Theta0)\*R\_F((cos(Theta0))^2,1-k^2\*(sin(Theta0))^2,1)+...

(1/3)\*n\*(sin(Theta0))^3\*...

R\_JRec((cos(Theta0))^2,1-k^2\*(sin(Theta0))^2,1,1-n\*(sin(Theta0))^2)-...

first\_quadrant\_arclength/multiplying\_factor;

if fx0 == 0 %found solution

break

end

if countA == 50

break

end

if fx0 < 0

fencelow = Theta0;

Theta0 = (Theta0 + fencehigh)/2;

else

fencehigh = Theta0;

Theta0 = (Theta0 + fencelow)/2;

end

%fprintf(1,'%20.15f %20.15f %20.15f %20.15f\n',Theta0,fencelow,fencehigh,fx0);

countA = countA + 1;

end

% Plot the found first quadrant angle

%?Plot\_Angle(Theta0,k);

% if in the 3rd, 5th ... quadrant then just add the found first quadrant

% angle to 180°, 360° etc. If in the 2nd, 4th ... quadrant then take away

% the found first quadrant angle from the next higher complete quadrant

% angle.

if evenquads %whole quads even

phi = whole\_quadrants\*pi()/2+Theta0;

else

phi = (1+whole\_quadrants)\*pi()/2-Theta0; %whole quads odd

end

% Plot\_Angle(phi,k);

end

MATLAB© code for solving the direct problem

function NextPoint = Direct(lat, long, azi,s,a,flat)

%UNTITLED Summary of this function goes here

% This program solves the direct problem.With an input of a point (A)

% with lat and long along with a distance s and an azimuth A

% then point (B) is returned with latitude of B and longitude of B

% along with the forward azimuth.

% The procedure is as follows:

% Do the Thomas change of variable.

% Calculate the constant in front of the integral

% Do Binary Search to find ThetaB

% Find the longitude difference

% From the starting longitude find the resulting longitude

format long g

% If there are six arguments to the function accept these

if nargin == 6

latitudeA = lat;

longitudeA = long;

azimuthA = azi;

distance = s;

else

prompt = {'Enter latitude A DD.MMSS','Enter longitude A DD.MMSS',...

'Azimuth A','Enter geodesic distance AB','Enter semi-major axis a',...

'enter flat'};

dlg\_title = 'Input phi';

num\_lines = 1;

def = {'-34','-72','-100','1E7','6378137','298.257222101'};

answer = inputdlg(prompt,dlg\_title,num\_lines,def);

%Change DD.MMSS into whole degrees, minutes, seconds

latitudeA = DecDMStoRad(str2num(answer{1}));

longitudeA = DecDMStoRad(str2num(answer{2}));

azimuthA = DecDMStoRad(str2num(answer{3}));

distance = str2num(answer{4});

a = str2num(answer{5});

flat = str2num(answer{6});

end

% westward azimuth

if (azimuthA > -pi()) & (azimuthA < 0)

westward = 1;

else

westward = 0;

end

% convert flat into flattening and find first eccentricity

f=1/flat;

e = (f\*(2-f))^0.5;

% Find the Clairaut constant

C = (sin(azimuthA)\*cos(latitudeA))/(1-e^2\*(sin(latitudeA))^2)^0.5;

% Find modulus k

k = ((1-C^2)/(1-C^2\*e^2))^0.5;

% Assign a sign to k. Positive if azimuthA in NE

% or NW quadrant.

% SE and SW quadrants negative

if (azimuthA < pi()/2) & (azimuthA > -pi()/2)

k = abs(k);

else

k = -abs(k);

end

% Thomas change of variable sin?=ksin?

% sin?=(sin?)/k

% ?=arcsin((sin?)/k)

ThetaA = asin((sin(latitudeA)/k));

% Calculate m (=k^2 in the notation)

m = k^2\*e^2;

%binary

% To call the CarlsonThirdInv program the value of the integral will be the

% geodesic length multiplied by sqrt(1-c^2e^2)divided by a\*(1-e^2) added to

% the ThirdCarlson integral with ThetaA as the phi, n=k^2\*e^2, k= ke.

% So call ThirdCarlsonInv\_Bin(distance, k^2\*e^2, k\*e)

% This is Rollins (2010) equation (6)

new\_integral\_result = distance\*(1-C^2\*e^2)^0.5/(a\*(1-e^2))+ThirdCarlson(k^2\*e^2,k\*e,ThetaA);

ThetaB = ThirdCarlsonInv\_Binary(new\_integral\_result, k^2\*e^2, k\*e);

%fprintf(1,'==========================================================\n');

%fprintf(1,'%10s %10s %12s %12s\n','Latitude', 'Longitude','Geodesic Distance','ThetaB');

%fprintf(1,'==========================================================\n');

%fprintf(1,'%10s %10s %10s %19.9f\n',answer{1}, answer{2},answer{4}, ThetaB\*180/pi());

%fprintf(1,'----------------------------------------------------------\n');

%fprintf(1,'==========================================================\n');

%fprintf(1,'%10s %10s %12s %15s\n','Characteristic', 'modulus','ThetaB rads','ThetaB degrees');

%fprintf(1,'==========================================================\n');

%fprintf(1,'%10s %10s %10s %19.9f\n',k^2, m^0.5, ThetaB,ThetaB\*180/pi());

%fprintf(1,'----------------------------------------------------------\n');

% difference in longitude equation Rollins (eq 5)

%fprintf(1,'==========================================================\n');

%fprintf(1,'%12s %15s\n','Theta1', 'Theta2');

%fprintf(1,'==========================================================\n');

%fprintf(1,'%10f %10f\n',ThetaA\*180/pi(),ThetaB\*180/pi());

%fprintf(1,'----------------------------------------------------------\n');

Longitude\_Difference = C\*(1-e^2)/(1-C^2\*e^2)^0.5\*...

(ThirdCarlson(k^2,k\*e,ThetaB)-ThirdCarlson(k^2,k\*e,ThetaA));

longitudeB = (Longitude\_Difference + longitudeA);

% keep longitude between -pi() and pi()

if (longitudeB < -pi()) & (longitudeB > -2\*pi())

longitudeB = longitudeB + 2\*pi();

end

if (longitudeB > pi()) & (longitudeB < 2\*pi())

longitudeB = longitudeB - 2\*pi();

end

longitudeBD = 180/pi()\*longitudeB;

%fprintf(1,'==========================================================\n');

%fprintf(1,'%15s %10s %15s %10s\n','Longitude diff', 'Degrees','LongitudeB','Degrees');

%fprintf(1,'==========================================================\n');

%fprintf(1,'%18.15f %8.4f %18.15f %8.4f\n',Longitude\_Difference, Longitude\_Difference\*180/pi(),longitudeB,180/pi()\*longitudeB);

%fprintf(1,'----------------------------------------------------------\n');

latitudeB = asin(k\*sin(ThetaB));

latitudeBD = (180/pi())\*asin(k\*sin(ThetaB));

azimuthA;

azimuthB=asin((C\*(1-e^2\*(sin(latitudeB))^2)^0.5)/cos(latitudeB));

% Does the azimuth pass through -90°?

% The extreme latitudes correspond to theta = pi()/2

% So if thetaA to thetaB passes through -pi()/2 or pi()/2 then an extreme

% latitude has been crossed.

if (ThetaA < pi()/2) & (ThetaB < pi()/2)

%does not pass through pi()/2 so keep quadrant of original azimuth.

% if azimuthA is in the 3rd quadrant and azimuthB is in the 2nd then

% change azimuthB to the 3rd quadrant

if (azimuthA < -pi()/2) & (azimuthA > -pi()) & (azimuthB < 0) & (azimuthB > -pi()/2)

azimuthB=-(pi()+(azimuthB));

end

end

if (ThetaA < pi()/2) & (ThetaB > pi()/2)

%passes through pi()/2 so change quadrant of azimuthB.

% if azimuthA is in the 2nd quadrant and azimuthB is in the 2nd then

% change azimuthB to the 3rd quadrant

if (azimuthA > -pi()/2) & (azimuthA < 0) & (azimuthB < 0) & (azimuthB > -pi()/2)

azimuthB=-(pi()+(azimuthB));

end

end

%end

%if westward & ((azimuthB > 0) & (azimuthB < pi()/2))

% azimuthB = azimuthB + pi()

%end

%if westward & ((azimuthB > pi()/2) & (azimuthB < pi()))

% azimuthB = azimuthB + pi()

%end

azimuthBD = azimuthB\*180/pi();

backazimuth = azimuthB+pi();

backazimuthD = backazimuth\*180/pi();

%fprintf(1,'==========================================================\n');

%fprintf(1,'%15s %19s %25s\n','Next lat(D)', 'Next long(D)','Forward Azimuth(D)');

%fprintf(1,'==========================================================\n');

%fprintf(1,'%18.15f %18.15f %18.15f\n',latitudeBD,longitudeBD,azimuthBD);

%fprintf(1,'----------------------------------------------------------\n');

% return the forward azimuth

NextPoint=[latitudeB longitudeB azimuthB];

end

MATLAB© code for testing the direct problem

function Geodesy\_Tester\_Direct

%GEODESY\_TESTER\_DIRECT Tests the Direct function

%

format long g

prompt = {'Enter latitude A DD.MMSS','Enter longitude A DD.MMSS',...

'Azimuth A','Enter geodesic distance AB','Enter semi-major axis a',...

'enter flat','Run Test?'};

dlg\_title = 'Input phi';

num\_lines = 1;

def = {'-34','-72','-100','1E5','6378137','298.257222101','Yes'};

answer = inputdlg(prompt,dlg\_title,num\_lines,def);

%Change DD.MMSS into whole degrees, minutes, seconds

latitudeA = DecDMStoRad(str2num(answer{1}));

longitudeA = DecDMStoRad(str2num(answer{2}));

azimuthA = DecDMStoRad(str2num(answer{3}));

dist = str2num(answer{4});

a = str2num(answer{5});

flat = str2num(answer{6});

runtest = answer{7};

%Initialize the map

figure

geoshow('landareas.shp', 'FaceColor', [0.5 1.0 0.5]);

%load geoid;

%geoshow(geoid, geoidrefvec, 'DisplayType', 'texturemap');

% Calculate successive geodesics from start and plot

Rlooplat = latitudeA;

Rlooplong = longitudeA;

Rloopazimuth = azimuthA;

Klooplat = (180/pi())\*latitudeA;

Klooplong = (180/pi())\*longitudeA;

Kloopazimuth = (180/pi())\*azimuthA;

s=dist;

geoshow((180/pi())\*latitudeA, (180/pi())\*longitudeA,'DisplayType','point','MarkerEdgeColor','r','Marker', '.');

fprintf(1,'==========================================================\n');

fprintf(1,'%36s, %15.5f %10s\n','Starting Point with distance',s, 'metres');

fprintf(1,'%15s %16s %25s\n','lat(D)', 'long(D)','Forward Azimuth(D)');

fprintf(1,'==========================================================\n');

fprintf(1,'%18.15f %18.15f %18.15f\n',180/pi()\*Rlooplat,180/pi()\*Rlooplong,180/pi()\*Rloopazimuth);

fprintf(1,'----------------------------------------------------------\n');

for i = 1:1000

NextPoint = Direct(Rlooplat, Rlooplong, Rloopazimuth,s,a,flat);

KarneyPoint = geodreckon(Klooplat,Klooplong,s,Kloopazimuth, [a 0.08181919104281579]);

geoshow((180/pi())\*NextPoint(1), (180/pi())\*NextPoint(2),'DisplayType','Line');

fprintf(1,'%2s %18.15f %18.15f %18.15f\n','R',(180/pi())\*NextPoint(1),(180/pi())\*NextPoint(2),(180/pi())\*(NextPoint(3)));

fprintf(1,'%2s %18.15f %18.15f %18.15f\n', 'K',KarneyPoint(1),KarneyPoint(2),KarneyPoint(3));

fprintf(1,'----------------------------------------------------------\n');

% update for next round in loop

Rlooplat = NextPoint(1);

Rlooplong = NextPoint(2);

Rloopazimuth = NextPoint(3);

Klooplat = KarneyPoint(1);

Klooplong = KarneyPoint(2);

Kloopazimuth = KarneyPoint(3);

end

fprintf(1,'==========================================================\n');

end

Results of running the General Elliptic Integrals of the Third Kind program ThirdCarlson and the inbuilt MATLAB© program ellipticPI.

=============================================================================

n k phi(rads) MATLAB© inbuilt ThirdCarlson.m diff at 1E-14

=============================================================================

0.33351 0.08182 -12.56637 -15.421166067881860 -15.421166067881867 0

-----------------------------------------------------------------------

0.33351 0.08182 -12.31250 -15.165463049172731 -15.165463049172740 0

-----------------------------------------------------------------------

0.33351 0.08182 -12.05864 -14.898764950561381 -14.898764950561382 0

-----------------------------------------------------------------------

0.33351 0.08182 -11.80477 -14.610466558370360 -14.610466558370359 0

-----------------------------------------------------------------------

0.33351 0.08182 -11.55091 -14.291992964264590 -14.291992964264592 0

-----------------------------------------------------------------------

0.33351 0.08182 -11.29704 -13.940737834593010 -13.940737834593001 0

-----------------------------------------------------------------------

0.33351 0.08182 -11.04317 -13.565152579991119 -13.565152579991119 0

-----------------------------------------------------------------------

0.33351 0.08182 -10.78931 -13.185168176188380 -13.185168176188387 0

-----------------------------------------------------------------------

0.33351 0.08182 -10.53544 -12.823095840194361 -12.823095840194355 0

-----------------------------------------------------------------------

0.33351 0.08182 -10.28158 -12.492140044039330 -12.492140044039333 0

-----------------------------------------------------------------------

0.33351 0.08182 -10.02771 -12.193322314024730 -12.193322314024730 0

-----------------------------------------------------------------------

0.33351 0.08182 -9.77384 -11.919714295876600 -11.919714295876597 0

-----------------------------------------------------------------------

0.33351 0.08182 -9.51998 -11.661171209094499 -11.661171209094498 0

-----------------------------------------------------------------------

0.33351 0.08182 -9.26611 -11.406759739138989 -11.406759739138987 0

-----------------------------------------------------------------------

0.33351 0.08182 -9.01225 -11.145465417569380 -11.145465417569376 0

-----------------------------------------------------------------------

0.33351 0.08182 -8.75838 -10.866440286135500 -10.866440286135502 0

-----------------------------------------------------------------------

0.33351 0.08182 -8.50451 -10.560022718056191 -10.560022718056187 0

-----------------------------------------------------------------------

0.33351 0.08182 -8.25065 -10.220758743841399 -10.220758743841396 0

-----------------------------------------------------------------------

0.33351 0.08182 -7.99678 -9.852478583258746 -9.852478583258744 0

-----------------------------------------------------------------------

0.33351 0.08182 -7.74292 -9.471367523019080 -9.471367523019080 0

-----------------------------------------------------------------------

0.33351 0.08182 -7.48905 -9.100361469614832 -9.100361469614832 0

-----------------------------------------------------------------------

0.33351 0.08182 -7.23518 -8.757013257643880 -8.757013257643880 0

-----------------------------------------------------------------------

0.33351 0.08182 -6.98132 -8.446651290170172 -8.446651290170172 0

-----------------------------------------------------------------------

0.33351 0.08182 -6.72745 -8.164683685683050 -8.164683685683050 0

-----------------------------------------------------------------------

0.33351 0.08182 -6.47358 -7.901757610621062 -7.901757610621063 0

-----------------------------------------------------------------------

0.33351 0.08182 -6.21972 -7.647087810229798 -7.647087810229798 0

-----------------------------------------------------------------------

0.33351 0.08182 -5.96585 -7.389663263310301 -7.389663263310301 0

-----------------------------------------------------------------------

0.33351 0.08182 -5.71199 -7.118523165615182 -7.118523165615183 0

-----------------------------------------------------------------------

0.33351 0.08182 -5.45812 -6.823338547594692 -6.823338547594693 0

-----------------------------------------------------------------------

0.33351 0.08182 -5.20425 -6.496561420420138 -6.496561420420139 0

-----------------------------------------------------------------------

0.33351 0.08182 -4.95039 -6.137920416518296 -6.137920416518298 0

-----------------------------------------------------------------------

0.33351 0.08182 -4.69652 -5.759051783609740 -5.759051783609741 0

-----------------------------------------------------------------------

0.33351 0.08182 -4.44266 -5.381654152165701 -5.381654152165702 0

-----------------------------------------------------------------------

0.33351 0.08182 -4.18879 -5.026627758917812 -5.026627758917813 0

-----------------------------------------------------------------------

0.33351 0.08182 -3.93492 -4.704017748483885 -4.704017748483885 0

-----------------------------------------------------------------------

0.33351 0.08182 -3.68106 -4.412342277306062 -4.412342277306062 0

-----------------------------------------------------------------------

0.33351 0.08182 -3.42719 -4.143506160919352 -4.143506160919352 0

-----------------------------------------------------------------------

0.33351 0.08182 -3.17333 -3.887028364279793 -3.887028364279793 0

-----------------------------------------------------------------------

0.33351 0.08182 -2.91946 -3.631928034469725 -3.631928034469725 0

-----------------------------------------------------------------------

0.33351 0.08182 -2.66559 -3.367200373295674 -3.367200373295674 0

-----------------------------------------------------------------------

0.33351 0.08182 -2.41173 -3.082139956659957 -3.082139956659957 0

-----------------------------------------------------------------------

0.33351 0.08182 -2.15786 -2.767754426353015 -2.767754426353014 0

-----------------------------------------------------------------------

0.33351 0.08182 -1.90400 -2.420400711742746 -2.420400711742746 0

-----------------------------------------------------------------------

0.33351 0.08182 -1.65013 -2.046952436024400 -2.046952436024400 0

-----------------------------------------------------------------------

0.33351 0.08182 -1.39626 -1.666218146116899 -1.666218146116899 0

-----------------------------------------------------------------------

0.33351 0.08182 -1.14240 -1.300924157977824 -1.300924157977824 0

-----------------------------------------------------------------------

0.33351 0.08182 -0.88853 -0.965798747688247 -0.965798747688247 0

-----------------------------------------------------------------------

0.33351 0.08182 -0.63467 -0.663233200182845 -0.663233200182845 0

-----------------------------------------------------------------------

0.33351 0.08182 -0.38080 -0.386995821767419 -0.386995821767420 0

-----------------------------------------------------------------------

0.33351 0.08182 -0.12693 -0.127162678947248 -0.127162678947248 0

-----------------------------------------------------------------------

0.33351 0.08182 0.12693 0.127162678947248 0.127162678947248 0

-----------------------------------------------------------------------

0.33351 0.08182 0.38080 0.386995821767419 0.386995821767420 0

-----------------------------------------------------------------------

0.33351 0.08182 0.63467 0.663233200182847 0.663233200182848 0

-----------------------------------------------------------------------

0.33351 0.08182 0.88853 0.965798747688247 0.965798747688247 0

-----------------------------------------------------------------------

0.33351 0.08182 1.14240 1.300924157977824 1.300924157977824 0

-----------------------------------------------------------------------

0.33351 0.08182 1.39626 1.666218146116896 1.666218146116896 0

-----------------------------------------------------------------------

0.33351 0.08182 1.65013 2.046952436024397 2.046952436024397 0

-----------------------------------------------------------------------

0.33351 0.08182 1.90400 2.420400711742743 2.420400711742743 0

-----------------------------------------------------------------------

0.33351 0.08182 2.15786 2.767754426353015 2.767754426353014 0

-----------------------------------------------------------------------

0.33351 0.08182 2.41173 3.082139956659957 3.082139956659957 0

-----------------------------------------------------------------------

0.33351 0.08182 2.66559 3.367200373295672 3.367200373295672 0

-----------------------------------------------------------------------

0.33351 0.08182 2.91946 3.631928034469725 3.631928034469725 0

-----------------------------------------------------------------------

0.33351 0.08182 3.17333 3.887028364279793 3.887028364279793 0

-----------------------------------------------------------------------

0.33351 0.08182 3.42719 4.143506160919353 4.143506160919354 0

-----------------------------------------------------------------------

0.33351 0.08182 3.68106 4.412342277306062 4.412342277306062 0

-----------------------------------------------------------------------

0.33351 0.08182 3.93492 4.704017748483885 4.704017748483885 0

-----------------------------------------------------------------------

0.33351 0.08182 4.18879 5.026627758917810 5.026627758917810 0

-----------------------------------------------------------------------

0.33351 0.08182 4.44266 5.381654152165704 5.381654152165704 0

-----------------------------------------------------------------------

0.33351 0.08182 4.69652 5.759051783609740 5.759051783609741 0

-----------------------------------------------------------------------

0.33351 0.08182 4.95039 6.137920416518295 6.137920416518297 0

-----------------------------------------------------------------------

0.33351 0.08182 5.20425 6.496561420420138 6.496561420420138 0

-----------------------------------------------------------------------

0.33351 0.08182 5.45812 6.823338547594694 6.823338547594694 0

-----------------------------------------------------------------------

0.33351 0.08182 5.71199 7.118523165615182 7.118523165615183 0

-----------------------------------------------------------------------

0.33351 0.08182 5.96585 7.389663263310301 7.389663263310301 0

-----------------------------------------------------------------------

0.33351 0.08182 6.21972 7.647087810229801 7.647087810229801 0

-----------------------------------------------------------------------

0.33351 0.08182 6.47358 7.901757610621063 7.901757610621064 0

-----------------------------------------------------------------------

0.33351 0.08182 6.72745 8.164683685683050 8.164683685683050 0

-----------------------------------------------------------------------

0.33351 0.08182 6.98132 8.446651290170170 8.446651290170170 0

-----------------------------------------------------------------------

0.33351 0.08182 7.23518 8.757013257643878 8.757013257643878 0

-----------------------------------------------------------------------

0.33351 0.08182 7.48905 9.100361469614834 9.100361469614834 0

-----------------------------------------------------------------------

0.33351 0.08182 7.74292 9.471367523019080 9.471367523019080 0

-----------------------------------------------------------------------

0.33351 0.08182 7.99678 9.852478583258744 9.852478583258744 0

-----------------------------------------------------------------------

0.33351 0.08182 8.25065 10.220758743841390 10.220758743841394 0

-----------------------------------------------------------------------

0.33351 0.08182 8.50451 10.560022718056191 10.560022718056187 0

-----------------------------------------------------------------------

0.33351 0.08182 8.75838 10.866440286135500 10.866440286135502 0

-----------------------------------------------------------------------

0.33351 0.08182 9.01225 11.145465417569371 11.145465417569373 0

-----------------------------------------------------------------------

0.33351 0.08182 9.26611 11.406759739138989 11.406759739138986 0

-----------------------------------------------------------------------

0.33351 0.08182 9.51998 11.661171209094499 11.661171209094494 0

-----------------------------------------------------------------------

0.33351 0.08182 9.77384 11.919714295876590 11.919714295876593 0

-----------------------------------------------------------------------

0.33351 0.08182 10.02771 12.193322314024730 12.193322314024732 0

-----------------------------------------------------------------------

0.33351 0.08182 10.28158 12.492140044039330 12.492140044039335 0

-----------------------------------------------------------------------

0.33351 0.08182 10.53544 12.823095840194361 12.823095840194355 0

-----------------------------------------------------------------------

0.33351 0.08182 10.78931 13.185168176188380 13.185168176188387 0

-----------------------------------------------------------------------

0.33351 0.08182 11.04317 13.565152579991119 13.565152579991118 0

-----------------------------------------------------------------------

0.33351 0.08182 11.29704 13.940737834593010 13.940737834593005 0

-----------------------------------------------------------------------

0.33351 0.08182 11.55091 14.291992964264590 14.291992964264592 0

-----------------------------------------------------------------------

0.33351 0.08182 11.80477 14.610466558370360 14.610466558370359 0

-----------------------------------------------------------------------

0.33351 0.08182 12.05864 14.898764950561381 14.898764950561381 0

-----------------------------------------------------------------------

0.33351 0.08182 12.31250 15.165463049172731 15.165463049172738 0

-----------------------------------------------------------------------

0.33351 0.08182 12.56637 15.421166067881860 15.421166067881867 0

Arc Length of a Curve

A differential distance on a plane curve is given by Pythagoras’s Theorem:

Dividing through by

Taking the square root of both sides:

Integrating to find the curve distance between and

|  |  |  |
| --- | --- | --- |
|  |  | (46) |

If the curve is defined parametrically then Pythagoras’s Theorem is:

Integrating between and :

|  |  |  |
| --- | --- | --- |
|  |  | (47) |

Arc length given a polar equation

Given polar coordinates

Then the arc length is:

Dividing through by

Arc Length of an Ellipse

Figure 1 shows the geometry of an ellipse with all relevant angles (from (Deakin, Geometric Geodesy Part A, 2010, p. 12))

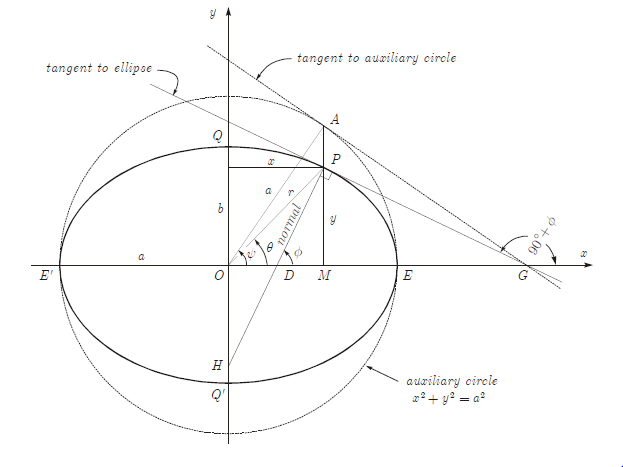


Figure 9

The angle in Figure 1 is the latitude and represents the angle of the normal to the ellipse with the axis. Latitudes of locations on the earth are given as this angle so it is very useful to give results in terms of this angle.

Parametric equations of the ellipse in terms of the latitude parameter

The following derivation follows Deakin (2010 pp13,14).

The Cartesian equation to the ellipse is:

Differentiating with respect to gives:

|  |  |  |
| --- | --- | --- |
|  |  | (48) |

Now is the gradient of the tangent to the ellipse and is given in Figure 9 as:

|  |  |  |
| --- | --- | --- |
|  |  | (49) |

From (48)

From which

|  |  |  |
| --- | --- | --- |
|  |  | (50) |

|  |  |  |
| --- | --- | --- |
|  |  | (51) |

Now the Cartesian equation to the ellipse is:

So substituting (50) into this equation gives:

|  |  |  |
| --- | --- | --- |
|  |  | (52) |

|  |  |  |
| --- | --- | --- |
|  |  | (53) |

Now the eccentricity is defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (54) |

Bringing out of equation (52)

Substituting in equation (54)

Adding together the fraction inside the brackets

|  |  |  |
| --- | --- | --- |
|  |  | (55) |

Which is Deakin (2010) equation (42).

Similarly substituting (51) into the Cartesian equation of the ellipse so as to eliminate :

Substituting in (54)

Adding the fractions in the brackets

|  |  |  |
| --- | --- | --- |
|  |  | (56) |

Now that we have parametric equations for the ellipse in terms of the latitude we can use equation (47) to find arc length as a function of latitude.

|  |  |  |
| --- | --- | --- |
|  |  | (57) |

Arc length as a function of Latitude

Finding the integrand of (57):

Now find:

+ +

Arc length as a function of Latitude using radius of curvature

To find the arc length of an ellipse the easiest way determine the radius of curvature as a function of latitude. From the radius of curvature the arc length is then the integral from one latitude to the other.

General Equation for the Radius of Curvature

Consider Figure 2 (from (Deakin, Geometric Geodesy Part A, 2010, p. 17)) which provides an illustration of tangent and arc length.

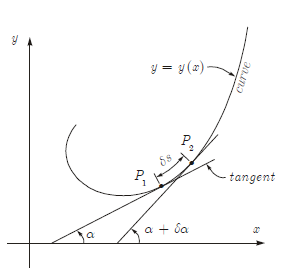


Figure 10

The curvature is the rate of change of the tangent with respect to the arc length.

The radius of curvature is

|  |  |  |
| --- | --- | --- |
|  |  | (58) |

Now from the elemental triangle (Figure 3)

Figure 11

|  |  |  |
| --- | --- | --- |
|  |  | (59) |

Also we know that the tangent to a curve is given by So

Substituting in (59)

Giving

Now from (58)

|  |  |  |
| --- | --- | --- |
|  |  | (60) |

Now from the elemental triangle (Fig. 11):

Dividing through by

Taking the square root of both sides:

|  |  |  |
| --- | --- | --- |
|  |  | (61) |

Also from (59) and (61)

Therefore (60) becomes

|  |  |  |
| --- | --- | --- |
|  |  | (62) |

Which is the general equation for the radius of curvature.

The Radius of Curvature of the Ellipse

Using equation (62) to find the equation to the radius of curvature of the ellipse first note equation (49) above;

|  |  |  |
| --- | --- | --- |
|  |  | (63) |

Equation (56) above gives as a function of

Therefore

The accepted result is which is used in real world situations so the accepted result will be used:

Therefore inverting:

Substituting into (63)

The general equation to the radius of curvature is

We only need the absolute value of the radius of curvature when finding meridian distances and as is always positive and the denominator is always positive and is always less than one then we can write the radius of curvature as:

|  |  |  |
| --- | --- | --- |
|  |  | (64) |

Which agrees with Deakin (2010) equation (66).

Meridian Distance

An infinitesimal distance on the ellipse is given (Deakin, 2010, p60)

Integration between latitudes will then give the meridian distance between the two latitudes

Substituting in (64) we get

|  |  |  |
| --- | --- | --- |
|  |  | (65) |

This is an incomplete integral of the Third Kind defined in the general case by:

|  |  |  |
| --- | --- | --- |
|  |  | (66) |

Note that in the above equation the and the can be different. Equation (65) is the special case in which the and the are the same. In this special case Equation (65) can be transformed into an elliptic integral of the Second Kind minus a constant by the following algebra with this preliminary result:

Note:

So this preliminary result is:

|  |  |  |
| --- | --- | --- |
|  |  | (67) |

Multiplying both sides by

Adding one to and subtracting one from the right hand side:

Rearranging:

Integrating both sides

|  |  |  |
| --- | --- | --- |
|  |  | (68) |

With the right most term evaluated between the limits of integration. Which makes the right hand side:

So we have arrived at a situation where the left hand side is the meridian arc length formula (19) without the factor So multiplying both sides by we get:

|  |  |  |
| --- | --- | --- |
|  |  | (69) |

Now is an elliptic integral of the Second Kind so we can write (22) as (when ):

|  |  |  |
| --- | --- | --- |
|  |  | (70) |

Which is equation (14) in Deakin (2014) and equation (3) in Rösch (2011).

When neither latitude is zero:

|  |  |  |
| --- | --- | --- |
|  |  | (71) |

Where

Equation (71) is useful when the arc length is found between two latitudes neither of which are zero.

Evaluating Elliptic Integrals of the First Kind.

Consider this incomplete integral of the First Kind:

|  |  |  |
| --- | --- | --- |
|  |  | (72) |

Making the substitution (Rösch, 2011, p. 5)

|  |  |  |
| --- | --- | --- |
|  |  | (73) |

then

Now

Now

Therefore

Therefore

|  |  |  |
| --- | --- | --- |
|  |  | (74) |

Which is the same as Rösch gives as his equation (7). The original integral (Eq 1) is

|  |  |  |
| --- | --- | --- |
|  |  | (75) |

Now it is known that:

|  |  |  |
| --- | --- | --- |
|  |  | (76) |

So substituting (73)

into the denominator of the integrand of (75) gives:

Which from (76) gives:

So

|  |  |  |
| --- | --- | --- |
|  |  | (77) |

Which agrees with Rösch equation (8).

Substituting (77) equation (1) gives:

|  |  |  |
| --- | --- | --- |
|  |  | (78) |

Now the double angle formula for cosine is:

|  |  |  |
| --- | --- | --- |
|  |  | (79) |

So (77) becomes after substituting (78):

Bringing out the constants:

Changing the limits of integration to 0 and gives the final result:

This is also an elliptic integral but with a different modulus and amplitude. If we replace the modulus of our original elliptic integral of the First Find (1) with

and change the limits of integration so that

then do the integration and then multiply by

we will arrive at the same value of the integral. This procedure is summarised by:

The modulus q is increasing

As the original modulus is less than unity then the updated modulus q will be more than . The argument is as follows (Rösch, 2011, p. 6):

Due to being less then unity then

So

The modulus converges to unity.

Determining the new amplitude

The new amplitude will be smaller than the old amplitude for an old amplitude between 0 and . The new amplitude is determined (Rösch, 2011, p. 7) from the following formula arising from the change of variable (13).

From the trigonometric identity

|  |  |  |
| --- | --- | --- |
|  |  | (80) |

As the modulus is less than unity then from (79):

For

In which case

|  |  |  |
| --- | --- | --- |
|  |  | (81) |

The chain of transformed elliptic integrals

So we have for the Landen transformation of an incomplete elliptic integral of the First Kind:

with

and

|  |  |  |
| --- | --- | --- |
|  |  | (82) |

In addition we know that for an amplitude in the first quadrant the modulus is increasing and the amplitude is decreasing. Beginning with an initial modulus of and an initial amplitude of then one transformation will bring about this equality:

Repeated transformations will bring about the following equality (Rösch, 2011, p. 7):

|  |  |  |
| --- | --- | --- |
|  |  | (83) |

with:

|  |  |  |
| --- | --- | --- |
|  |  | (84) |

From equation (81) the transformed amplitude is from the previous amplitude . In the terminology of equation (83) is and is . So equation (81) becomes in the new terminology (Rösch, 2011, p. 7)

|  |  |  |
| --- | --- | --- |
|  |  | (85) |

Equations (82) and (84) are Landen’s Ascending Transformation (Deakin, 2014 p13).

The last factor in the product (82) is:

As the limit of the moduli is unity and converges to then this becomes

which from maths tables is:

So the formula for the elliptic integral of the First Kind is:

with

and

Landen’s descending transformation for First Kind integrals

Landen’s ascending transformation is (from (82), (83) and (84)):

|  |  |  |
| --- | --- | --- |
|  |  | (86) |

where

|  |  |  |
| --- | --- | --- |
|  |  | (87) |

and

|  |  |  |
| --- | --- | --- |
|  |  | (88) |

with terminating value :

|  |  |  |
| --- | --- | --- |
|  |  | (89) |

where is the limit of the amplitude sequence.

Landen’s ascending transformation (1) can be rearranged as (Deakin, 2014 p15):

|  |  |  |
| --- | --- | --- |
|  |  | (90) |

Known values of amplitude and modulus are replaced with larger values of amplitude and a smaller modulus. To find the new modulus and amplitude equations (2) and (3) (Deakin, 2014 p15):

|  |  |  |
| --- | --- | --- |
|  |  | (91) |

Which is Deakin (2014) Eqn (66).

To obtain in terms of

Starting with the change of variable defining the Landen transformation (Deakin, 2014 p16):

From the trigonometric identity

So

|  |  |  |
| --- | --- | --- |
|  |  | (92) |

Now

So

Now

and

So

|  |  |  |
| --- | --- | --- |
|  |  | (93) |

From which

|  |  |  |
| --- | --- | --- |
|  |  | (94) |

Equations (90) and (93) give the new moduli and amplitudes. In summary:

|  |  |  |
| --- | --- | --- |
|  |  | (95) |
|  |  | (96) |
|  |  | (97) |

As goes to zero the amplitude will have added to it each time. For example if at one point the amplitude is 20° then the sequence will be approximately:

20°, 40°, 80°, 160°, 320°, 640° …

However as is positive an angle slightly less than will be added to it each time if is in the first quadrant and slightly larger if is in the second quadrant.

The elliptic integral of the First Kind is:

Using Landen’s descending transformation in which goes to zero the integrand becomes in the limit:

Knowing the integral as

Rearranging gives:

Knowing this value can be put in as the terminating value of the recursion:

How to handle the accumulating π’s for the increasing

Evaluating Elliptic Integrals of the Second Kind

An Elliptic Integral of the Second Kind (Eqn 2) is

and this is used in the meridian distance calculation:

a is the semi-major axis

is the first eccentricity

is the latitude

Now the idea is to be able to write this integral of the second kind as a factor times a transformed integral of the second kind. However this is not possible and the best that can be done is to write the integral of the second kind as the sum of a transformed integral of the second kind and a integral of the first kind. This is how it is done.

Adding to both sides of (26) we get (Deakin, 2014 p24):

|  |  |  |
| --- | --- | --- |
|  |  | (98) |

Now we make the substitution as was done with the Elliptic Integral of the First Kind:

Which can be rearranged as:

|  |  |  |
| --- | --- | --- |
|  |  | (99) |

Which agrees with Deakin (2014) Eqn (39).

and from (74) with replacing the

|  |  |  |
| --- | --- | --- |
|  |  | (100) |

Now substituting (99) into (98):

|  |  |  |
| --- | --- | --- |
|  |  | (101) |

Now from the trigonometric identity

(101) becomes

|  |  |  |
| --- | --- | --- |
|  |  | (102) |

To make the change of variable in the integration consider the following diagram:

Figure 12

The change of variable

is shown diagrammatically in Figure 12. If we consider the general case in Figure 13

Figure 13

So

|  |  |  |
| --- | --- | --- |
|  |  | (103) |

Applying this to (102) we obtain using Figure 12:

So far we have

|  |  |  |
| --- | --- | --- |
|  |  | (104) |

which is equation (96) of Deakin (2014).

From the change of variable

Then (74)

Substituting this into (104) and changing the limits of integration gives

|  |  |  |
| --- | --- | --- |
|  |  | (105) |

Now the double angle formula for cosine is:

So (105) becomes

So

|  |  |  |
| --- | --- | --- |
|  |  | (106) |

Where

This last equation comes from the change of variable and is derived as equation (80) above.

Through algebra the right hand side of (106) can be changed into a sum of an elliptic integral of the Second Kind and an elliptic integral of the First Kind. The elliptic integral of the Second Kind will have a modulus which will converge to unity on successive transformations.

This is the algebra (Deakin, 2014 p35). From (106)

|  |  |  |
| --- | --- | --- |
|  |  | (107) |

|  |  |  |
| --- | --- | --- |
|  |  | (108) |

From the trigonometric identity

(108) becomes:

|  |  |  |
| --- | --- | --- |
|  |  | (109) |

Substituting in

(109) becomes

Putting in the left side of the equation which we started with (106):

|  |  |  |
| --- | --- | --- |
|  |  | (110) |

So the integral of an elliptic integral of the Second Kind plus the modulus times the sine of the amplitude is the sum of another elliptic integral of the Second Kind with larger modulus and smaller amplitude and an elliptic integral of the First Kind with the same larger modulus and smaller amplitude. Note the two multiplying factors on the right hand side.

Equation (40) can be written as:

|  |  |  |
| --- | --- | --- |
|  |  | (111) |

with and . The changes in amplitude and modulus are given by:

|  |  |  |
| --- | --- | --- |
|  |  | (112) |

and

|  |  |  |
| --- | --- | --- |
|  |  | (113) |

Now as the modulus converges to unity and the amplitude converges to (Deakin, 2014 p26) the elliptic integral of the Second Kind in (110) will converge to

Abandoned working to find the amplitude given the result of the integral: Second Kind.

Finding the latitude given meridian distance using the elliptic integral of the second kind.

Now the meridian distance formula is:

Inverting Elliptic Integrals of the Second Kind

Now

Where

Note that here is the first Landen descending transformation of the original So in the following equation which has been used above and

(6)

(8)

Eventually will go to which is the amplitude when goes to zero.

From

Using a descending transformation and restricting to the case where the modulus goes to zero with :

=+

=

+

=+

`

Now with

=+

Some simplifying

=+

Now = 2 as

=+

Simplifying

=+

Lets represent this with:

u=a+b+c

Try putting everything in terms of

u=2a+b+c

u=2a+b+c

u-2a-b=c

Dropping the subscript from the phi and the k:

There is no apparent way to solve for so give up this line of attack.

Abandoned: Inverting Elliptic Integrals of the Third Kind using Carlson form and Newton-Raphson

Let the result of the Third Kind integration be . In this case the Carlson form of the Third Kind integral is

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  | . |  |

The Newton-Raphson formula is:

|  |  |  |
| --- | --- | --- |
|  |  | (114) |

Now the derivative of a definite integral with constants as the limit of integration is zero.

Therefore

So the iteration formula (114) becomes:

|  |  |  |
| --- | --- | --- |
|  |  | (115) |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | (116) |

This is implemented in the MATLAB© program ThirdCarlsonInversion(u,n,k).

Now we don’t want the increment to have a denominator of zero. There is no problem about the numerator being zero as this means the solution has been found. So the denominator will be zero when:

This will be zero when:

|  |  |  |
| --- | --- | --- |
|  |  | (117) |

or

|  |  |  |
| --- | --- | --- |
|  |  | (118) |

Equation (117) just means not to invert a value of which corresponds to angle of 90°. Angles close to this will also have problems.

Unfortunately on the first implementation Newton-Raphson failed to converge for angles greater than about 62°. Investigation of why this occurred produced caveats for its use in this instance.

The program gave the following results for and for an arclength of 1.4.

>> ThirdCarlsonInversion

=============================================================================

n k u f(phi) fdash(phi) phi

=============================================================================

Theta0 =

0.785398163397448

0.33351 0.08182 1.40000 -0.560826056718172 0.945768721602425 1.378382520

0.33351 0.08182 1.40000 0.239743587415052 0.420521320086500 0.808272097

0.33351 0.08182 1.40000 -0.533200600338207 0.938854805511126 1.376198679

0.33351 0.08182 1.40000 0.236516047898995 0.424371332903875 0.818865915

0.33351 0.08182 1.40000 -0.520318910932855 0.935431847696388 1.375099822

0.33351 0.08182 1.40000 0.234892525106951 0.426301358112377 0.824098703

0.33351 0.08182 1.40000 -0.513935465476305 0.933687596100724 1.374534917

0.33351 0.08182 1.40000 0.234058029758953 0.427291676546669 0.826763822

0.33351 0.08182 1.40000 -0.510679054465414 0.932785398609753 1.374241361

0.33351 0.08182 1.40000 0.233624413674903 0.427805799898786 0.828142141

0.33351 0.08182 1.40000 -0.508993544130918 0.932315115919937 1.374087965

0.33351 0.08182 1.40000 0.233397839813893 0.428074315292787 0.828860579

0.33351 0.08182 1.40000 -0.508114607877796 0.932068981974702 1.374007577

0.33351 0.08182 1.40000 0.233279106421752 0.428214993328570 0.829236586

0.33351 0.08182 1.40000 -0.507654498336522 0.931939888965804 1.373965387

0.33351 0.08182 1.40000 0.233216791291345 0.428288816273706 0.829433794

I can be seen from the above that the result of phi is oscillating between two values with a starting phi of 0.785.

Running the program Geodesy\_Tester\_ThirdInv.m which gives 25 increments between 0° and 90° gives the following result for the found angle:



Figure 14 Inversion with Newton-Raphson and Carlson

It can be seen that inversion works nicely up to just over 60° and then fails.

This is a printout of the Figure data in which it can be seen that the inversion works nicely up to a value of equal to 1.225.

=============================================================================

n k u ThirdCarlsonInv ThirdCarlson.m diff

=============================================================================

0.33351 0.08182 0.00000 0.000000000000000 0.000000000000000 0

-----------------------------------------------------------------------

0.33351 0.08182 0.05833 0.058311070342834 0.058333333333333 0

-----------------------------------------------------------------------

0.33351 0.08182 0.11667 0.116489171880072 0.116666666666667 0

-----------------------------------------------------------------------

0.33351 0.08182 0.17500 0.174404347966059 0.175000000000000 0

-----------------------------------------------------------------------

0.33351 0.08182 0.23333 0.231932513154970 0.233333333333333 0

-----------------------------------------------------------------------

0.33351 0.08182 0.29167 0.288958020508356 0.291666666666667 0

-----------------------------------------------------------------------

0.33351 0.08182 0.35000 0.345375825401171 0.350000000000000 0

-----------------------------------------------------------------------

0.33351 0.08182 0.40833 0.401093169358374 0.408333333333333 0

-----------------------------------------------------------------------

0.33351 0.08182 0.46667 0.456030746432471 0.466666666666667 0

-----------------------------------------------------------------------

0.33351 0.08182 0.52500 0.510123352512670 0.525000000000000 0

-----------------------------------------------------------------------

0.33351 0.08182 0.58333 0.563320050764462 0.583333333333333 1.110223e-16

-----------------------------------------------------------------------

0.33351 0.08182 0.64167 0.615583911472922 0.641666666666667 1.110223e-16

-----------------------------------------------------------------------

0.33351 0.08182 0.70000 0.666891400724790 0.700000000000000 1.110223e-16

-----------------------------------------------------------------------

0.33351 0.08182 0.75833 0.717231499780747 0.758333333333333 1.110223e-16

-----------------------------------------------------------------------

0.33351 0.08182 0.81667 0.766604636834236 0.816666666666667 0

-----------------------------------------------------------------------

0.33351 0.08182 0.87500 0.815021506886427 0.875000000000000 2.220446e-16

-----------------------------------------------------------------------

0.33351 0.08182 0.93333 0.862501845636301 0.933333333333334 3.330669e-16

-----------------------------------------------------------------------

0.33351 0.08182 0.99167 0.909073211417847 0.991666666666667 0

-----------------------------------------------------------------------

0.33351 0.08182 1.05000 0.954769816827376 1.050000000000000 0

-----------------------------------------------------------------------

0.33351 0.08182 1.10833 0.999631439889792 1.108333333333333 0

-----------------------------------------------------------------------

0.33351 0.08182 1.16667 1.043702434132236 1.166666666666668 8.881784e-16

-----------------------------------------------------------------------

0.33351 0.08182 1.22500 1.087030848140510 1.225000000000001 1.554312e-15

-----------------------------------------------------------------------

0.33351 0.08182 1.28333 1.129667661838155 1.283333338384950 5.051616e-09

-----------------------------------------------------------------------

0.33351 0.08182 1.34167 1.179463100620013 1.352588606754284 1.092194e-02

-----------------------------------------------------------------------

0.33351 0.08182 1.40000 1.373918678966396 1.633147803968215 2.331478e-01

-----------------------------------------------------------------------

The following is a printout of , and the increment .

=============================================================================

n k u f(phi) fdash(phi) f/fdash

=============================================================================

0.33351 0.08182 1.40000 -1.000505461449768 0.996960030608942 -1.003556242

0.33351 0.08182 1.40000 0.266206166765002 0.388289054995122 0.685587614

0.33351 0.08182 1.40000 -0.649336802532510 0.964260169592729 -0.673404153

0.33351 0.08182 1.40000 0.248158090321646 0.410401366769685 0.604671696

0.33351 0.08182 1.40000 -0.568027799152521 0.947477800028837 -0.599515682

0.33351 0.08182 1.40000 0.240532002927115 0.419578184449230 0.573270994

0.33351 0.08182 1.40000 -0.536386640526606 0.939681640467270 -0.570817410

0.33351 0.08182 1.40000 0.236905535629116 0.423907656368856 0.558861186

0.33351 0.08182 1.40000 -0.521859832009041 0.935848102982302 -0.557633050

0.33351 0.08182 1.40000 0.235090940310550 0.426065721583665 0.551771542

0.33351 0.08182 1.40000 -0.514712189665823 0.933901549220870 -0.551141809

0.33351 0.08182 1.40000 0.234160660357245 0.427169944952288 0.548167452

0.33351 0.08182 1.40000 -0.511078646306386 0.932896559435110 -0.547840638

0.33351 0.08182 1.40000 0.233677914108484 0.427742383347377 0.546305260

0.33351 0.08182 1.40000 -0.509201263683966 0.932373194963005 -0.546134602

0.33351 0.08182 1.40000 0.233425841654795 0.428041134631846 0.545335069

0.33351 0.08182 1.40000 -0.508223167928697 0.932099416057859 -0.545245667

0.33351 0.08182 1.40000 0.233293793244001 0.428197593325346 0.544827428

0.33351 0.08182 1.40000 -0.507711393829492 0.931955861297725 -0.544780515

0.33351 0.08182 1.40000 0.233224502926911 0.428279680873053 0.544561214

0.33351 0.08182 1.40000 -0.507443013926384 0.931880496068022 -0.544536575

0.33351 0.08182 1.40000 0.233188111832272 0.428322789820482 0.544421444

0.33351 0.08182 1.40000 -0.507302107083922 0.931840904222218 -0.544408498

0.33351 0.08182 1.40000 0.233168990473182 0.428345440123698 0.544348016

0.33351 0.08182 1.40000 -0.507228081483808 0.931820098239101 -0.544341212

0.33351 0.08182 1.40000 0.233158940891310 0.428357344161649 0.544309428

0.33351 0.08182 1.40000 -0.507189179441536 0.931809162493761 -0.544305851

0.33351 0.08182 1.40000 0.233153658473943 0.428363601279237 0.544289145

0.33351 0.08182 1.40000 -0.507168732105051 0.931803414062400 -0.544287265

0.33351 0.08182 1.40000 0.233150881661211 0.428366890444714 0.544278484

0.33351 0.08182 1.40000 -0.507157983801757 0.931800392220306 -0.544277496

0.33351 0.08182 1.40000 0.233149421920233 0.428368619518914 0.544272879

0.33351 0.08182 1.40000 -0.507152333606059 0.931798803653580 -0.544272360

0.33351 0.08182 1.40000 0.233148654535659 0.428369528490300 0.544269933

0.33351 0.08182 1.40000 -0.507149363323296 0.931797968540886 -0.544269660

0.33351 0.08182 1.40000 0.233148251118315 0.428370006340040 0.544268384

0.33351 0.08182 1.40000 -0.507147801838570 0.931797529517327 -0.544268240

0.33351 0.08182 1.40000 0.233148039039010 0.428370257548864 0.544267570

0.33351 0.08182 1.40000 -0.507146980956731 0.931797298719240 -0.544267494

0.33351 0.08182 1.40000 0.233147927547143 0.428370389611417 0.544267142

0.33351 0.08182 1.40000 -0.507146549412698 0.931797177386657 -0.544267102

0.33351 0.08182 1.40000 0.233147868934851 0.428370459037898 0.544266917

0.33351 0.08182 1.40000 -0.507146322546181 0.931797113600984 -0.544266896

0.33351 0.08182 1.40000 0.233147838121812 0.428370495536057 0.544266798

0.33351 0.08182 1.40000 -0.507146203280328 0.931797080068243 -0.544266787

0.33351 0.08182 1.40000 0.233147821923097 0.428370514723493 0.544266736

0.33351 0.08182 1.40000 -0.507146140581117 0.931797062439752 -0.544266730

0.33351 0.08182 1.40000 0.233147813407274 0.428370524810517 0.544266703

0.33351 0.08182 1.40000 -0.507146107619531 0.931797053172281 -0.544266700

0.33351 0.08182 1.40000 0.233147808930421 0.428370530113365 0.544266686

0.33351 0.08182 1.40000 -0.507146090291297 0.931797048300279 -0.544266685

0.33351 0.08182 1.40000 0.233147806576895 0.428370532901125 0.544266677

0.33351 0.08182 1.40000 -0.507146081181673 0.931797045739019 -0.544266676

0.33351 0.08182 1.40000 0.233147805339624 0.428370534366678 0.544266673

0.33351 0.08182 1.40000 -0.507146076392655 0.931797044392540 -0.544266672

0.33351 0.08182 1.40000 0.233147804689178 0.428370535137133 0.544266670

0.33351 0.08182 1.40000 -0.507146073875021 0.931797043684683 -0.544266670

0.33351 0.08182 1.40000 0.233147804347232 0.428370535542169 0.544266669

0.33351 0.08182 1.40000 -0.507146072551475 0.931797043312555 -0.544266669

0.33351 0.08182 1.40000 0.233147804167467 0.428370535755102 0.544266668

0.33351 0.08182 1.40000 -0.507146071855672 0.931797043116923 -0.544266668

0.33351 0.08182 1.40000 0.233147804072963 0.428370535867042 0.544266668

0.33351 0.08182 1.40000 -0.507146071489881 0.931797043014077 -0.544266668

0.33351 0.08182 1.40000 0.233147804023282 0.428370535925889 0.544266667

0.33351 0.08182 1.40000 -0.507146071297586 0.931797042960012 -0.544266667

0.33351 0.08182 1.40000 0.233147803997164 0.428370535956827 0.544266667

0.33351 0.08182 1.40000 -0.507146071196491 0.931797042931588 -0.544266667

0.33351 0.08182 1.40000 0.233147803983433 0.428370535973090 0.544266667

0.33351 0.08182 1.40000 -0.507146071143346 0.931797042916646 -0.544266667

0.33351 0.08182 1.40000 0.233147803976214 0.428370535981641 0.544266667

0.33351 0.08182 1.40000 -0.507146071115405 0.931797042908789 -0.544266667

0.33351 0.08182 1.40000 0.233147803972420 0.428370535986135 0.544266667

0.33351 0.08182 1.40000 -0.507146071100719 0.931797042904661 -0.544266667

0.33351 0.08182 1.40000 0.233147803970425 0.428370535988499 0.544266667

0.33351 0.08182 1.40000 -0.507146071092995 0.931797042902489 -0.544266667

0.33351 0.08182 1.40000 0.233147803969376 0.428370535989740 0.544266667

0.33351 0.08182 1.40000 -0.507146071088937 0.931797042901348 -0.544266667

0.33351 0.08182 1.40000 0.233147803968825 0.428370535990394 0.544266667

0.33351 0.08182 1.40000 -0.507146071086802 0.931797042900748 -0.544266667

0.33351 0.08182 1.40000 0.233147803968535 0.428370535990738 0.544266667

0.33351 0.08182 1.40000 -0.507146071085679 0.931797042900432 -0.544266667

0.33351 0.08182 1.40000 0.233147803968382 0.428370535990918 0.544266667

0.33351 0.08182 1.40000 -0.507146071085089 0.931797042900266 -0.544266667

0.33351 0.08182 1.40000 0.233147803968303 0.428370535991012 0.544266667

0.33351 0.08182 1.40000 -0.507146071084782 0.931797042900180 -0.544266667

0.33351 0.08182 1.40000 0.233147803968261 0.428370535991062 0.544266667

0.33351 0.08182 1.40000 -0.507146071084620 0.931797042900134 -0.544266667

0.33351 0.08182 1.40000 0.233147803968238 0.428370535991088 0.544266667

0.33351 0.08182 1.40000 -0.507146071084532 0.931797042900109 -0.544266667

0.33351 0.08182 1.40000 0.233147803968227 0.428370535991102 0.544266667

0.33351 0.08182 1.40000 -0.507146071084487 0.931797042900097 -0.544266667

0.33351 0.08182 1.40000 0.233147803968220 0.428370535991110 0.544266667

0.33351 0.08182 1.40000 -0.507146071084461 0.931797042900090 -0.544266667

0.33351 0.08182 1.40000 0.233147803968217 0.428370535991115 0.544266667

0.33351 0.08182 1.40000 -0.507146071084448 0.931797042900086 -0.544266667

0.33351 0.08182 1.40000 0.233147803968216 0.428370535991116 0.544266667

0.33351 0.08182 1.40000 -0.507146071084443 0.931797042900085 -0.544266667

0.33351 0.08182 1.40000 0.233147803968215 0.428370535991117 0.544266667

0.33351 0.08182 1.40000 -0.507146071084441 0.931797042900084 -0.544266667

0.33351 0.08182 1.40000 0.233147803968215 0.428370535991117 0.544266667

It can be seen from the above that Newton-Raphson does settle down but it oscillates between two values of the increment. So it just moves from one value of to the other.

Is this problem solved by starting with a value of that is close the solution? Now with a value of the integral is with the values of as in the table above.

So putting in this value of into the inversion program gives the following table which shows that starting close to the solution does not solve the problem:

>> ThirdCarlsonInversionB

=============================================================================

n k u f(phi) fdash(phi) f/fdash

=============================================================================

0.33351 0.08182 1.39404 -0.994549976507908 0.996960030608942 -0.997582597

0.33351 0.08182 1.39404 0.263307727066542 0.399206393037700 0.659577932

0.33351 0.08182 1.39404 -0.619906514775918 0.959871616578652 -0.645822320

0.33351 0.08182 1.39404 0.242955302387847 0.423795377647336 0.573284456

0.33351 0.08182 1.39404 -0.533342572277920 0.940430416019172 -0.567126034

0.33351 0.08182 1.39404 0.233860254059256 0.434557954359082 0.538156653

0.33351 0.08182 1.39404 -0.498056816633785 0.930912024467212 -0.535020285

0.33351 0.08182 1.39404 0.229232447202340 0.439981259621932 0.521005025

0.33351 0.08182 1.39404 -0.480835102990859 0.925906772652213 -0.519312653

0.33351 0.08182 1.39404 0.226736476819061 0.442891510844075 0.511945863

0.33351 0.08182 1.39404 -0.471743124153772 0.923166315368299 -0.511005565

0.33351 0.08182 1.39404 0.225350049756853 0.444503598289760 0.506970136

0.33351 0.08182 1.39404 -0.466750981267582 0.921632386145014 -0.506439431

0.33351 0.08182 1.39404 0.224567663225772 0.445411925526567 0.504179727

0.33351 0.08182 1.39404 -0.463951930248445 0.920763190703996 -0.503877582

0.33351 0.08182 1.39404 0.224122265358440 0.445928567044249 0.502596788

0.33351 0.08182 1.39404 -0.462364279670645 0.920267244357477 -0.502423924

0.33351 0.08182 1.39404 0.223867455239716 0.446223987921635 0.501693009

0.33351 0.08182 1.39404 -0.461457875800394 0.919983150916442 -0.501593834

0.33351 0.08182 1.39404 0.223721270132065 0.446393423087502 0.501175104

0.33351 0.08182 1.39404 -0.460938487566389 0.919820046782669 -0.501118115

0.33351 0.08182 1.39404 0.223637268849909 0.446490768411533 0.500877699

0.33351 0.08182 1.39404 -0.460640238756144 0.919726284287024 -0.500844922

0.33351 0.08182 1.39404 0.223588955402213 0.446546751423247 0.500706711

0.33351 0.08182 1.39404 -0.460468767778955 0.919672343744862 -0.500687849

0.33351 0.08182 1.39404 0.223561153176714 0.446578965391841 0.500608337

0.33351 0.08182 1.39404 -0.460370116153010 0.919641299106494 -0.500597479

0.33351 0.08182 1.39404 0.223545149380036 0.446597508141536 0.500551717

0.33351 0.08182 1.39404 -0.460313336678669 0.919623427465188 -0.500545466

0.33351 0.08182 1.39404 0.223535935503158 0.446608183579717 0.500519121

0.33351 0.08182 1.39404 -0.460280649434065 0.919613137744266 -0.500515522

0.33351 0.08182 1.39404 0.223530630257706 0.446614330311928 0.500500354

0.33351 0.08182 1.39404 -0.460261829299843 0.919607212884755 -0.500498281

0.33351 0.08182 1.39404 0.223527575380793 0.446617869714319 0.500489547

0.33351 0.08182 1.39404 -0.460250992521290 0.919603801168966 -0.500488354

0.33351 0.08182 1.39404 0.223525816256867 0.446619907840977 0.500483324

0.33351 0.08182 1.39404 -0.460244752346898 0.919601836545450 -0.500482637

0.33351 0.08182 1.39404 0.223524803261412 0.446621081498460 0.500479741

0.33351 0.08182 1.39404 -0.460241158957992 0.919600705206936 -0.500479345

0.33351 0.08182 1.39404 0.223524219919260 0.446621757358457 0.500477678

0.33351 0.08182 1.39404 -0.460239089683841 0.919600053714036 -0.500477450

0.33351 0.08182 1.39404 0.223523883994520 0.446622146560477 0.500476489

0.33351 0.08182 1.39404 -0.460237898070162 0.919599678543231 -0.500476358

0.33351 0.08182 1.39404 0.223523690547425 0.446622370687977 0.500475805

0.33351 0.08182 1.39404 -0.460237211863355 0.919599462495516 -0.500475730

0.33351 0.08182 1.39404 0.223523579147898 0.446622499755264 0.500475411

0.33351 0.08182 1.39404 -0.460236816700820 0.919599338081003 -0.500475368

0.33351 0.08182 1.39404 0.223523514996663 0.446622574080758 0.500475184

0.33351 0.08182 1.39404 -0.460236589140145 0.919599266434852 -0.500475159

0.33351 0.08182 1.39404 0.223523478054105 0.446622616882331 0.500475053

0.33351 0.08182 1.39404 -0.460236458095574 0.919599225176214 -0.500475039

0.33351 0.08182 1.39404 0.223523456780105 0.446622641530342 0.500474978

0.33351 0.08182 1.39404 -0.460236382631345 0.919599201416724 -0.500474970

0.33351 0.08182 1.39404 0.223523444529109 0.446622655724320 0.500474935

0.33351 0.08182 1.39404 -0.460236339173986 0.919599187734415 -0.500474930

0.33351 0.08182 1.39404 0.223523437474164 0.446622663898164 0.500474910

0.33351 0.08182 1.39404 -0.460236314148324 0.919599179855224 -0.500474907

0.33351 0.08182 1.39404 0.223523433411452 0.446622668605213 0.500474896

0.33351 0.08182 1.39404 -0.460236299736868 0.919599175317856 -0.500474894

0.33351 0.08182 1.39404 0.223523431071870 0.446622671315848 0.500474887

0.33351 0.08182 1.39404 -0.460236291437782 0.919599172704936 -0.500474886

0.33351 0.08182 1.39404 0.223523429724580 0.446622672876816 0.500474882

0.33351 0.08182 1.39404 -0.460236286658606 0.919599171200237 -0.500474882

0.33351 0.08182 1.39404 0.223523428948721 0.446622673775724 0.500474880

0.33351 0.08182 1.39404 -0.460236283906439 0.919599170333733 -0.500474879

0.33351 0.08182 1.39404 0.223523428501928 0.446622674293377 0.500474878

0.33351 0.08182 1.39404 -0.460236282321553 0.919599169834740 -0.500474878

0.33351 0.08182 1.39404 0.223523428244636 0.446622674591476 0.500474877

0.33351 0.08182 1.39404 -0.460236281408873 0.919599169547387 -0.500474877

0.33351 0.08182 1.39404 0.223523428096469 0.446622674763140 0.500474877

0.33351 0.08182 1.39404 -0.460236280883290 0.919599169381911 -0.500474877

0.33351 0.08182 1.39404 0.223523428011144 0.446622674861998 0.500474876

0.33351 0.08182 1.39404 -0.460236280580621 0.919599169286618 -0.500474876

0.33351 0.08182 1.39404 0.223523427962008 0.446622674918927 0.500474876

0.33351 0.08182 1.39404 -0.460236280406322 0.919599169231740 -0.500474876

0.33351 0.08182 1.39404 0.223523427933712 0.446622674951711 0.500474876

0.33351 0.08182 1.39404 -0.460236280305950 0.919599169200138 -0.500474876

0.33351 0.08182 1.39404 0.223523427917419 0.446622674970588 0.500474876

0.33351 0.08182 1.39404 -0.460236280248154 0.919599169181941 -0.500474876

0.33351 0.08182 1.39404 0.223523427908037 0.446622674981459 0.500474876

0.33351 0.08182 1.39404 -0.460236280214872 0.919599169171463 -0.500474876

0.33351 0.08182 1.39404 0.223523427902633 0.446622674987718 0.500474876

0.33351 0.08182 1.39404 -0.460236280195705 0.919599169165428 -0.500474876

0.33351 0.08182 1.39404 0.223523427899522 0.446622674991324 0.500474876

0.33351 0.08182 1.39404 -0.460236280184668 0.919599169161953 -0.500474876

0.33351 0.08182 1.39404 0.223523427897730 0.446622674993400 0.500474876

0.33351 0.08182 1.39404 -0.460236280178312 0.919599169159953 -0.500474876

0.33351 0.08182 1.39404 0.223523427896697 0.446622674994597 0.500474876

0.33351 0.08182 1.39404 -0.460236280174646 0.919599169158798 -0.500474876

0.33351 0.08182 1.39404 0.223523427896103 0.446622674995285 0.500474876

0.33351 0.08182 1.39404 -0.460236280172541 0.919599169158135 -0.500474876

0.33351 0.08182 1.39404 0.223523427895762 0.446622674995681 0.500474876

0.33351 0.08182 1.39404 -0.460236280171328 0.919599169157754 -0.500474876

0.33351 0.08182 1.39404 0.223523427895563 0.446622674995911 0.500474876

0.33351 0.08182 1.39404 -0.460236280170625 0.919599169157532 -0.500474876

0.33351 0.08182 1.39404 0.223523427895450 0.446622674996042 0.500474876

0.33351 0.08182 1.39404 -0.460236280170223 0.919599169157405 -0.500474876

0.33351 0.08182 1.39404 0.223523427895385 0.446622674996117 0.500474876

0.33351 0.08182 1.39404 -0.460236280169992 0.919599169157333 -0.500474876

0.33351 0.08182 1.39404 0.223523427895347 0.446622674996161 0.500474876

This is a plot of the angles from (angle ) to (angle + increment). The angles are oscillating between and . The answer provided is then one of the angles at either end.



Something like the problem is shown in the following graph which shows the gradients at each end as dashed lines. The gradients are following the function curve. This is not exactly clear but it is sort of showing this.



To attempt to solve this I doubled the gradient.

Abandoned: Inverting Elliptic Integrals of the Third Kind using Newton-Raphson

This is the MATLAB© code which implemented Newton-Raphson in the program Direct.m but was subsequently replaced with the slower but simpler binary method. The theory follows after the program.

if NR == 'Y'

% Newton-Raphson (Note: not fully tested may have times it falls over such

% as when the derivative goes to zero.

% Option of Binary search follows in else clause.

% x1=x0-f(x0)/f'(x0)

%

% (?(s?\_B-s\_A)(1-?k^2 e?^2)?(1-C^2 e^2 ))/((1-e^2))

RHSformula23FirstTerm = distance\*(1-k^2\*e^2)\*(1-C^2\*e^2)^0.5/(1-e^2);

RHSformula23SecondTerm = a\*(E\_Rec(ThetaA,abs(k\*e))-...

k^2\*e^2\*(sin(ThetaA)\*cos(ThetaA))/((1-(k\*e\*sin(ThetaA))^2)^0.5));

% Dividing through by the 'a' on the left of equation (23)

% leaves the Second Kind integral take away the sin cos bit

SecondKind\_Plus\_sincosbit = (RHSformula23FirstTerm+RHSformula23SecondTerm)/a;

Result = SecondKind\_Plus\_sincosbit;

Theta0 = pi()/4;

diff = 1;

while diff > 1E-15

fx0=E\_Rec(Theta0,abs(k\*e))-...

k^2\*e^2\*(sin(Theta0)\*cos(Theta0))/(1-(k\*e\*sin(Theta0))^2)^0.5-...

Result;

fdashx0=(1-(k\*e\*sin(Theta0))^2)^0.5-...

((k\*e)^2/((1-(k\*e\*sin(Theta0))^2)^1.5))\*...

((1-(k\*e\*sin(Theta0))^2)+(sin(Theta0)\*cos(Theta0))^2);

Theta1 = Theta0-fx0/fdashx0;

diff = abs(Theta1-Theta0);

Theta0 = Theta1;

end

ThetaB = Theta0;

% End of Newton-Raphson.

The following method converts the Third Kind integral into the sum of a First Kind and a Second Kind integration (see Appendix O). Then it finds the derivate in preparation for using Newton-Raphson to determine . The Newton-Raphson formula is:

|  |  |  |
| --- | --- | --- |
|  |  | (119) |

Beginning with the geodesic length formula:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | (120) |

Taking the Right Hand Side of Equation (37):

=

()

The second term is known in the direct problem. So the entire equation is:

()

As we will find this can be simplified by substituting into the known values of .:

Now the derivative of will be:

|  |  |  |
| --- | --- | --- |
|  |  | (121) |

|  |  |  |
| --- | --- | --- |
|  |  | (122) |

Now

Abandoned working to evaluate Elliptic Integrals of the Third Kind using a Landen transformation

An elliptic integral of the Third Kind is:

(1)

For the purposes of finding meridian distances the special case of this integral is when

So equation (1) becomes:

(2)

Applying the change of variable of the Landen transformation:

Or as previously derived (see second kind above (30)):

(2) becomes

(3)

(4)

Now the denominator of the integrand can be expanded as:

+

From the diagram:

+

+

+

+)

+)

(5)

So equation (4), the integral of the Third Kind is:

(6)

Also

(31)

So (6) becomes:

(7)

(8)

The double angle formula is:

(9)

Try 1

Try 2

Try 3 making denom into a third kind

Try 4 Factorise denom

Try 5 make the top look like the bottom

Now the original integral of the Third Kind is before the change of variable:

Khan’s endpoint formula

My program M\_Rec was used to test the precision of Khan’s endpoint formula. The formula was found to be only 5% precise.

Khan’s endpoint formula is:

where and are the endpoints of the ellipse. This is given as approximation one in Weintrit (2013 p262).

Figure 6 shows the geometry of an ellipse with all relevant angles (from (Deakin, Geometric Geodesy Part A, 2010, p. 12))

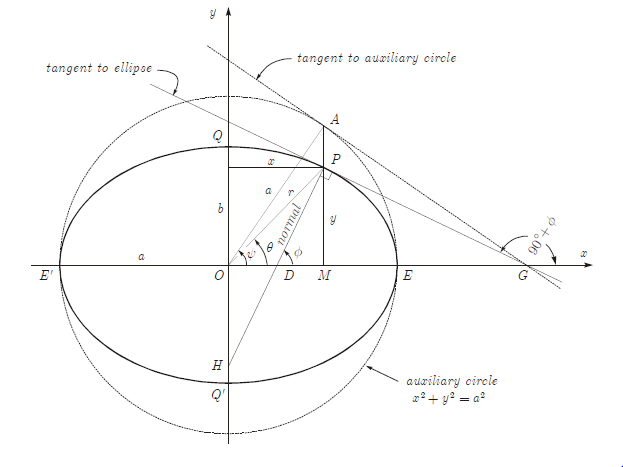


Figure 15

Now (Deakin p14):

Substituting this into Khan’s formula:

When then

|  |  |  |
| --- | --- | --- |
|  |  | (123) |

The integral formula for meridian distance is (Eqn ):

|  |  |  |
| --- | --- | --- |
|  |  | (124) |

Putting this into a MATLAB© program gives the following results with the same test inputs as M\_Rec:

>> M\_Khan

==========================================================

Latitude Semi-major flat Meridian distance

==========================================================

60.0000 6378137 298.257222101 7057598.202121996

----------------------------------------------------------

30.000000 Bessel -- 3645337.831611826

45.000000 Bessel -- 5395032.688236587

60.000000 Bessel -- 7056806.327400127

90.000000 Bessel -- 10000812.580836166

My previous program using the Landen transformation and checked with Dorrer gave the following results (above in section 8 and repeated here)

>> M\_Rec

==========================================================

Latitude Semi-major flat Meridian distance

==========================================================

60.0000 6378137 298.257222101 6654072.819367446

----------------------------------------------------------

30.000000 Bessel -- 3319786.509543302

45.000000 Bessel -- 4984439.265470861

60.000000 Bessel -- 6653376.120611615

90.000000 Bessel -- 10000855.764435511

Conclusion: Khan’s formula is not precise enough for geodetic calculations.

MATLAB code to test Khan’s formula

function M\_Khan

% M\_Khan This function gives meridian distance by the Khan formula

format long g

% This creates a window for the user to enter latitude,

% semi-major axis and flat

prompt = {'Enter latitude DD.MMSS','Enter semi-major axis a','enter flat','Run Test?'};

dlg\_title = 'Input phi';

num\_lines = 1;

def = {'60.0000','6378137','298.257222101','Yes'};

answer = inputdlg(prompt,dlg\_title,num\_lines,def);

phi0 = abs(str2num(answer{1}));

a = str2num(answer{2});

flat = str2num(answer{3});

runtest = answer{4};

%convert input latitude to radians

%Change DD.MMSS into whole degrees, minutes, seconds

phi0deg = fix(phi0);

mins\_secs = phi0 - phi0deg;

phi0minutes = abs(fix(mins\_secs\*100));

phi0seconds = abs(fix((phi0 - phi0deg - phi0minutes/100)\*10000));

phi0r = (phi0deg+(phi0minutes/60)+(phi0seconds/3600))\*(pi()/180);

% convert flat into flattening and find first eccentricity

f=1/flat;

e = (f\*(2-f))^0.5;

b=a\*(1-f);

% find meridian distance by using Khan's formula

xcoord=a\*cos(phi0r)/(1-e^2\*(sin(phi0r))^2)^0.5;

ycoord=(b\*(1-e^2)^0.5\*sin(phi0r))/(1-e^2\*(sin(phi0r))^2)^0.5;

M=(pi()/(2\*2^0.5))\*((xcoord-a)^2+ycoord^2)^0.5;

fprintf(1,'==========================================================\n');

fprintf(1,'%12s %12s %5s %26s\n','Latitude', 'Semi-major', 'flat','Meridian distance');

fprintf(1,'==========================================================\n');

fprintf(1,'%10s %10s %16s %19.9f\n',answer{1}, answer{2},answer{3}, M);

fprintf(1,'----------------------------------------------------------\n');

if runtest == 'Yes'

lats = (pi()/180)\*[30,45,60,90];

a=6377397.155; %Bessel's ellipsoid

e = 0.08169683121517; %Bessel's ellipsoid

b=a\*(1-f); %Bessel's ellipsoid

for n = 1:4

xcoord=a\*cos(lats(n))/(1-e^2\*(sin(lats(n)))^2)^0.5;

ycoord=(b\*(1-e^2)^0.5\*sin(lats(n)))/(1-e^2\*(sin(lats(n)))^2)^0.5;

M=(pi()/(2\*2^0.5))\*((xcoord-a)^2+ycoord^2)^0.5;

fprintf(1,'%10f %10s %16s %19.9f\n',180/pi()\*lats(n), 'Bessel','--', M)

end

end

end

MATLAB code to change DD.MMSS into decimal degrees

function phi0r = DecDMStoRad( phi )

%DECDMSTORAD Decimal Degrees mins secs to radians

% This converts an input of DD.MMSS into radians

phi0=abs(phi);

phi0deg = fix(phi0);

mins\_secs = phi0 - phi0deg;

phi0minutes = abs(fix(mins\_secs\*100));

phi0seconds = abs(fix((phi0 - phi0deg - phi0minutes/100)\*10000));

phi0r = (phi0deg+(phi0minutes/60)+(phi0seconds/3600))\*(pi()/180);

if phi < 0

phi0r = -1\*phi0r;

end

end