GEODESIC ALGORITHMS: AN EXPERIMENTAL STUDY

Vissarion Fisikopoulos[1]

[1] Oracle, Greece - vissarion.fysikopoulos@oracle.com

Commission IV, WG IV/4

KEY WORDS: geodesics, distance computation, numerical methods, experiments

ABSTRACT:

The figure of the Earth can be modelled either by a cartesian plane, a sphere or an (oblate) ellipsoid, in decreasing order with respect to the approximation quality. Based on those models, we experimentally study the accuracy-performance trade-offs of various methods for some basic geodesic problems. For our experiments we use the open source libraries `Boost Geometry` and `GeographicLib`. Our results can be used as a reference for practitioners that want to use the most efficient method with respect to some given accuracy. Geodesic computations are building blocks for many higher level algorithms such as k-nearest neighbour problems, line interpolation, area and buffer, to name a few.

1. INTRODUCTION

The figure of the Earth can be modelled either by a cartesian plane, a sphere or an (oblate) ellipsoid, in decreasing order with respect to the approximation quality. The shortest path between two points on such a surface is called a geodesic. Studying geodesic problems on ellipsoids dates back to Newton. However, many open-source GIS systems today use spherical methods or even methods on cartesian plane. The main advantages of those approaches are simplicity of implementation and performance. On the other hand, those approaches come with a handicap: accuracy.

In this report we experimentally study the accuracy-performance trade-offs of various methods for distance computation (as well as similar geodesic problems such as azimuth and area computation). We test projections paired with cartesian computations, spherical-trigonometric computations and a number of ellipsoidal methods (Thomas, 1965), (Thomas, 1970) formulas, (Vincenty, 1975) iterative method, great elliptic arc's method, and (Karney, 2013) series approximation. For our experiments we use the open source libraries Boost Geometry (Gehrels et al., 2019) and GeographicLib (Karney, 2017).

Our results are of independent interest and can be used as a reference for practitioners that want to use the most efficient method with respect to some given accuracy.

Geodesic computations (such as distance computations) apart from being a fundamental problem in computational geometry and geography/geodesy are also building blocks for many higher level algorithms such as k-nearest neighbour problems, line interpolation, densification of geometries, area and buffer, to name a few. In section 4 we provide some examples for area and distance between a point and a segment.

2. GEODESIC ALGORITHMS

There are two main problems in geodesics:

- Direct problem: given ϕ_1, s_{12}, α_1, compute $\phi_2, \lambda_{12}, \alpha_2$.
- Inverse problem: given $\phi_1, \phi_2, \lambda_{12}$, compute $s_{12}, \alpha_1, \alpha_2$.

We denote $\lambda,\ \phi, \alpha, s$ the longitude, latitude, azimuth and distance on ellipsoid respectively; λ_1 is the longitude of first point, s_{12} the distance between the first and the second point etc.

We consider the following methods for approximating the solutions of the geodesic problems.

1. Flat earth approximation (Williams, 2012)
2. Spherical approximation (haversine formula)
3. Andoyer-Lambert first order solutions (Thomas, 1965)
4. (Andoyer-Lambert)-Thomas second order solutions (Thomas, 1970)
5. Vincenty's iterative method (Vincenty, 1975)
6. Karney's iterative method (Karney, 2013)

Apart from the methods above there is a variety of methods based on numerical integration e.g. (Sjöberg , Shirazian, 2012), (Panou et al., 2013) which are not considered in this report.

We also test great elliptic arc methods[1]. It was found to have both performance and accuracy between the corresponding performance and accuracy of method 3 and 4 and thus the details are not included here.

3. BENCHMARKS

We run experiments to evaluate the accuracy and performance of the methods under consideration. We use the dataset (Karney, 2010). It is computed with `GeographicLib` (Karney, 2017) using high precision arithmetic for the WGS84 ellipsoid; *i.e.* the ellipsoid of revolution with equatorial radius $a = 6378137m$ and flattening $f = 1/298.257223563$.

Experiments run on an `Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz` with `gcc 7.4`. We test methods 1 - 5 as in `Boost.Geometry` (Gehrels et al., 2019) and method 6 from `GeographicLib` (Karney, 2017). In all figures both axes are in logarithmic scale.

[1] See `https://github.com/boostorg/geometry/pull/431` for implementation details and benchmarks.

method	2	10	100	1000	2000	1000000	2000000	5000000
flat (1)	1.63E-02	1.10E-01	1.33E+00	2.33E+02	4.27E+01	4.80E+05	1.64E+06	6.15E+06
spherical (2)	8.57E-03	4.77E-02	5.54E-01	5.51E+00	6.16E+00	5.43E+03	1.08E+04	2.52E+04
andoyer (3)	6.36E-02	2.90E-03	1.19E-03	1.24E-02	1.40E-02	1.14E+01	2.40E+01	5.16E+01
thomas (3)	1.74E-01	1.02E-03	2.09E-04	2.25E-05	5.04E-06	2.76E-03	5.53E-03	1.34E-02
vincenty (5)	4.25E-06	3.95E-06	3.59E-06	6.31E-06	4.33E-06	6.93E-06	1.43E-05	3.41E-05
GeographicLib; order 3 (6)	2.30E-09	2.33E-09	2.83E-09	3.73E-09	3.78E-09	1.31E-06	2.49E-06	4.12E-06
GeographicLib; order 4 (6)	2.30E-09	2.34E-09	2.80E-09	3.40E-09	3.57E-09	2.68E-09	4.89E-09	1.12E-08
GeographicLib; order 5 (6)	2.30E-09	2.34E-09	2.80E-09	3.40E-09	3.57E-09	2.68E-09	2.33E-09	3.73E-09
GeographicLib; order 6 (6)	2.30E-09	2.34E-09	2.80E-09	3.40E-09	3.57E-09	2.68E-09	2.33E-09	3.73E-09

method	7500000	10000000	12500000	15000000	17500000	19500000	20000000
flat (1)	1.09E+07	1.18E+07	1.18E+07	9.21E+06	5.91E+06	3.35E+06	3.36E+06
spherical (2)	3.37E+04	3.76E+04	3.79E+04	3.65E+04	3.04E+04	2.13E+04	2.16E+04
andoyer (3)	6.31E+01	6.78E+01	6.84E+01	6.80E+01	1.30E+02	8.12E+02	3.09E+03
thomas (4)	1.95E-02	2.99E-02	9.23E-02	3.14E-01	1.72E+00	4.87E+01	9.28E+02
vincenty (5)	4.84E-05	5.98E-05	6.80E-05	7.79E-05	7.87E-05	7.89E-05	1.06E+03
GeographicLib; order 3 (6)	4.41E-06	4.72E-06	5.81E-06	6.86E-06	7.38E-06	7.44E-06	7.39E-06
GeographicLib; order 4 (6)	1.40E-08	1.68E-08	1.86E-08	2.05E-08	2.24E-08	2.24E-08	2.24E-08
GeographicLib; order 5 (6)	4.66E-09	5.59E-09	5.59E-09	7.45E-09	7.45E-09	7.45E-09	3.73E-09
GeographicLib; order 6 (6)	4.66E-09	5.59E-09	5.59E-09	7.45E-09	7.45E-09	7.45E-09	3.73E-09

Table 1. Maximum absolute error for distance (m); columns contain geodesices of length less than the value descibed by the column label, e.g. second column contains geodesics of legth l s.t. $2m < l \leq 10m$.

We compute the maximum absolute error for distance (in meters) and azimuth (in degrees). All computations are performed with double arithmetic. More details about computations are in this wiki[2].

Regarding performance Table 2 shows that method 3 is around $2\times$ faster than method 4 which is around $2\times$ faster than method 5, which is around $3\times$ faster than method 6.

For testing accuracy we use a subset of the dataset (Karney, 2010) that contains

1. 100000 geodesics uniform distributed in the ellipsoid and
2. 50000 short geodesics (of length less than 10 km).

Figure 1 and Table 1 show that for long geodesics (case 1 above) the faster the method is the lower the accuracy obtained. For geodesics with endpoints far from antipodal (geodesics of length less than $19500km$) methods 2-6 have accuracy $21km$, $811m$, $48m$, $78\mu m$, $7nm$ respectively. Method 1 has very low accuracy and could be only useful for very short distance and only in specific areas on the ellipsoid, *e.g.* for the distance between point$(10, 10)$ and point$(10.1, 10.002)$ it obtains cm accuracy.

The only method that is accurate (nm accuracy) for geodesics that has endpoints nearly antipodal is method 6. For that method we also illustrate the accuracy obtained for different orders of series approximation (order 3,4,5,6). Even for order 3 that method is more accurate than method 5. When it is not explicitly referred in the text the order used is 6.

Figure 1 shows similar results for azimuth computation with the main difference be the case of short geodesics where the accuracy of all methods is considerably lower than the case foe distance computation.

4. APPLICATIONS

The two main geodesic problems are the core procedures for many other geodesic computations.

[2] https://github.com/vissarion/geometry/wiki/Geographic-algorithms

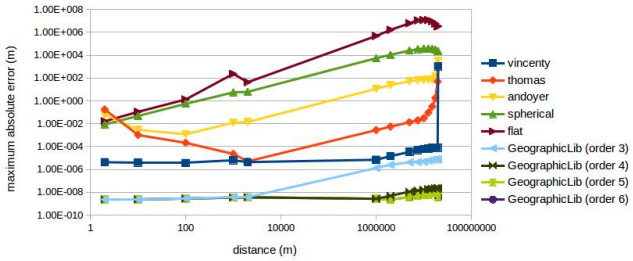

Figure 1. The maximum absolute distance error as a function of distance.

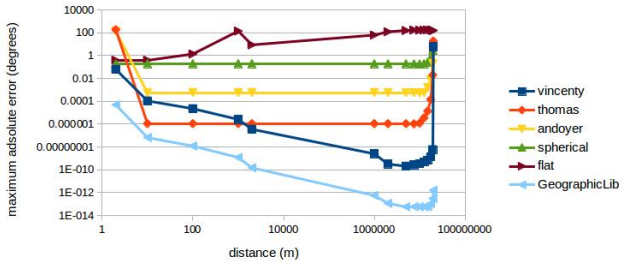

Figure 2. The maximum absolute azimuth error as a function of distance.

4.1 Area of a polygon

We test the accuracy of area computation using the methods 3 - 6 as core procedures. The algorithm for area computation is described in (Karney, 2013). We also consider the following projections followed by a cartesian area computation.

aea Albers Equal Area
https://proj4.org/operations/projections/aea.html)

cea Equal Area Cylindrical
https://proj4.org/operations/projections/cea.html

leac Lambert Equal Area Conic
https://proj4.org/operations/projections/leac.html

method	distance time	azimuth time
flat (1)	7	35
spherical (1)	15	27
andoyer (3)	117	260
thomas (4)	270	371
vincenty (5)	485	463
GeographicLib (6)	1224	1228

Table 2. Average timings for 150000 geodesics in `nsec` (100 runs for each geodesic).

laea Lambert Azimuthal Equal Area
https://proj4.org/operations/projections/laea.html

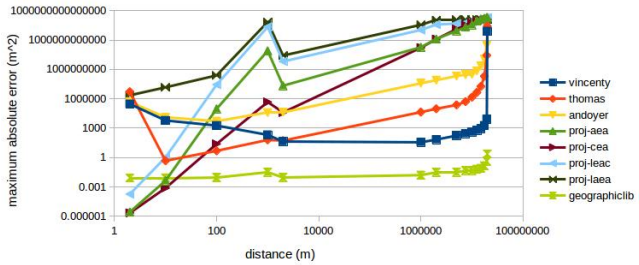

Figure 3. The maximum absolute area error as a function of distance

4.2 Point-segment distance

The main two procedures for computing the distance between a point and a segment on an ellipsoid are the solutions of the direct and the inverse geodesic problems. Below we illustrate how the choice of the solution method affects the result by computing the shortest distance between the segment Sarajevo (43.8563° N, 18.4131° E) - Bucharest (44.4268° N, 26.1025° E) and Athens (37.9838° N, 23.7275° E).

method	distance (m)
spherical	700524.9704
andoyer	699703.5137
thomas	699696.7728
vincenty	699696.7746

REFERENCES

Gehrels, B., Lalande, B., Loskot, M., Wulkiewicz, A., Karavelas, M., Fisikopoulos, V., 2019. Boost C++ libraries: Geometry, version 1.70. http://boost.org/libs/geometry.

Karney, C.F.F., 2013. Algorithms for geodesics. *Journal of Geodesy*, 87, 43–55. https://doi.org/10.1007/s00190-012-0578-z.

Karney, C.F.F., 2017. Geographiclib, version 1.49. https://geographiclib.sourceforge.io/1.49.

Karney, Charles F. F., 2010. Test set for geodesics. https://doi.org/10.5281/zenodo.32156.

Panou, G., Delikaraoglou, D., Korakitis, R., 2013. Solving the geodesics on the ellipsoid as a boundary value problem. *Journal of Geodetic Science*, 3, 40-47. https://doi.org/10.2478/jogs-2013-0007.

Sjöberg, L.E., Shirazian, M., 2012. Solving the Direct and Inverse Geodetic Problems on the Ellipsoid by Numerical Integration. *Journal of Surveying Engineering*, 138, 9-16. https://doi.org/10.1061/(ASCE)SU.1943-5428.0000061.

Thomas, P.D., 1965. Mathematical Models for Navigation Systems. Technical report, US Naval Oceanographic Office.

Thomas, P.D., 1970. *Spheroidal Geodesics, Reference Systems and Local Geometry*. US Naval Oceanographic Office.

Vincenty, T., 1975. "Direct and Inverse Solutions of Geodesics on the Ellipsoid with application of nested equations. *Survey Review*, 23, 88-93. https://doi.org/10.1179/sre.1975.23.176.88.

Williams, E., 2012. Aviation formulary v1.46. Technical report. https://edwilliams.org/avform.htm.