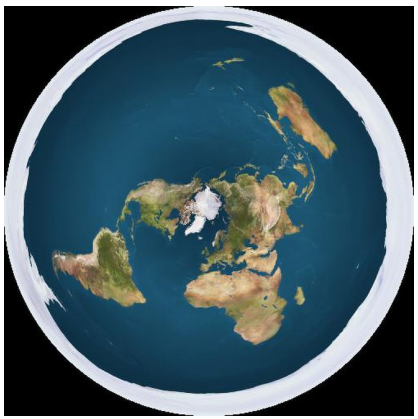
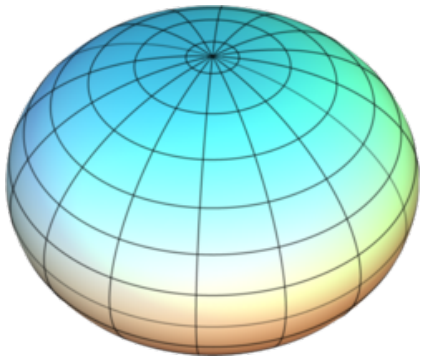
Geodesic algorithms: an experimental study

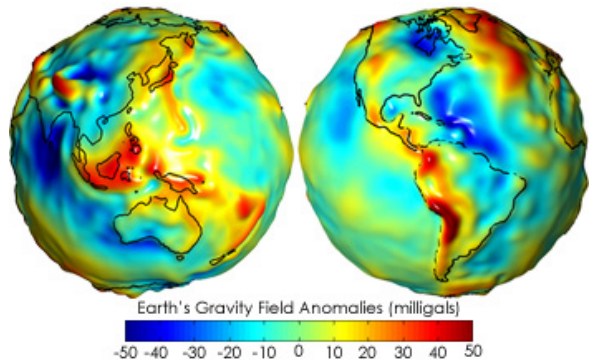
Vissarion Fisikopoulos

Models of the Earth and coordinate systems

- Flat (*local approximation*)
- Sphere (*Widely used e.g.* `google.maps`)
- Ellipsoid of revolution (*geographic GIS state-of-the-art*)
- Geoid (*Special applications, geophysics etc*)



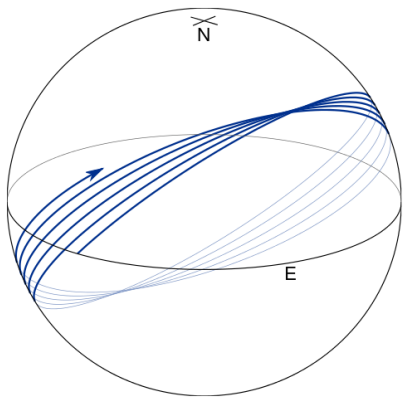


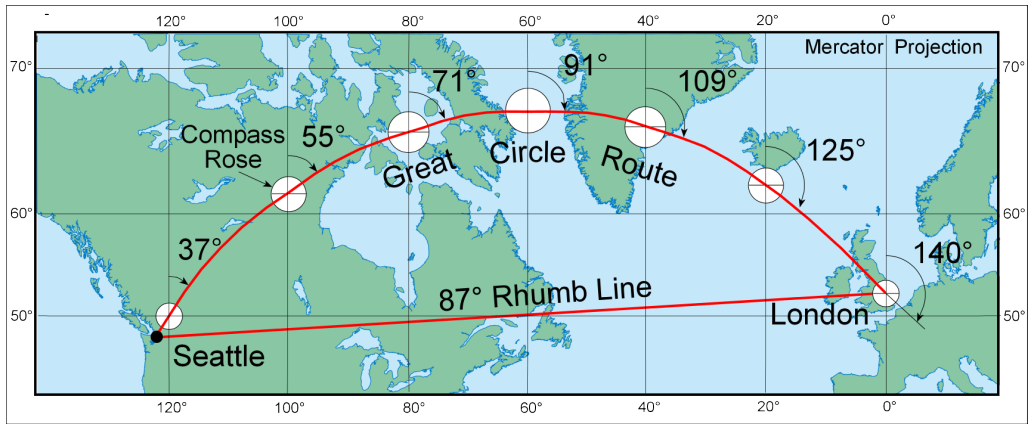


Geodesics

Definition: Geodesic = shortest path between a pair of points

- flat: geodesic = straight line
- sphere: geodesic = great circle
- ellipsoid: geodesic = not closed curve (*except meridians and equator*)



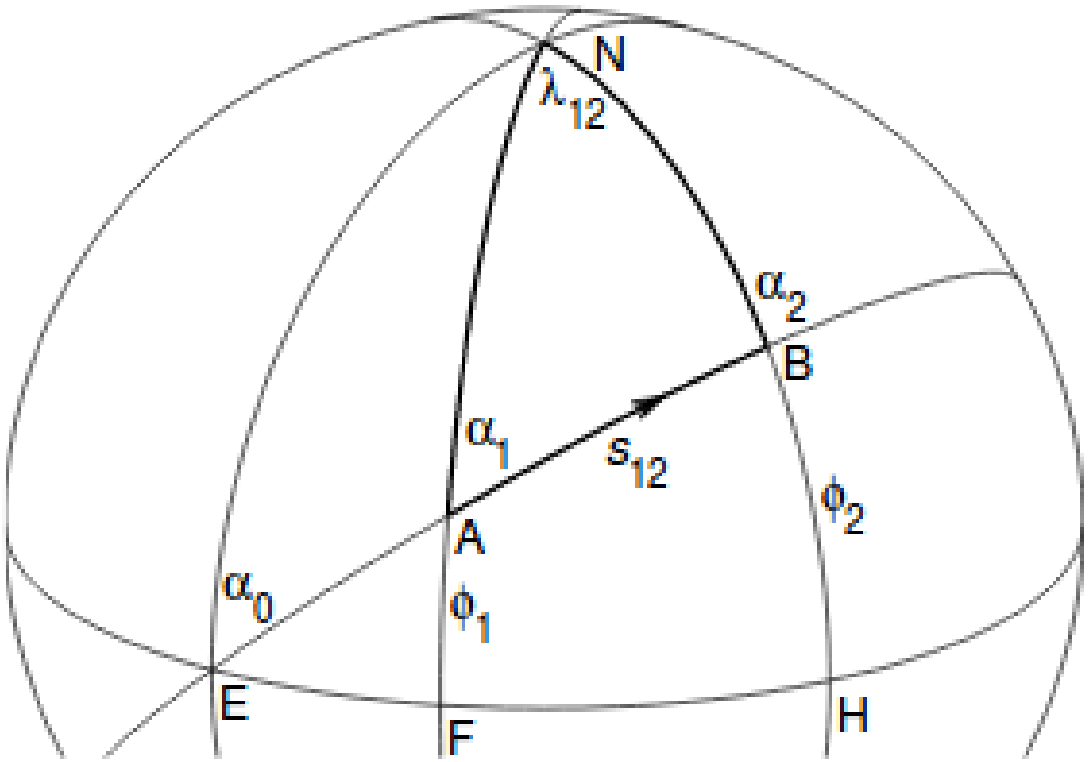


Note: loxodrome or rhump line is an arc crossing all meridians at the same angle (=azimuth). These are straight lines in Mercator projection and not shortest paths.

Geodesic problems

Direct given point, azimuth, distance compute new point

Inverse given two points compute distance, azimuth



(shortest) Distance between points

or inverse geodesic problem

flat: $s = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ (Pythagoras)

sphere: $s = \mathsf{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\mathsf{hav}(\lambda_2 - \lambda_1)$, where
$\mathsf{hav}(\theta) = \sin^2(\frac{\theta}{2})$ (Haversine formula)

ellipsoid:

$$\frac{s}{b} = \int_0^\sigma \sqrt{1 + k^2 \sin^2 \sigma'}\, \mathrm{d}\sigma', \tag{1}$$

$$\lambda = \omega - f \sin \alpha_0 \int_0^\sigma \frac{2 - f}{1 + (1 - f)\sqrt{1 + k^2 \sin^2 \sigma'}}\, \mathrm{d}\sigma'. \tag{2}$$

λ, ϕ, s: *longitude, latitude, distance*
ω, σ: *longitude, distance on auxiliary sphere*
α_0: *azimuth at the equatorial*
$k = e' \cos \alpha_0$ *and* f, e', b *constants*

Algorithms for geodesic problems

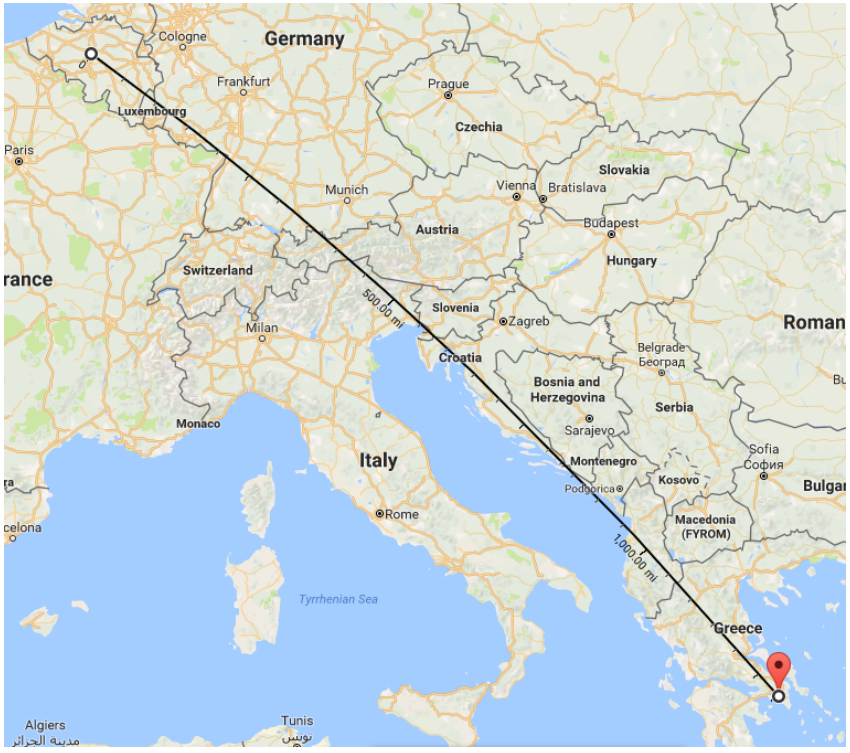
1. Higher series approximation *(up-to order 8)* + iterative methods [Bessel'1865, Karney'12]
2. Numerical integration [Sjoberg'12, Panou et al.'13]
3. Iterative methods [Vincenty'75] *(widely used)*
4. Series approximation of order 2 [Thomas'70]
5. Series approximation of order 1 [Andoyer'65,Thomas'65]
6. Spherical *(trigonometric approximation)*
7. inverse elliptic arc length formulas [Thomas'65]
8. Projection + cartesian methods (*local approximation*)

(1) GeographicLib (PR in Boost Geometry)
(2) not considered here
(3-8) supported by Boost Geometry

Example: distance between points on ellipsoid

Athens (23.725750, 37.971536) — Brussels(4.3826169, 50.8119483)

higher series	2088384.366
vincenty	2088384.366
thomas	2088384.364
andoyer	2088389.079
elliptic-0	2087512.071
elliptic-1	2088385.162
elliptic-2	2088384.422
elliptic-3	2088384.422
spherical	2085992.801
flat approx	2213471.759



Algorithms using geodesic problems

- distance between geometries
  use distance computation, inverse problem
- area
  use azimuth computation, inverse problem
- line interpolate point, densify
  use direct and inverse problems
- curve similarity: Hausdorff and Fréchet distance
  use distance computation, inverse problem

Example: distance between point and segment

```
L := LINESTRING(-16.42203 -7.52882,4.89998 -6.15568)
P := POINT(3.32696 -6.29345)
```

`Boost.Geometry (version 1.71.0)`

spherical:	508.0731159
andoyer:	480.3460262
thomas:	481.7457616
vincenty:	481.7390876

`POSTGIS (version 2.5)`

spherical:	508.073115931
geographic:	505.320928239

Homework: find the most accurate result!

Benchmarks

- Dataset:
  `https://zenodo.org/record/32156#.WOyaMrUmv7C`

- WGS84 ellipsoid
  1. 100000 geodesics uniform distributed in the ellipsoid and
  2. 50000 short geodesics (of length less than 10 km).

- `https://github.com/vissarion/geometry/wiki/Accuracy-and-performance-of-geographic-algorithms`

Performance

method	distance time	azimuth time
flat*	7	35
spherical*	15	27
andoyer*	117	260
thomas*	270	371
vincenty*	485	463
`GeographicLib`	1224	1228

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

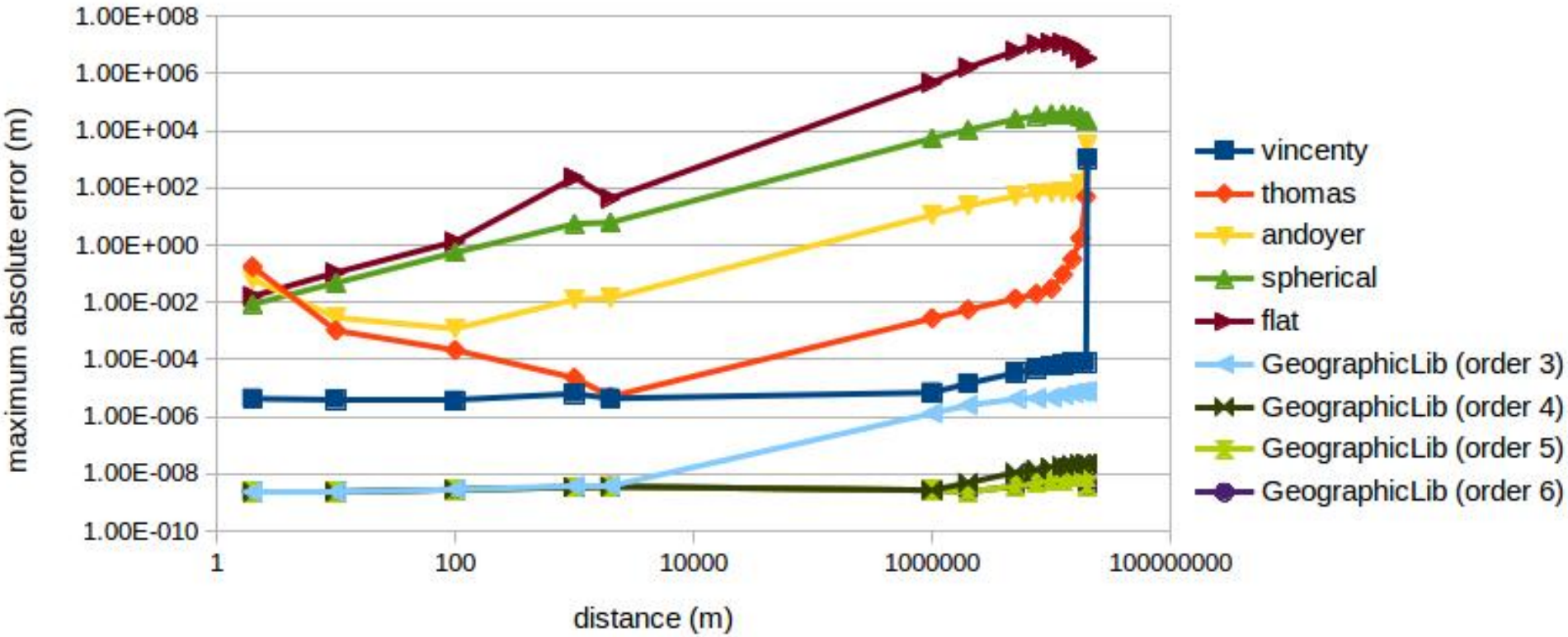
* *as implemented in* `Boost Geomerty`

Distance accuracy

- long geodesics ($> 1000km$): the faster the method is the lower the accuracy obtained
- geodesics with endpoints far from antipodal (geodesics of length less than $19500km$):
  spherical, andoyer, thomas, vincenty, high series have accuracy $21km$, $811m$, $48m$, $78\mu m$, $7nm$ respectively
- flat approximation: very low accuracy, only useful for very short distance *e.g.* distance between point$(10, 10)$ and point$(10.1, 10.002)$ w. cm accuracy
- The only method that is accurate (nm accuracy) for geodesics that has endpoints nearly antipodal is high series method

Accuracy

distance



Area and azimuth accuracy

- lower accuracy of short geodesics ($< 10km$)

For area we also test projection methods followed by cartesian area computation

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

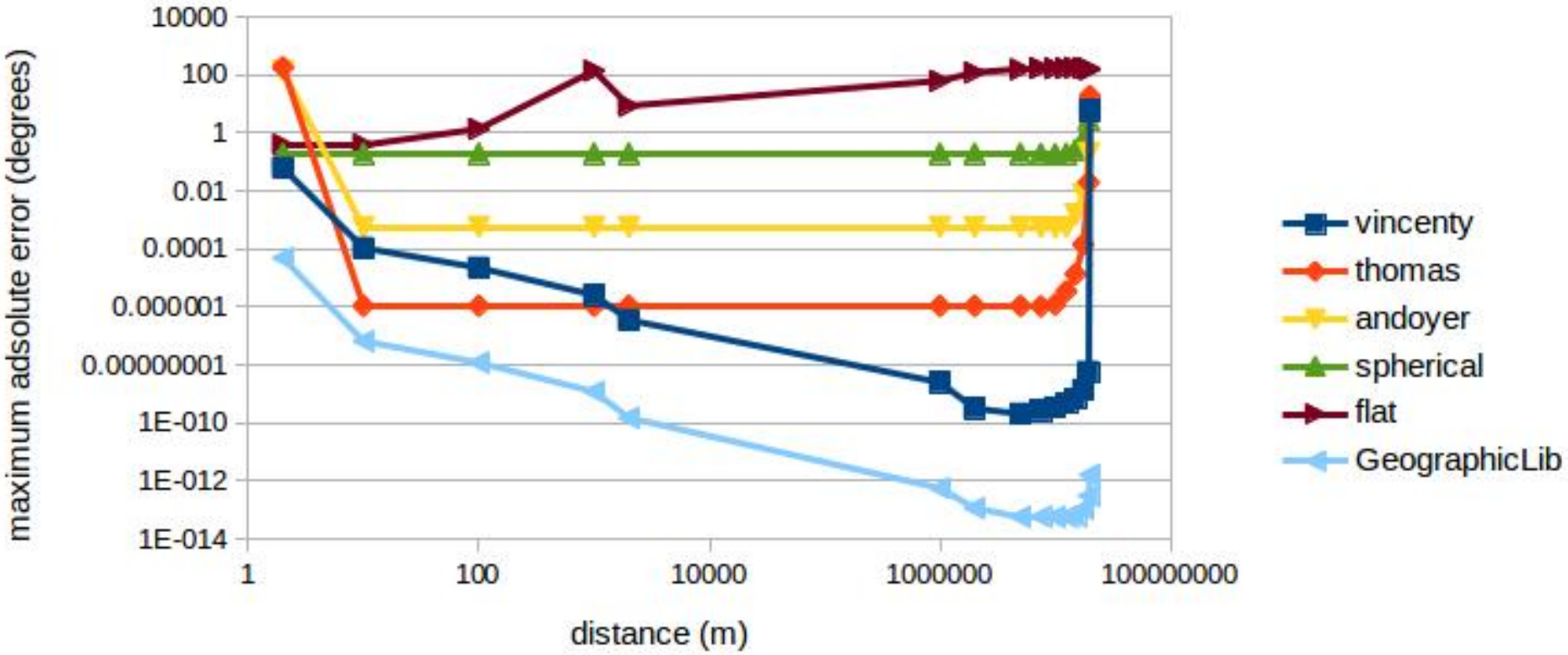
cea Equal Area Cylindrical
`https://proj4.org/operations/projections/cea.html`

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

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

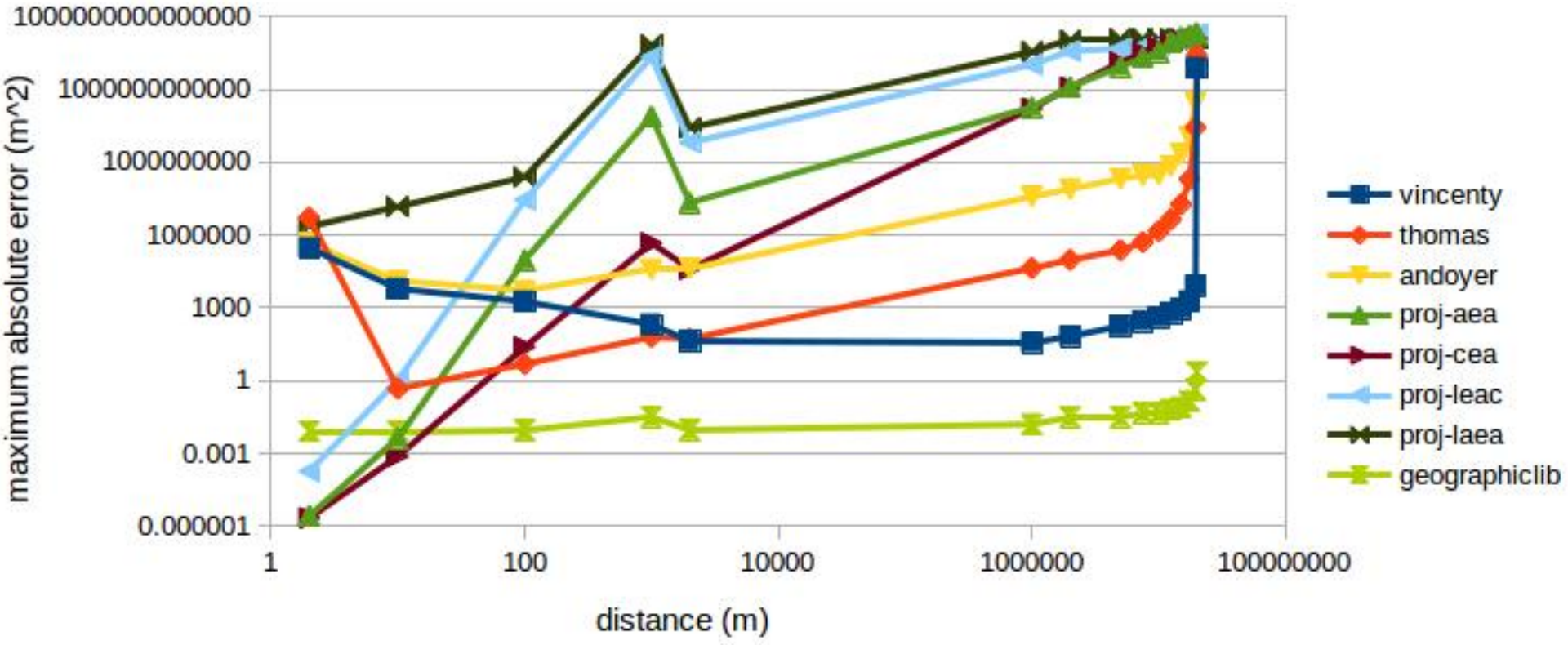
Accuracy

azimuth



Accuracy

area



Thank you!

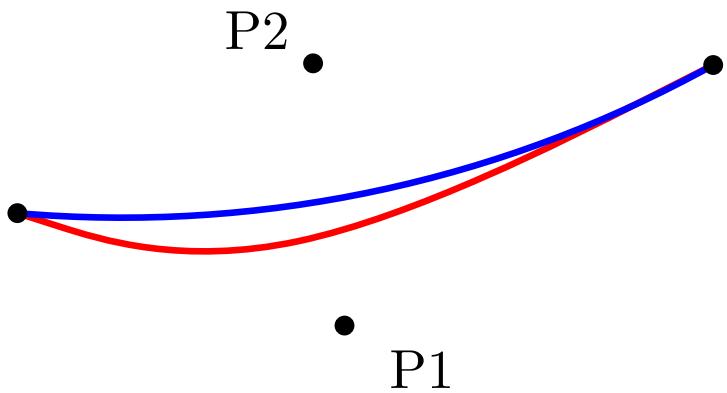
Questions?

Solution: distance between point and segment

```
L := LINESTRING(-16.42203 -7.52882,4.89998 -6.15568)
P1 := POINT(3.32696 -6.29345)
P2 := POINT(3.3262, -6.28451)
```

```
Boost.Geometry (version 1.71.0)
```

P1	spherical:	508.0731159
P1	vincenty:	481.7390876
P2	spherical	489.553212
P2	vincenty	510.4827314




```
POSTGIS (version 2.5)
```

P1	spherical:	508.073115931
P1	geographic:	505.320928239
P2	spherical:	489.553211954
P2	geographic:	486.901261892