Geographiclib 1.47

Contenu

[Introduction 3](#_Toc475446929)

[Installation 3](#_Toc475446930)

[Contents 3](#_Toc475446931)

[GeographicLib in various languages 4](#_Toc475446932)

[Change log 4](#_Toc475446933)

[Indices and tables 4](#_Toc475446934)

[Geodesics on an ellipsoid 5](#_Toc475446935)

[Introduction 5](#_Toc475446936)

[Additional properties 5](#_Toc475446937)

[Multiple shortest geodesics 6](#_Toc475446938)

[Background 6](#_Toc475446939)

[References 7](#_Toc475446940)

[The library interface 8](#_Toc475446941)

[The units 8](#_Toc475446942)

[Geodesic dictionary 8](#_Toc475446943)

[*outmask* and *caps* 8](#_Toc475446944)

[Restrictions on the parameters 9](#_Toc475446945)

[GeographicLib API 11](#_Toc475446946)

[geographiclib 11](#_Toc475446947)

[geographiclib.geodesic 11](#_Toc475446948)

[geographiclib.geodesicline 16](#_Toc475446949)

[geographiclib.polygonarea 18](#_Toc475446950)

[geographiclib.constants 21](#_Toc475446951)

[Examples 22](#_Toc475446952)

[Initializing 22](#_Toc475446953)

[Basic geodesic calculations 22](#_Toc475446954)

[Computing waypoints 22](#_Toc475446955)

[Measuring areas 23](#_Toc475446956)

geographiclib

Author: Charles F. F. Karney ([charles@karney.com](mailto:charles%40karney.com))

Version: 1.47

The documentation for other versions is available at http://geographiclib.sourceforge.net/m.nn/python/ for versions numbers m.nn ≥ 1.46.

Licensed under the MIT/X11 License; see [LICENSE.txt](http://geographiclib.sourceforge.net/html/LICENSE.txt).

<http://geographiclib.sourceforge.net/1.47/python/>

# Introduction

This is a python implementation of the geodesic routines in [GeographicLib](http://geographiclib.sourceforge.net).

Although it is maintained in conjunction with the larger C++ library, this python package can be used independently.

## Installation

The full [Geographic](http://geographiclib.sourceforge.net) package can be downloaded from [sourceforge](http://sourceforge.net/projects/geographiclib/files/distrib). However the python implementation is available as a stand-alone package. To install this, run

pip install geographiclib

Alternatively downloaded the package directly from [Python Package Index](http://pypi.python.org/pypi/geographiclib) and install it with

tar xpfz geographiclib-1.47.tar.gz

cd geographiclib-1.47

python setup.py install

It’s a good idea to run the unit tests to verify that the installation worked OK by running

python -m unittest geographiclib.test.test\_geodesic

## Contents

* [Geodesics on an ellipsoid](http://geographiclib.sourceforge.net/1.47/python/geodesics.html)
  + [Introduction](http://geographiclib.sourceforge.net/1.47/python/geodesics.html#introduction)
  + [Additional properties](http://geographiclib.sourceforge.net/1.47/python/geodesics.html#additional-properties)
  + [Multiple shortest geodesics](http://geographiclib.sourceforge.net/1.47/python/geodesics.html#multiple-shortest-geodesics)
  + [Background](http://geographiclib.sourceforge.net/1.47/python/geodesics.html#background)
  + [References](http://geographiclib.sourceforge.net/1.47/python/geodesics.html#references)
* [The library interface](http://geographiclib.sourceforge.net/1.47/python/interface.html)
  + [The units](http://geographiclib.sourceforge.net/1.47/python/interface.html#the-units)
  + [Geodesic dictionary](http://geographiclib.sourceforge.net/1.47/python/interface.html#geodesic-dictionary)
  + [outmask and caps](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask-and-caps)
  + [Restrictions on the parameters](http://geographiclib.sourceforge.net/1.47/python/interface.html#restrictions-on-the-parameters)
* [GeographicLib API](http://geographiclib.sourceforge.net/1.47/python/code.html)
  + [geographiclib](http://geographiclib.sourceforge.net/1.47/python/code.html#module-geographiclib)
  + [geographiclib.geodesic](http://geographiclib.sourceforge.net/1.47/python/code.html#module-geographiclib.geodesic)
  + [geographiclib.geodesicline](http://geographiclib.sourceforge.net/1.47/python/code.html#module-geographiclib.geodesicline)
  + [geographiclib.polygonarea](http://geographiclib.sourceforge.net/1.47/python/code.html#module-geographiclib.polygonarea)
  + [geographiclib.constants](http://geographiclib.sourceforge.net/1.47/python/code.html#module-geographiclib.constants)
* [Examples](http://geographiclib.sourceforge.net/1.47/python/examples.html)
  + [Initializing](http://geographiclib.sourceforge.net/1.47/python/examples.html#initializing)
  + [Basic geodesic calculations](http://geographiclib.sourceforge.net/1.47/python/examples.html#basic-geodesic-calculations)
  + [Computing waypoints](http://geographiclib.sourceforge.net/1.47/python/examples.html#computing-waypoints)
  + [Measuring areas](http://geographiclib.sourceforge.net/1.47/python/examples.html#measuring-areas)

## GeographicLib in various languages

* C++ (complete library): [documentation](http://geographiclib.sourceforge.net/1.47/index.html), [download](https://sourceforge.net/projects/geographiclib/files/distrib)
* C (geodesic routines): [documentation](http://geographiclib.sourceforge.net/1.47/C/index.html), also included with recent versions of [proj.4](https://github.com/OSGeo/proj.4/wiki)
* Fortran (geodesic routines): [documentation](http://geographiclib.sourceforge.net/1.47/Fortran/index.html)
* Java (geodesic routines): [Maven Central package](http://repo1.maven.org/maven2/net/sf/geographiclib/GeographicLib-Java/), [documentation](http://geographiclib.sourceforge.net/1.47/java/index.html)
* JavaScript (geodesic routines): [npm package](https://www.npmjs.com/package/geographiclib), [documentation](http://geographiclib.sourceforge.net/1.47/js/index.html)
* Python (geodesic routines): [PyPI package](http://pypi.python.org/pypi/geographiclib), [documentation](http://geographiclib.sourceforge.net/1.47/python/index.html)
* Matlab/Octave (geodesic and some other routines): [Matlab Central package](http://www.mathworks.com/matlabcentral/fileexchange/50605), [documentation](http://www.mathworks.com/matlabcentral/fileexchange/50605/content/Contents.m)
* C# (.NET wrapper for complete C++ library): [documentation](http://geographiclib.sourceforge.net/1.47/NET/index.html)

## Change log

* Version 1.47 (released 2017-02-15)
  + Fix the packaging, incorporating the patches in version 1.46.3.
  + Improve accuracy of area calculation (fixing a flaw introduced in version 1.46)
* Version 1.46 (released 2016-02-15)
  + Add Geodesic.DirectLine, Geodesic.ArcDirectLine, Geodesic.InverseLine, GeodesicLine.SetDistance, GeodesicLine.SetArc, GeodesicLine.s13, GeodesicLine.a13.
  + More accurate inverse solution when longitude difference is close to 180°.
  + Remove unnecessary functions, CheckPosition, CheckAzimuth, CheckDistance.

## Indices and tables

* [Index](http://geographiclib.sourceforge.net/1.47/python/genindex.html)
* [Module Index](http://geographiclib.sourceforge.net/1.47/python/py-modindex.html)
* [Search Page](http://geographiclib.sourceforge.net/1.47/python/search.html)

# Geodesics on an ellipsoid

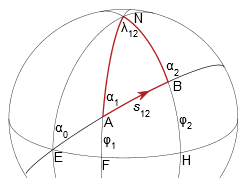
Jump to

* [Introduction](http://geographiclib.sourceforge.net/1.47/python/geodesics.html#intro)
* [Additional properties](http://geographiclib.sourceforge.net/1.47/python/geodesics.html#additional)
* [Multiple shortest geodesics](http://geographiclib.sourceforge.net/1.47/python/geodesics.html#multiple)
* [Background](http://geographiclib.sourceforge.net/1.47/python/geodesics.html#background)
* [References](http://geographiclib.sourceforge.net/1.47/python/geodesics.html#references)

## Introduction

Consider a ellipsoid of revolution with equatorial radius a, polar semi-axis b, and flattening f = (a − b)/a . Points on the surface of the ellipsoid are characterized by their latitude φ and longitude λ. (Note that latitude here means the geographical latitude, the angle between the normal to the ellipsoid and the equatorial plane).

The shortest path between two points on the ellipsoid at (φ1, λ1) and (φ2, λ2) is called the geodesic. Its length is s12 and the geodesic from point 1 to point 2 has forward azimuths α1 and α2 at the two end points. In this figure, we have λ12 = λ2 − λ1.



A geodesic can be extended indefinitely by requiring that any sufficiently small segment is a shortest path; geodesics are also the straightest curves on the surface.

Traditionally two geodesic problems are considered:

* the direct problem — given φ1, λ1, α1, s12, determine φ2, λ2, and α2; this is solved by [Geodesic.Direct](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Direct).
* the inverse problem — given φ1, λ1, φ2, λ2, determine s12, α1, and α2; this is solved by [Geodesic.Inverse](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Inverse).

## Additional properties

The routines also calculate several other quantities of interest

* S12 is the area between the geodesic from point 1 to point 2 and the equator; i.e., it is the area, measured counter-clockwise, of the quadrilateral with corners (φ1,λ1), (0,λ1), (0,λ2), and (φ2,λ2). It is given in meters2.
* m12, the reduced length of the geodesic is defined such that if the initial azimuth is perturbed by dα1 (radians) then the second point is displaced by m12 dα1 in the direction perpendicular to the geodesic. m12 is given in meters. On a curved surface the reduced length obeys a symmetry relation, m12 + m21 = 0. On a flat surface, we have m12 = s12.
* M12 and M21 are geodesic scales. If two geodesics are parallel at point 1 and separated by a small distance dt, then they are separated by a distance M12 dt at point 2. M21 is defined similarly (with the geodesics being parallel to one another at point 2). M12 and M21 are dimensionless quantities. On a flat surface, we have M12 = M21 = 1.
* σ12 is the arc length on the auxiliary sphere. This is a construct for converting the problem to one in spherical trigonometry. The spherical arc length from one equator crossing to the next is always 180°.

If points 1, 2, and 3 lie on a single geodesic, then the following addition rules hold:

* s13 = s12 + s23
* σ13 = σ12 + σ23
* S13 = S12 + S23
* m13 = m12M23 + m23M21
* M13 = M12M23 − (1 − M12M21) m23/m12
* M31 = M32M21 − (1 − M23M32) m12/m23

## Multiple shortest geodesics

The shortest distance found by solving the inverse problem is (obviously) uniquely defined. However, in a few special cases there are multiple azimuths which yield the same shortest distance. Here is a catalog of those cases:

* φ1 = −φ2 (with neither point at a pole). If α1 = α2, the geodesic is unique. Otherwise there are two geodesics and the second one is obtained by setting [α1,α2] ← [α2,α1], [M12,M21] ← [M21,M12], S12 ← −S12. (This occurs when the longitude difference is near ±180° for oblate ellipsoids.)
* λ2 = λ1 ± 180° (with neither point at a pole). If α1 = 0° or ±180°, the geodesic is unique. Otherwise there are two geodesics and the second one is obtained by setting [α1,α2] ← [−α1,−α2], S12 ← −S12. (This occurs when φ2 is near −φ1 for prolate ellipsoids.)
* Points 1 and 2 at opposite poles. There are infinitely many geodesics which can be generated by setting [α1,α2] ← [α1,α2] + [δ,−δ], for arbitrary δ. (For spheres, this prescription applies when points 1 and 2 are antipodal.)
* s12 = 0 (coincident points). There are infinitely many geodesics which can be generated by setting [α1,α2] ← [α1,α2] + [δ,δ], for arbitrary δ.

## Background

The algorithms implemented by this package are given in Karney (2013) and are based on Bessel (1825) and Helmert (1880); the algorithm for areas is based on Danielsen (1989). These improve on the work of Vincenty (1975) in the following respects:

* The results are accurate to round-off for terrestrial ellipsoids (the error in the distance is less then 15 nanometers, compared to 0.1 mm for Vincenty).
* The solution of the inverse problem is always found. (Vincenty’s method fails to converge for nearly antipodal points.)
* The routines calculate differential and integral properties of a geodesic. This allows, for example, the area of a geodesic polygon to be computed.

## References

* F. W. Bessel, [The calculation of longitude and latitude from geodesic measurements (1825)](https://arxiv.org/abs/0908.1824), Astron. Nachr. **331**(8), 852–861 (2010), translated by C. F. F. Karney and R. E. Deakin.
* F. R. Helmert, [Mathematical and Physical Theories of Higher Geodesy, Vol 1](https://doi.org/10.5281/zenodo.32050), (Teubner, Leipzig, 1880), Chaps. 5–7.
* T. Vincenty, [Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations](http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf), Survey Review **23**(176), 88–93 (1975).
* J. Danielsen, [The area under the geodesic](https://doi.org/10.1179/003962689791474267), Survey Review **30**(232), 61–66 (1989).
* C. F. F. Karney, [Algorithms for geodesics](https://doi.org/10.1007/s00190-012-0578-z), J. Geodesy **87**(1) 43–55 (2013); [addenda](http://geographiclib.sourceforge.net/geod-addenda.html).
* C. F. F. Karney, [Geodesics on an ellipsoid of revolution](https://arxiv.org/abs/1102.1215v1), Feb. 2011; [errata](http://geographiclib.sourceforge.net/geod-addenda.html#geod-errata).
* [A geodesic bibliography](http://geographiclib.sourceforge.net/geodesic-papers/biblio.html).
* The wikipedia page, [Geodesics on an ellipsoid](https://en.wikipedia.org/wiki/Geodesics_on_an_ellipsoid).

# The library interface

Jump to

* [The units](http://geographiclib.sourceforge.net/1.47/python/interface.html#units)
* [Geodesic dictionary](http://geographiclib.sourceforge.net/1.47/python/interface.html#dict)
* [outmask and caps](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask)
* [Restrictions on the parameters](http://geographiclib.sourceforge.net/1.47/python/interface.html#restrictions)

## The units

All angles (latitude, longitude, azimuth, arc length) are measured in degrees with latitudes increasing northwards, longitudes increasing eastwards, and azimuths measured clockwise from north. For a point at a pole, the azimuth is defined by keeping the longitude fixed, writing φ = ±(90° − ε), and taking the limit ε → 0+

## Geodesic dictionary

The results returned by [Geodesic.Direct](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Direct), [Geodesic.Inverse](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Inverse), [GeodesicLine.Position](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.Position), etc., return a dictionary with some of the following 12 fields set:

* lat1 = φ1, latitude of point 1 (degrees)
* lon1 = λ1, longitude of point 1 (degrees)
* azi1 = α1, azimuth of line at point 1 (degrees)
* lat2 = φ2, latitude of point 2 (degrees)
* lon2 = λ2, longitude of point 2 (degrees)
* azi2 = α2, (forward) azimuth of line at point 2 (degrees)
* s12 = s12, distance from 1 to 2 (meters)
* a12 = σ12, arc length on auxiliary sphere from 1 to 2 (degrees)
* m12 = m12, reduced length of geodesic (meters)
* M12 = M12, geodesic scale at 2 relative to 1 (dimensionless)
* M21 = M21, geodesic scale at 1 relative to 2 (dimensionless)
* S12 = S12, area between geodesic and equator (meters2)

## outmask and caps

By default, the geodesic routines return the 7 basic quantities: lat1, lon1, azi1, lat2, lon2, azi2, s12, together with the arc length a12. The optional output mask parameter, outmask, can be used to tailor which quantities to calculate. In addition, when a [GeodesicLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine) is constructed it can be provided with the optional capabilities parameter, caps, which specifies what quantities can be returned from the resulting object.

Both outmask and caps are obtained by or’ing together the following values

* [EMPTY](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.EMPTY), no capabilities, no output
* [LATITUDE](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.LATITUDE), compute latitude, lat2
* [LONGITUDE](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.LONGITUDE), compute longitude, lon2
* [AZIMUTH](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.AZIMUTH), compute azimuths, azi1 and azi2
* [DISTANCE](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.DISTANCE), compute distance, s12
* [STANDARD](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.STANDARD), all of the above
* [DISTANCE\_IN](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.DISTANCE_IN), allow s12 to be used as input in the direct problem
* [REDUCEDLENGTH](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.REDUCEDLENGTH), compute reduced length, m12
* [GEODESICSCALE](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.GEODESICSCALE), compute geodesic scales, M12 and M21
* [AREA](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.AREA), compute area, S12
* [ALL](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.ALL), all of the above;
* [LONG\_UNROLL](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.LONG_UNROLL), unroll longitudes

DISTANCE\_IN is a capability provided to the GeodesicLine constructor. It allows the position on the line to specified in terms of distance. (Without this, the position can only be specified in terms of the arc length.) This only makes sense in the caps parameter.

LONG\_UNROLL controls the treatment of longitude. If it is not set then the lon1 and lon2 fields are both reduced to the range [−180°, 180°). If it is set, then lon1 is as given in the function call and (lon2 − lon1) determines how many times and in what sense the geodesic has encircled the ellipsoid. This only makes sense in the outmask parameter.

Note that a12 is always included in the result.

## Restrictions on the parameters

* Latitudes must lie in [−90°, 90°]. Latitudes outside this range are replaced by NaNs.
* The distance s12 is unrestricted. This allows geodesics to wrap around the ellipsoid. Such geodesics are no longer shortest paths. However they retain the property that they are the straightest curves on the surface.
* Similarly, the spherical arc length a12 is unrestricted.
* Longitudes and azimuths are unrestricted; internally these are exactly reduced to the range [−180°, 180°); but see also the LONG\_UNROLL bit.
* The equatorial radius a and the polar semi-axis b must both be positive and finite (this implies that −∞ < f < 1).
* The flattening f should satisfy f ∈ [−1/50,1/50] in order to retain full accuracy. This condition holds for most applications in geodesy.

Reasonably accurate results can be obtained for −0.2 ≤ f ≤ 0.2. Here is a table of the approximate maximum error (expressed as a distance) for an ellipsoid with the same equatorial radius as the WGS84 ellipsoid and different values of the flattening.

|  |  |
| --- | --- |
| abs(f) | error |
| 0.003 | 15 nm |
| 0.01 | 25 nm |
| 0.02 | 30 nm |
| 0.05 | 10 μm |
| 0.1 | 1.5 mm |
| 0.2 | 300 mm |

Here 1 nm = 1 nanometer = 10−9 m (not 1 nautical mile!)

# GeographicLib API

## geographiclib

geographiclib: geodesic routines from GeographicLib

geographiclib.\_\_version\_info\_\_ = (1, 47, 0)

GeographicLib version as a tuple

geographiclib.\_\_version\_\_ = '1.47'

GeographicLib version as a string

## geographiclib.geodesic

Define the [Geodesic](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic) class

The ellipsoid parameters are defined by the constructor. The direct and inverse geodesic problems are solved by

* [Inverse()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Inverse) Solve the inverse geodesic problem
* [Direct()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Direct) Solve the direct geodesic problem
* [ArcDirect()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.ArcDirect) Solve the direct geodesic problem in terms of spherical arc length

[GeodesicLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine) objects can be created with

* [Line()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Line)
* [DirectLine()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.DirectLine)
* [ArcDirectLine()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.ArcDirectLine)
* [InverseLine()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.InverseLine)

[PolygonArea](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea) objects can be created with

* [Polygon()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Polygon)

The public attributes for this class are

* [a](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.a) [f](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.f)

outmask and caps bit masks are

* [EMPTY](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.EMPTY)
* [LATITUDE](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.LATITUDE)
* [LONGITUDE](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.LONGITUDE)
* [AZIMUTH](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.AZIMUTH)
* [DISTANCE](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.DISTANCE)
* [STANDARD](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.STANDARD)
* [DISTANCE\_IN](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.DISTANCE_IN)
* [REDUCEDLENGTH](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.REDUCEDLENGTH)
* [GEODESICSCALE](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.GEODESICSCALE)
* [AREA](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.AREA)
* [ALL](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.ALL)
* [LONG\_UNROLL](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.LONG_UNROLL)

|  |  |
| --- | --- |
| **Example:** | >>> from geographiclib.geodesic import Geodesic  >>> # The geodesic inverse problem  ... Geodesic.WGS84.Inverse(-41.32, 174.81, 40.96, -5.50)  {'lat1': -41.32,  'a12': 179.6197069334283,  's12': 19959679.26735382,  'lat2': 40.96,  'azi2': 18.825195123248392,  'azi1': 161.06766998615882,  'lon1': 174.81,  'lon2': -5.5} |

Geodesic.WGS84 = Instantiation for the WGS84 ellipsoid

class geographiclib.geodesic.Geodesic(a, f)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesic.html#Geodesic)

Solve geodesic problems

Construct a Geodesic object

|  |  |
| --- | --- |
| **Parameters:** | * **a** – the equatorial radius of the ellipsoid in meters * **f** – the flattening of the ellipsoid |

An exception is thrown if a or the polar semi-axis b = a (1 - f) is not a finite positive quantity.

a = None

The equatorial radius in meters (readonly)

f = None

The flattening (readonly)

Inverse(lat1, lon1, lat2, lon2, outmask=1929)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesic.html#Geodesic.Inverse)

Solve the inverse geodesic problem

|  |  |
| --- | --- |
| **Parameters:** | * **lat1** – latitude of the first point in degrees * **lon1** – longitude of the first point in degrees * **lat2** – latitude of the second point in degrees * **lon2** – longitude of the second point in degrees * **outmask** – the [output mask](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) |
| **Returns:** | a [Geodesic dictionary](http://geographiclib.sourceforge.net/1.47/python/interface.html#dict) |

Compute geodesic between (lat1, lon1) and (lat2, lon2). The default value of outmask is STANDARD, i.e., the lat1, lon1, azi1, lat2, lon2, azi2, s12, a12 entries are returned.

Direct(lat1, lon1, azi1, s12, outmask=1929)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesic.html#Geodesic.Direct)

Solve the direct geodesic problem

|  |  |
| --- | --- |
| **Parameters:** | * **lat1** – latitude of the first point in degrees * **lon1** – longitude of the first point in degrees * **azi1** – azimuth at the first point in degrees * **s12** – the distance from the first point to the second in meters * **outmask** – the [output mask](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) |
| **Returns:** | a [Geodesic dictionary](http://geographiclib.sourceforge.net/1.47/python/interface.html#dict) |

Compute geodesic starting at (lat1, lon1) with azimuth azi1 and length s12. The default value of outmask is STANDARD, i.e., the lat1, lon1, azi1, lat2, lon2, azi2, s12, a12 entries are returned.

ArcDirect(lat1, lon1, azi1, a12, outmask=1929)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesic.html#Geodesic.ArcDirect)

Solve the direct geodesic problem in terms of spherical arc length

|  |  |
| --- | --- |
| **Parameters:** | * **lat1** – latitude of the first point in degrees * **lon1** – longitude of the first point in degrees * **azi1** – azimuth at the first point in degrees * **a12** – spherical arc length from the first point to the second in degrees * **outmask** – the [output mask](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) |
| **Returns:** | a [Geodesic dictionary](http://geographiclib.sourceforge.net/1.47/python/interface.html#dict) |

Compute geodesic starting at (lat1, lon1) with azimuth azi1 and arc length a12. The default value of outmask is STANDARD, i.e., the lat1, lon1, azi1, lat2, lon2, azi2, s12, a12 entries are returned.

Line(lat1, lon1, azi1, caps=3979)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesic.html#Geodesic.Line)

Return a GeodesicLine object

|  |  |
| --- | --- |
| **Parameters:** | * **lat1** – latitude of the first point in degrees * **lon1** – longitude of the first point in degrees * **azi1** – azimuth at the first point in degrees * **caps** – the [capabilities](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) |
| **Returns:** | a [GeodesicLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine) |

This allows points along a geodesic starting at (lat1, lon1), with azimuth azi1 to be found. The default value of caps is STANDARD | DISTANCE\_IN, allowing direct geodesic problem to be solved.

DirectLine(lat1, lon1, azi1, s12, caps=3979)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesic.html#Geodesic.DirectLine)

Define a GeodesicLine object in terms of the direct geodesic problem specified in terms of spherical arc length

|  |  |
| --- | --- |
| **Parameters:** | * **lat1** – latitude of the first point in degrees * **lon1** – longitude of the first point in degrees * **azi1** – azimuth at the first point in degrees * **s12** – the distance from the first point to the second in meters * **caps** – the [capabilities](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) |
| **Returns:** | a [GeodesicLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine) |

This function sets point 3 of the GeodesicLine to correspond to point 2 of the direct geodesic problem. The default value of caps is STANDARD | DISTANCE\_IN, allowing direct geodesic problem to be solved.

ArcDirectLine(lat1, lon1, azi1, a12, caps=3979)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesic.html#Geodesic.ArcDirectLine)

Define a GeodesicLine object in terms of the direct geodesic problem specified in terms of spherical arc length

|  |  |
| --- | --- |
| **Parameters:** | * **lat1** – latitude of the first point in degrees * **lon1** – longitude of the first point in degrees * **azi1** – azimuth at the first point in degrees * **a12** – spherical arc length from the first point to the second in degrees * **caps** – the [capabilities](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) |
| **Returns:** | a [GeodesicLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine) |

This function sets point 3 of the GeodesicLine to correspond to point 2 of the direct geodesic problem. The default value of caps is STANDARD | DISTANCE\_IN, allowing direct geodesic problem to be solved.

InverseLine(lat1, lon1, lat2, lon2, caps=3979)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesic.html#Geodesic.InverseLine)

Define a GeodesicLine object in terms of the invese geodesic problem

|  |  |
| --- | --- |
| **Parameters:** | * **lat1** – latitude of the first point in degrees * **lon1** – longitude of the first point in degrees * **lat2** – latitude of the second point in degrees * **lon2** – longitude of the second point in degrees * **caps** – the [capabilities](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) |
| **Returns:** | a [GeodesicLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine) |

This function sets point 3 of the GeodesicLine to correspond to point 2 of the inverse geodesic problem. The default value of caps is STANDARD | DISTANCE\_IN, allowing direct geodesic problem to be solved.

Polygon(polyline=False)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesic.html#Geodesic.Polygon)

Return a PolygonArea object

|  |  |
| --- | --- |
| **Parameters:** | **polyline** – if True then the object describes a polyline instead of a polygon |
| **Returns:** | a [PolygonArea](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea) |

EMPTY = 0

No capabilities, no output.

LATITUDE = 128

Calculate latitude lat2.

LONGITUDE = 264

Calculate longitude lon2.

AZIMUTH = 512

Calculate azimuths azi1 and azi2.

DISTANCE = 1025

Calculate distance s12.

STANDARD = 1929

All of the above.

DISTANCE\_IN = 2051

Allow distance s12 to be used as input in the direct geodesic problem.

REDUCEDLENGTH = 4101

Calculate reduced length m12.

GEODESICSCALE = 8197

Calculate geodesic scales M12 and M21.

AREA = 16400

Calculate area S12.

ALL = 32671

All of the above.

LONG\_UNROLL = 32768

Unroll longitudes, rather than reducing them to the reducing them to the range [-180d,180d).

## geographiclib.geodesicline

Define the [GeodesicLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine) class

The constructor defines the starting point of the line. Points on the line are given by

* [Position()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.Position) position given in terms of distance
* [ArcPosition()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.ArcPosition) position given in terms of spherical arc length

A reference point 3 can be defined with

* [SetDistance()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.SetDistance) set position of 3 in terms of the distance from the starting point
* [SetArc()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.SetArc) set position of 3 in terms of the spherical arc length from the starting point

The object can also be constructed by

* [Geodesic.Line](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Line)
* [Geodesic.DirectLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.DirectLine)
* [Geodesic.ArcDirectLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.ArcDirectLine)
* [Geodesic.InverseLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.InverseLine)

The public attributes for this class are

* [a](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.a) [f](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.f) [caps](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.caps) [lat1](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.lat1) [lon1](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.lon1) [azi1](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.azi1) [salp1](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.salp1) [calp1](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.calp1) [s13](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.s13) [a13](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.a13)

class geographiclib.geodesicline.GeodesicLine(geod, lat1, lon1, azi1, caps=3979, salp1=nan, calp1=nan)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesicline.html#GeodesicLine)

Points on a geodesic path

Construct a GeodesicLine object

|  |  |
| --- | --- |
| **Parameters:** | * **geod** – a [Geodesic](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic) object * **lat1** – latitude of the first point in degrees * **lon1** – longitude of the first point in degrees * **azi1** – azimuth at the first point in degrees * **caps** – the [capabilities](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) |

This creates an object allowing points along a geodesic starting at (lat1, lon1), with azimuth azi1 to be found. The default value of caps is STANDARD | DISTANCE\_IN. The optional parameters salp1 and calp1 should not be supplied; they are part of the private interface.

a = None

The equatorial radius in meters (readonly)

f = None

The flattening (readonly)

caps = None

the capabilities (readonly)

lat1 = None

the latitude of the first point in degrees (readonly)

lon1 = None

the longitude of the first point in degrees (readonly)

azi1 = None

the azimuth at the first point in degrees (readonly)

salp1 = None

the sine of the azimuth at the first point (readonly)

calp1 = None

the cosine of the azimuth at the first point (readonly)

s13 = None

the distance between point 1 and point 3 in meters (readonly)

a13 = None

the arc length between point 1 and point 3 in degrees (readonly)

Position(s12, outmask=1929)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesicline.html#GeodesicLine.Position)

Find the position on the line given s12

|  |  |
| --- | --- |
| **Parameters:** | * **s12** – the distance from the first point to the second in meters * **outmask** – the [output mask](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) |
| **Returns:** | a [Geodesic dictionary](http://geographiclib.sourceforge.net/1.47/python/interface.html#dict) |

The default value of outmask is STANDARD, i.e., the lat1, lon1, azi1, lat2, lon2, azi2, s12, a12 entries are returned. The [GeodesicLine](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine) object must have been constructed with the DISTANCE\_IN capability.

ArcPosition(a12, outmask=1929)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesicline.html#GeodesicLine.ArcPosition)

Find the position on the line given a12

|  |  |
| --- | --- |
| **Parameters:** | * **a12** – spherical arc length from the first point to the second in degrees * **outmask** – the [output mask](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) |
| **Returns:** | a [Geodesic dictionary](http://geographiclib.sourceforge.net/1.47/python/interface.html#dict) |

The default value of outmask is STANDARD, i.e., the lat1, lon1, azi1, lat2, lon2, azi2, s12, a12 entries are returned.

SetDistance(s13)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesicline.html#GeodesicLine.SetDistance)

Specify the position of point 3 in terms of distance

|  |  |
| --- | --- |
| **Parameters:** | **s13** – distance from point 1 to point 3 in meters |

SetArc(a13)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/geodesicline.html#GeodesicLine.SetArc)

Specify the position of point 3 in terms of arc length

|  |  |
| --- | --- |
| **Parameters:** | **a13** – spherical arc length from point 1 to point 3 in degrees |

## geographiclib.polygonarea

Define the [PolygonArea](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea) class

The constructor initializes a empty polygon. The available methods are

* [Clear()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.Clear) reset the polygon
* [AddPoint()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.AddPoint) add a vertex to the polygon
* [AddEdge()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.AddEdge) add an edge to the polygon
* [Compute()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.Compute) compute the properties of the polygon
* [TestPoint()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.TestPoint) compute the properties of the polygon with a tentative additional vertex
* [TestEdge()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.TestEdge) compute the properties of the polygon with a tentative additional edge

The public attributes for this class are

* [earth](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.earth) [polyline](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.polyline) [area0](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.area0) [num](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.num) [lat1](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.lat1) [lon1](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea.lon1)

class geographiclib.polygonarea.PolygonArea(earth, polyline=False)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/polygonarea.html#PolygonArea)

Area of a geodesic polygon

Construct a PolygonArea object

|  |  |
| --- | --- |
| **Parameters:** | * **earth** – a [Geodesic](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic) object * **polyline** – if true, treat object as a polyline instead of a polygon |

Initially the polygon has no vertices.

earth = None

The geodesic object (readonly)

polyline = None

Is this a polyline? (readonly)

area0 = None

The total area of the ellipsoid in meter^2 (readonly)

num = None

The current number of points in the polygon (readonly)

lat1 = None

The current latitude in degrees (readonly)

lon1 = None

The current longitude in degrees (readonly)

Clear()[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/polygonarea.html#PolygonArea.Clear)

Reset to empty polygon.

AddPoint(lat, lon)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/polygonarea.html#PolygonArea.AddPoint)

Add the next vertex to the polygon

|  |  |
| --- | --- |
| **Parameters:** | * **lat** – the latitude of the point in degrees * **lon** – the longitude of the point in degrees |

This adds an edge from the current vertex to the new vertex.

AddEdge(azi, s)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/polygonarea.html#PolygonArea.AddEdge)

Add the next edge to the polygon

|  |  |
| --- | --- |
| **Parameters:** | * **azi** – the azimuth at the current the point in degrees * **s** – the length of the edge in meters |

This specifies the new vertex in terms of the edge from the current vertex.

Compute(reverse=False, sign=True)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/polygonarea.html#PolygonArea.Compute)

Compute the properties of the polygon

|  |  |
| --- | --- |
| **Parameters:** | * **reverse** – if true then clockwise (instead of counter-clockwise) traversal counts as a positive area * **sign** – if true then return a signed result for the area if the polygon is traversed in the “wrong” direction instead of returning the area for the rest of the earth |
| **Returns:** | a tuple of number, perimeter (meters), area (meters^2) |

If the object is a polygon (and not a polygon), the perimeter includes the length of a final edge connecting the current point to the initial point. If the object is a polyline, then area is nan.

More points can be added to the polygon after this call.

TestPoint(lat, lon, reverse=False, sign=True)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/polygonarea.html#PolygonArea.TestPoint)

Compute the properties for a tentative additional vertex

|  |  |
| --- | --- |
| **Parameters:** | * **lat** – the latitude of the point in degrees * **lon** – the longitude of the point in degrees * **reverse** – if true then clockwise (instead of counter-clockwise) traversal counts as a positive area * **sign** – if true then return a signed result for the area if the polygon is traversed in the “wrong” direction instead of returning the area for the rest of the earth |
| **Returns:** | a tuple of number, perimeter (meters), area (meters^2) |

TestEdge(azi, s, reverse=False, sign=True)[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/polygonarea.html#PolygonArea.TestEdge)

Compute the properties for a tentative additional edge

|  |  |
| --- | --- |
| **Parameters:** | * **azi** – the azimuth at the current the point in degrees * **s** – the length of the edge in meters * **reverse** – if true then clockwise (instead of counter-clockwise) traversal counts as a positive area * **sign** – if true then return a signed result for the area if the polygon is traversed in the “wrong” direction instead of returning the area for the rest of the earth |
| **Returns:** | a tuple of number, perimeter (meters), area (meters^2) |

## geographiclib.constants

Define the WGS84 ellipsoid

class geographiclib.constants.Constants[[source]](http://geographiclib.sourceforge.net/1.47/python/_modules/geographiclib/constants.html" \l "Constants)

Constants describing the WGS84 ellipsoid

WGS84\_a = 6378137.0

the equatorial radius in meters of the WGS84 ellipsoid in meters

WGS84\_f = 0.0033528106647474805

the flattening of the WGS84 ellipsoid, 1/298.257223563

# Examples

## Initializing

The following examples all assume that the following commands have been carried out:

>>> from geographiclib.geodesic import Geodesic

>>> import math

>>> geod = Geodesic.WGS84 # define the WGS84 ellipsoid

You can determine the ellipsoid parameters with the a and f member variables, for example,

>>> geod.a, 1/geod.f

(6378137.0, 298.257223563)

If you need to use a different ellipsoid, construct one by, for example

>>> geod = Geodesic(6378388, 1/297.0) # the international ellipsoid

## Basic geodesic calculations

The distance from Wellington, NZ (41.32S, 174.81E) to Salamanca, Spain (40.96N, 5.50W) using [Inverse()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Inverse):

>>> g = geod.Inverse(-41.32, 174.81, 40.96, -5.50)

>>> print "The distance is {:.3f} m.".format(g['s12'])

The distance is 19959679.267 m.

The point the point 20000 km SW of Perth, Australia (32.06S, 115.74E) using [Direct()](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Direct):

>>> g = geod.Direct(-32.06, 115.74, 225, 20000e3)

>>> print "The position is ({:.8f}, {:.8f}).".format(g['lat2'],g['lon2'])

The position is (32.11195529, -63.95925278).

The area between the geodesic from JFK Airport (40.6N, 73.8W) to LHR Airport (51.6N, 0.5W) and the equator. This is an example of setting the the [output mask](http://geographiclib.sourceforge.net/1.47/python/interface.html#outmask) parameter.

>>> g = geod.Inverse(40.6, -73.8, 51.6, -0.5, Geodesic.AREA)

>>> print "The area is {:.1f} m^2".format(g['S12'])

The area is 40041368848742.5 m^2

## Computing waypoints

Consider the geodesic between Beijing Airport (40.1N, 116.6E) and San Fransisco Airport (37.6N, 122.4W). Compute waypoints and azimuths at intervals of 1000 km using [Geodesic.Line](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Line) and [GeodesicLine.Position](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.Position):

>>> l = geod.InverseLine(40.1, 116.6, 37.6, -122.4)

>>> ds = 1000e3; n = int(math.ceil(l.s13 / ds))

>>> for i in range(n + 1):

... if i == 0:

... print "distance latitude longitude azimuth"

... s = min(ds \* i, l.s13)

... g = l.Position(s, Geodesic.STANDARD | Geodesic.LONG\_UNROLL)

... print "{:.0f} {:.5f} {:.5f} {:.5f}".format(

... g['s12'], g['lat2'], g['lon2'], g['azi2'])

...

distance latitude longitude azimuth

0 40.10000 116.60000 42.91642

1000000 46.37321 125.44903 48.99365

2000000 51.78786 136.40751 57.29433

3000000 55.92437 149.93825 68.24573

4000000 58.27452 165.90776 81.68242

5000000 58.43499 183.03167 96.29014

6000000 56.37430 199.26948 109.99924

7000000 52.45769 213.17327 121.33210

8000000 47.19436 224.47209 129.98619

9000000 41.02145 233.58294 136.34359

9513998 37.60000 237.60000 138.89027

The inclusion of Geodesic.LONG\_UNROLL in the call to GeodesicLine.Position ensures that the longitude does not jump on crossing the international dateline.

If the purpose of computing the waypoints is to plot a smooth geodesic, then it’s not important that they be exactly equally spaced. In this case, it’s faster to parameterize the line in terms of the spherical arc length with [GeodesicLine.ArcPosition](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesicline.GeodesicLine.ArcPosition). Here the spacing is about 1° of arc which means that the distance between the waypoints will be about 60 NM.

>>> l = geod.InverseLine(40.1, 116.6, 37.6, -122.4,

... Geodesic.LATITUDE | Geodesic.LONGITUDE)

>>> da = 1; n = int(math.ceil(l.a13 / da)); da = l.a13 / n

>>> for i in range(n + 1):

... if i == 0:

... print "latitude longitude"

... a = da \* i

... g = l.ArcPosition(a, Geodesic.LATITUDE |

... Geodesic.LONGITUDE | Geodesic.LONG\_UNROLL)

... print "{:.5f} {:.5f}".format(g['lat2'], g['lon2'])

...

latitude longitude

40.10000 116.60000

40.82573 117.49243

41.54435 118.40447

42.25551 119.33686

42.95886 120.29036

43.65403 121.26575

44.34062 122.26380

...

39.82385 235.05331

39.08884 235.91990

38.34746 236.76857

37.60000 237.60000

The variation in the distance between these waypoints is on the order of 1/f.

## Measuring areas

Measure the area of Antarctica using [Geodesic.Polygon](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.geodesic.Geodesic.Polygon) and the [PolygonArea](http://geographiclib.sourceforge.net/1.47/python/code.html#geographiclib.polygonarea.PolygonArea) class:

>>> p = geod.Polygon()

>>> antarctica = [

... [-63.1, -58], [-72.9, -74], [-71.9,-102], [-74.9,-102], [-74.3,-131],

... [-77.5,-163], [-77.4, 163], [-71.7, 172], [-65.9, 140], [-65.7, 113],

... [-66.6, 88], [-66.9, 59], [-69.8, 25], [-70.0, -4], [-71.0, -14],

... [-77.3, -33], [-77.9, -46], [-74.7, -61]

... ]

>>> for pnt in antarctica:

... p.AddPoint(pnt[0], pnt[1])

...

>>> num, perim, area = p.Compute()

>>> print "Perimeter/area of Antarctica are {:.3f} m / {:.1f} m^2".format(

... perim, area)

Perimeter/area of Antarctica are 16831067.893 m / 13662703680020.1 m^2