Geodesic Spatial Operators on the ellipsoid

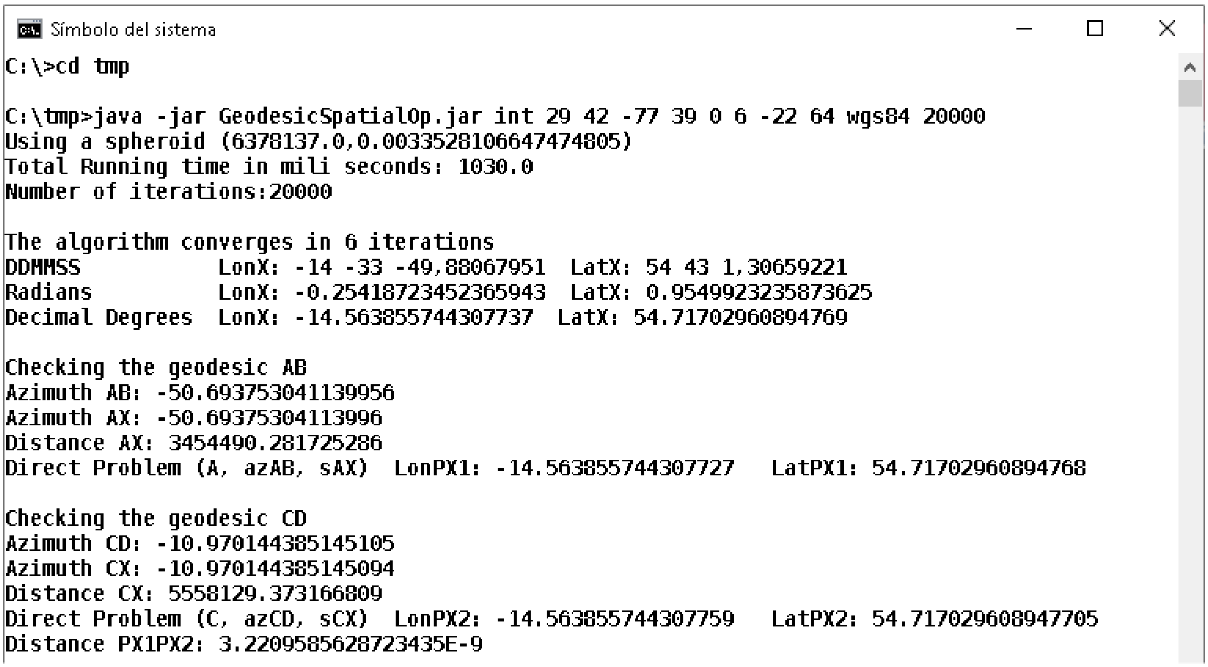
Two implementations (in Java and PlpgSQL for PostGIS) to obtain the intersection of two geodesic lines and the minimun distance from a point to a geodesic line with a high accuracy (better than 100nm), supporting long distances (greater than 180 degrees).

The research is published in this paper: (write the paper reference when it is published). The directory *journal_data* contains all the data from the tests and results calculated for the paper.

- We prove that the most powerful spatial analysis software products (from Oracle DBMS, Google, ESRI, etc.) make large errors when performing some spatial operators on the spheroid.
- We prove that these errors range from centimeters to hundreds of meters.
- We have designed and implemented (in a spatial database) two algorithms that calculate these spatial operators on the spheroid with high accuracy.
- We prove that the accuracy of these new spatial operators working on the spheroid are better than 100 nm.
- You do not have to worry anymore about choosing the best-fit local projection to analysis your spatial data.

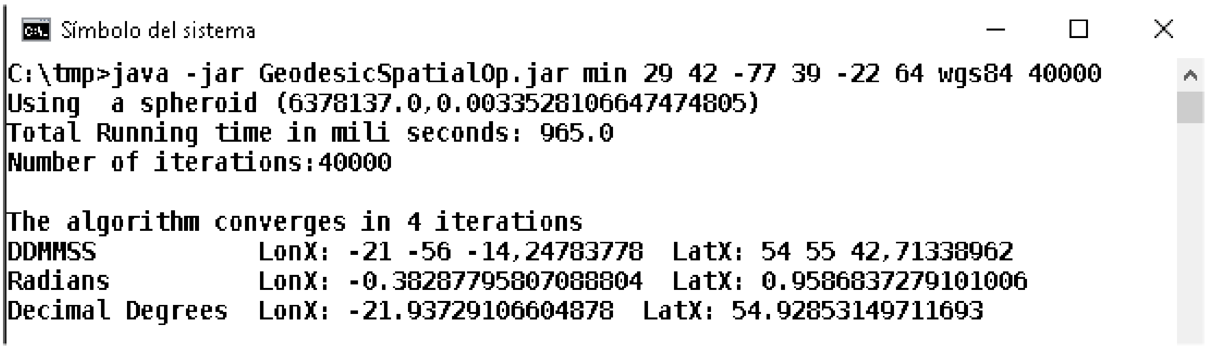
The file *GeodesicSpatialOp.Java* contains the Java implementation, and the file *GeodesicSpatialOp.sql* the Pl/pgSQL implementation for PostGIS.

You can try the geodesic intersection with the Java implementation *GeodesicIntersection.jar* from a shell like (20000 executions per second in a Core i7 4771 processor):



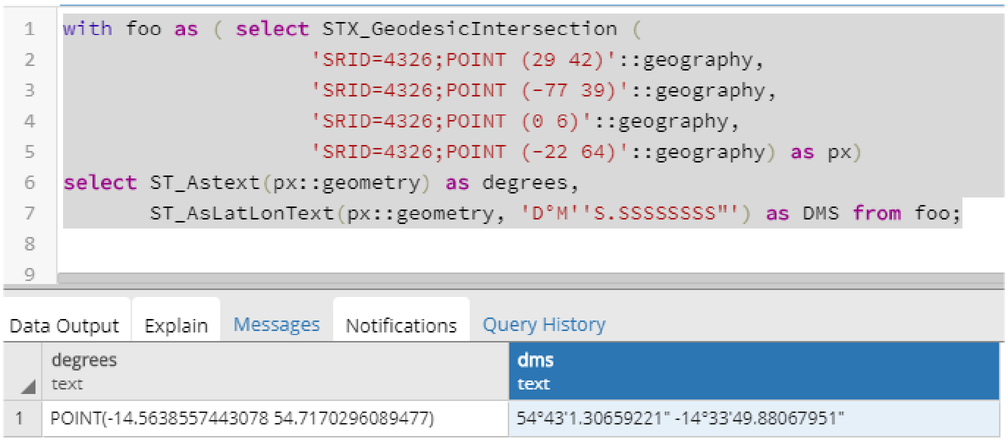
```
C:\>cd tmp

C:\tmp>java -jar GeodesicSpatialOp.jar int 29 42 -77 39 0 6 -22 64 wgs84 20000
Using a spheroid (6378137.0,0.0033528106647474805)
Total Running time in mili seconds: 1030.0
Number of iterations:20000

The algorithm converges in 6 iterations
DDMMSS           LonX: -14 -33 -49,88067951  LatX: 54 43 1,30659221
Radians          LonX: -0.25418723452365943  LatX: 0.9549923235873625
Decimal Degrees  LonX: -14.563855744307737  LatX: 54.71702960894769

Checking the geodesic AB
Azimuth AB: -50.693753041139956
Azimuth AX: -50.69375304113996
Distance AX: 3454490.281725286
Direct Problem (A, azAB, sAX)  LonPX1: -14.563855744307727   LatPX1: 54.71702960894768

Checking the geodesic CD
Azimuth CD: -10.970144385145105
Azimuth CX: -10.970144385145094
Distance CX: 5558129.373166809
Direct Problem (C, azCD, sCX)  LonPX2: -14.563855744307759   LatPX2: 54.717029608947705
Distance PX1PX2: 3.2209585628723435E-9
```

and the minimum distance from a point to a geodesic line with (40000 executions per second):



```
C:\tmp>java -jar GeodesicSpatialOp.jar min 29 42 -77 39 -22 64 wgs84 40000
Using  a spheroid (6378137.0,0.0033528106647474805)
Total Running time in mili seconds: 965.0
Number of iterations:40000

The algorithm converges in 4 iterations
DDMMSS           LonX: -21 -56 -14,24783778  LatX: 54 55 42,71338962
Radians          LonX: -0.38287795807088804  LatX: 0.9586837279101006
Decimal Degrees  LonX: -21.93729106604878  LatX: 54.92853149711693
```

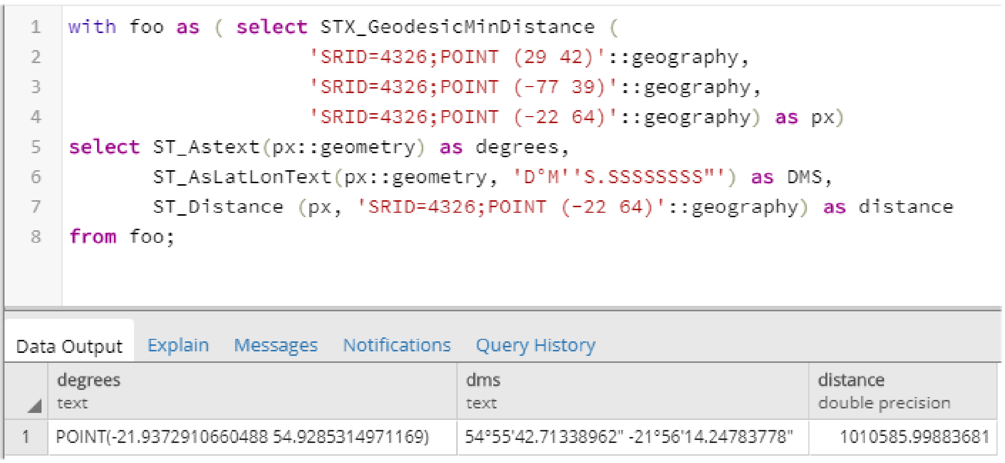
To use the new spatial operators with PostGIS, run the file *src/GeodesicSpatialOp.sql* in a PostGIS database. The SQL method *STX_GeodesicIntersection* returns the intersection point of the two geodesics in PostGIS:



```sql
with foo as ( select STX_GeodesicIntersection (
                     'SRID=4326;POINT (29 42)'::geography,
                     'SRID=4326;POINT (-77 39)'::geography,
                     'SRID=4326;POINT (0 6)'::geography,
                     'SRID=4326;POINT (-22 64)'::geography) as px)
select ST_Astext(px::geometry) as degrees,
       ST_AsLatLonText(px::geometry, 'D°M''S.SSSSSSSS"') as DMS from foo;
```

	degrees text	dms text
1	POINT(-14.5638557443078 54.7170296089477)	54°43'1.30659221" -14°33'49.88067951"

The SQL method *STX_GeodesicMinDistance* returns the closest point on the geodesic to a third point:



```sql
with foo as ( select STX_GeodesicMinDistance (
                     'SRID=4326;POINT (29 42)'::geography,
                     'SRID=4326;POINT (-77 39)'::geography,
                     'SRID=4326;POINT (-22 64)'::geography) as px)
select ST_Astext(px::geometry) as degrees,
       ST_AsLatLonText(px::geometry, 'D°M''S.SSSSSSSS"') as DMS,
       ST_Distance (px, 'SRID=4326;POINT (-22 64)'::geography) as distance
from foo;
```

	degrees text	dms text	distance double precision
1	POINT(-21.9372910660488 54.9285314971169)	54°55'42.71338962" -21°56'14.24783778"	1010585.99883681