Distance calculations

Let point 1 be latitude $\phi_1$and longitude λ_1, and point 2 be latitude ϕ_2 and longitude λ_2. The mean radius of the earth is $R_e = 6{,}371{,}000$ meters (see https://rosettacode.org/wiki/Haversine_formula for discussion on mean earth radii). The distances between these two locations can be computed – with different precision – the following ways.

Note that all calculations require degrees be converted to radians. This is achieved by multiplying by π (3.1415) then dividing by 180. For example, 30 deg should be converted as $(30)(3.1415)/(180) = 0.5236$. If using a calculator, make sure it is in radian mode. By default, most calculators are in degree mode, which will cause errors.

All software and spreadsheets are automatically in radians mode, so one MUST convert angles from degrees to radians.

Equirectangular approximation

Let the average latitude be $\bar{\phi} = (\phi_1 + \phi_2)/2$.

The following formulation, though slightly less accurate, is sufficient for many applications of smaller distances, including map analysis. It's also a "well-conditioned" formulation which always produces a correct answer.

$$\Delta\phi = \phi_2 - \phi_1$$

$$\Delta\lambda = (\lambda_2 - \lambda_1)\cos\bar{\phi}$$

$$d = R_e\sqrt{(\Delta\phi)^2 + (\Delta\lambda)^2}$$

From this formulation, also note that the following distance unit conversions can be approximated as:

1 deg latitude ≈ 111 km=111000 meters

≈ 60 nautical miles

≈ 69 miles

1 deg longitude ≈ (111 km) $\cos\bar{\phi}$ = (111000 meters) $\cos\bar{\phi}$

≈ (60 nautical miles) $\cos\bar{\phi}$

≈ (69 miles) $\cos\bar{\phi}$

Great circle equation using spherical law of cosines equation

This calculates a section of a sphere along its diameter. A good explanation is at http://mathworld.wolfram.com/GreatCircle.html . This formulation works well for relatively large distances. It does not work well for small distances (i.e., < hundreds of meters) unless the computer is accurate to 15 significant figures of precision. For small distances, the $\cos^{-1}$ function loses all precision (the computing problem is not "well-conditioned"), and the answer will default to a distance of 0 meters! More information on this problem is below in the Appendix. Also, because the earth is an oblate spheroid, errors occur for very large distances. It can be used for many applications, but caution is advised. The formulation is:

$$d = R_e \cos^{-1}(\sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos[\lambda_2 - \lambda_1])$$

In Excel and in programming applications, the function statement for $\cos^{-1}$ is *acos*.

Great circle equation using Haversine formulation

Because the spherical law of cosines is not a "well-conditioned" formulation as discussed above, an alternative can be derived using trigonometric identities with better results called the Haversine formulation:

$$d = 2R_e \sin^{-1}\left(\sqrt{\sin[0.5(\phi_2 - \phi_1)]\sin[0.5(\phi_2 - \phi_1)] + \sin[0.5(\lambda_2 - \lambda_1)]\sin[0.5(\lambda_2 - \lambda_1)]\cos\phi_2\cos\phi_1}\right)$$

which can also be written as the following:

$$d = 2R_e \sin^{-1}\left(\sqrt{\{\sin[0.5(\phi_2 - \phi_1)]\}^2 + \{\sin[0.5(\lambda_2 - \lambda_1)]\}^2\cos\phi_2\cos\phi_1}\right)$$

but because many overlook the power of 2, I've provided the first equation as well. It is accurate up to 0.5%, and sufficient for most distance computations, unless accuracy < 1 meter is needed. In Excel and in programming applications, the function statement for $\sin^{-1}$ is *asin*.

Simplified Vincenty formulation

For very high precision (within 0.5 millimeters accuracy!), an iterative algorithm which computes the shortest geodesic distance on the surface of an ellipsoid is used known as the Vicenty formulation. For example, it is used for tracking on Android phones!

A "simplified" Vicenty formula exists requiring no iteration, though, and is written as:

$$numerator = \{\cos\phi_2 \sin[\lambda_2 - \lambda_1]\}^2 + \{\cos\phi_1 \sin\phi_2 - \sin\phi_1 \cos\phi_2 \cos[\lambda_2 - \lambda_1]\}^2$$

$$denominator = \sin\phi_1 \sin\phi_2 + \cos\phi_1 \cos\phi_2 \cos[\lambda_2 - \lambda_1]$$

$$d = R_e \tan^{-1}\frac{\sqrt{numerator}}{denominator}$$

Assuming a WGS-84 ellipsoid, this author uses is $R_e = 6{,}372.7954776$ meters, which is the ellipsoid quadratic mean as discussed at http://math.wikia.com/wiki/Ellipsoidal_quadratic_mean_radius . However, I could not find documentation for the best value in this simplified version.

Website information

More details about distance calculations are available at these websites:

http://williams.best.vwh.net/avform.htm

http://www.movable-type.co.uk/scripts/latlong.html

http://en.wikipedia.org/wiki/Great-circle_distance

https://rosettacode.org/wiki/Haversine_formula (includes codes to use Haversine)

Details on the Vicenty formulation:

https://www.movable-type.co.uk/scripts/latlong-vincenty.html

https://community.esri.com/groups/coordinate-reference-systems/blog/2017/10/11/vincenty-formula

https://stackoverflow.com/questions/38248046/is-the-haversine-formula-or-the-vincentys-formula-better-for-calculating-distan

https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Vincenty's_formulae.html

https://geographiclib.sourceforge.io/html/Fortran/ (FORTRAN code for Vicenty)

Simplified Vicenty formulation in Wiki (no reference for equation --- if anyone has source, please provide):

https://en.wikipedia.org/wiki/Great-circle_distance

There are also online calculators for distance. Here are three examples:

http://www.daftlogic.com/projects-google-maps-distance-calculator.htm

http://www.nhc.noaa.gov/gccalc.shtml (this site rounds off to 1 km, so be careful)

https://www.movable-type.co.uk/scripts/latlong.html

Appendix – Why the Law of Cosines fails and Haversine works at small distances: a lesson on well-conditioned algorithms

Adopted from the excellent discussion at:

https://gis.stackexchange.com/questions/4906/why-is-law-of-cosines-more-preferable-than-haversine-when-calculating-distance-b

The Law of Cosines equation can fail because it is not "well-conditioned." It's an issue of computer arithmetic, not mathematics.

Here are the basic facts to consider:

1. One radian on the earth spans almost 10^7 meters.
2. The cosine function for arguments x near 0 is approximately equal to $1 - x^2/2$.
3. Double-precision floating point has about 15 decimal digits of precision.

Points (2) and (3) imply that when x is around one meter, or 10^{-7} radians (point 1), almost all precision is lost: $1 - (10^{-7})^2 = 1 - 10^{-14}$ is a calculation in which the first 14 of the 15 significant digits all cancel, leaving just one digit to represent the result. Flipping this around (which is what the inverse cosine, "acos", does) means that **computing acos for angles that correspond to meter-length distances cannot be done with any meaningful accuracy.** (In certain bad cases the loss of precision gives a value where acos is not even defined, so the code will break down and give no answer, a nonsense answer, or crash the machine.) Similar considerations suggest one should avoid using the inverse cosine if distances less than a few hundred meters are involved, depending on how much precision one is willing to lose.

The role played by acos in the naive Law of Cosines formula is to convert an angle to a distance. The Haversine formulation uses trigonometric identities in which a small angle x is approximately equal to x itself. As a result, the inverse of a number, being approximately that number, is computed essentially with no loss in precision. This is why different versions of the Haversine formula (which also includes an atan2 version not shown here), although mathematically equivalent to the law of cosines formula, is **far superior** for small distances (on the order of 1 meter or less).