LIBERTAS MATHEMATICA, VOL. 5, 1985

A NEW ALGORITHM FOR THE COMPUTATION OF THE GEODETIC COORDINATES AS A FUNCTION OF EARTH-CENTERED EARTH-FIXED COORDINATES

Lawrence O. Lupash

Introduction

There are many applications which require computation of the geodetic coordinates as a functions of Earth-Centered Earth-Fixed (ECEF) coordinates in real-time (such as the NAVSTAR Global Positioning System). Therefore, it is highly desirable to find efficient algorithms which can be used in real-time applications. The intent of this paper is to present a new algorithm which is comparable in efficiency to the best described in the literature from the stand point of the number of elementary operations required per iteration. The proposed method does not require a square root evaluation within the iterative part of the algorithm.

Problem Formulation

It is well known that for an ellipsoid of revolution

$$x_e = (r+h)\cos\phi\cos\lambda \tag{1}$$

$$y_e = (r+h)\cos\phi\sin\lambda \tag{2}$$

$$z_e = [(1-e^2)r+h]\sin\phi \tag{3}$$

$$r = \frac{a}{(1-e^2\sin^2\phi)^{1/2}} \tag{4}$$

where, for a given point - see Figure 1 -,

x_e, y_e, z_e - ECEF coordinates

ϕ - geodetic latitude

λ - geodetic longitude (measured East from the Greenwich meridian)

h - altitude above reference ellipsoid measured along the normal passing through the point

r - East-West radius of curvature at surface

a - ellipsoidal equatorial radius (a = 6378135 m. based on WGS-72 model)

e^2 - square of the eccentricity of the reference ellipsoidal (e^2 = 0.006694317778 for the WGS-72 model)

We seek to determine the geodetic coordinates (ϕ,λ,h) as a function of ECEF coordinates (x_e,y_e,z_e).

To the author's best knowledge, there is at least one direct method[1] and several iterative methods (see Appendix) for the computation of geodetic coordinates as a function of ECEF coordinates. The direct method (unpublished in open literature)[1] requires three square root operations, two cube root operations, one inverse tangent evaluation, one sine function evaluation and about 15 multiplication/division operations. As the geodetic coordinates are difficult and inefficient to evaluate in closed form, they are usually computed by an iterative method based on a Newton-Raphson technique. In the next section a new iterative method is presented.

Derivation of the Algorithm

First, an iterative process for the computation of the radius of curvature, involving the quantity $v = r^2/a^2$, is derived. Then the altitude and latitude are computed as a function of v.

From (1) and (2) it follows that

$$\cos^2\phi = \frac{x_e^2 + y_e^2}{(r + h)^2} \tag{5}$$

and (4) can be written as

$$\sin^2\phi = \frac{r^2 - a^2}{e^2 r^2} \tag{6}$$

Thus from (5) and (6) we have

$$\frac{r^2 - a^2}{e^2 r^2} + \frac{x_e^2 + y_e^2}{(r + h)^2} = 1 \tag{7}$$

Substituting $\sin\phi$ from (3) into (6), the result is

$$\frac{z_e^2}{[(1-e^2)r + h]^2} = \frac{r^2 - a^2}{e^2 r^2} \tag{8}$$

Now introduce the notation

$$\frac{r + h}{a} = u, \quad \frac{r}{a} = t \geq 1, \quad \frac{r^2}{a^2} = v \geq 1 \tag{9}$$

and

$$\frac{x_e^2 + y_e^2}{a^2} = b, \quad \frac{z_e^2}{a^2} = c \tag{10}$$

Then by substituting (9) and (10) into (7) and (8) we obtain

$$\frac{b}{u^2} + \frac{t^2 - 1}{e^2 t^2} = 1 \tag{11}$$

and

$$\frac{c}{(u - e^2 t)^2} = \frac{t^2 - 1}{e^2 t^2} \tag{12}$$

We eliminate u between (11) and (12) and use the notation $t^2 = v$. After

elementary algebraic manipulations, the result is the following equation in v:

$$f(v) \triangleq a_0 v^4 + a_1 v^3 + a_2 v^2 + a_3 v + a_4 \tag{13}$$

where

$$\begin{aligned} a_0 &= e^4(e^2-1)^2 \\ a_1 &= 2e^2(e^2-1)(d-2b) \\ a_2 &= d^2 - 2e^2(e^2-1)f - 4e^2(3-2e^2)b \\ a_3 &= -2df - 4be^2(e^2-3) \\ a_4 &= f - 4be^2 \end{aligned} \tag{14}$$

and

$$\begin{aligned} d &= f - ce^2 - e^2(e^2-1) \\ f &= b + c + e^2 \end{aligned} \tag{15}$$

Equation (13) can be solved using a Newton-Raphson technique, for example

$$v_{k+1} = v_k - \frac{f(v_k)}{f'(v_k)}, \quad k = 0,1,2,... \tag{16}$$

where

v_0 - value obtained at the last iterative process (solution) or a best "guess" for v

and

$$\begin{aligned} f'(v) &= 4a_0 v^3 + 3a_1 v^2 + 2a_2 v + a_3 \\ &\triangleq b_0 v^3 + b_1 v^2 + b_2 v + b_3 \end{aligned} \tag{17}$$

Since for many applications (such as the Global Positioning System), $h \ll r$, a good initial value for v can be obtained by setting h = 0, i.e. the starting point is at the surface of the ellipsoid. By a simple computation, introducing h = 0 in eqs. (1) - (4), it follows that

$$v_0 = \frac{1}{1 - \frac{e^2 c}{(1-e^2)^2 b + c}} \tag{18}$$

With v determined, r is obtained from (9) as

$$r = a\sqrt{v} \tag{19}$$

Introducing $v = t^2$ in (11) it follows that

$$u^2 = \frac{be^2 v}{(e^2-1)v + 1} = \frac{(r+h)^2}{a^2} \tag{20}$$

and, by using (19), equation (20) becomes

$$h = r\left[-1 + \sqrt{\frac{be^2}{(e^2-1)v+1}}\right] \tag{21}$$

On the other hand, tanϕ can be expressed as a function of v by using eqs. (5) and (6). Thus

$$\phi = \pm\arctan\sqrt{\frac{v - 1}{(e^2-1)v + 1}} \tag{22}$$

Finally, the geodetic longitude can be determined from (1) and (2) as follows

$$\lambda = \arctan\frac{y_e}{x_e} \tag{23}$$

Remark: From (9) and (20) it can be shown that

$$1 \leq v < \frac{1}{1 - e^2} \tag{24}$$

In summary, the entire algorithm can be implemented as follows:

Algorithm A

0°. Given: x_e, y_e, z_e and a, e^2

ε = imposed accuracy

k_{max} = maximum number of iterations permitted

1°. Compute the following coefficients

b, c, d, f - see (10) and (15)

a_0, a_1, a_2, a_3, a_4 - see (14)

b_0, b_1, b_2, b_3 - see (17)

and initialize v_0 - see (18)

$k = 0$.

2°. Set $k \leftarrow k+1$.

3°. Compute v_k as a function of v_{k-1} - see (16).

4°. If $|(v_k - v_{k-1})/v_k| \leq \varepsilon$ go to 6°, else go to 5°.

5°. If $k < k_{max}$ go to 2°, else stop and write a message "the algorithm does not converge" or set a flag.

6°. Determine r as a function of v - see (19).

7°. Compute h from (21).

8°. Compute ϕ from (22).

9°, Compute λ from (23).

Numerical Considerations and Conclusions

Call the new iterative algorithm, presented in the previous section, algorithm A. It requires: 10 additions/subtractions and 9 multiplications/additions per iteration. The most efficient known algorithm (see Appendix), call it algorithm B, requires: 5 additions/subtractions, 4 multiplications/divisions and one square root evaluation per iteration. Note that algorithm A does not require a square root evaluation within the iterative part of the algorithm.

The algorithms were coded in FORTRAN, in double precision, and numerous examples were run. The following conclusions were observed:

- the execution times for algorithms A and B are about the same for identical imposed accuracy (by using the algorithms coded in VAX FORTRAN and run under VMS on a VAX-750).
- the difference between the number of iterations required, for a given example, by algorithms A and B are not greater than 2 (for an imposed accuracy between 10^{-5} and 10^{-10}); algorithm A requires less than or the same number of iterations as algorithms B.
- algorithm A has a poor sensitivity when r approaches the value of "a" (equatorial radius), i.e. v is very close to 1, which implies that the given point is very close to the equatorial plane (this corresponds to the case when $|z_e|$ is less that 0.1 km.). In order to eliminate this problem for the points very close to the equatorial plane ($z_e \approx 0$), a direct computation can be used, for example $h = \sqrt{x_e^2 + y_e^2} - a$, $\phi = 0$, $\lambda = \arctan(y_e/x_e)$.
- algorithm B has also a very poor sensitivity when the points are close to the poles ($\sqrt{x_e^2+y_e^2} \approx 0$). In order to deal with this case when $\sqrt{x_e^2 + y_e^2} \leq 10$ km.), a direct computation can be used, for example: $h = |z_e| - b$, $\phi = \pm 90^\circ$, $\lambda = \arctan(y_e/x_e)$.

In conclusion, algorithm A is definitely preferred over algorithm B when points are close to the poles.

REFERENCE

[1] Barbee, T. "Geodetic latitude of target point," Space Applications Corp., Unpublished Notes, 1979.

Appendix

One of the most efficient iterative algorithms known in the literature (based on a direct use of Newton-Raphson technique) can be implemented as follows:

Algorithm B

0°. Given: x_e, y_e, z_e and a, e^2

ε = imposed accuracy

k_{max} = maximum number of iterations permitted

1°. Initialize: $h = 0$

$R = a$

$k = 0$

2°. Set $k \leftarrow k + 1$

$h_{old} \leftarrow h$

3°. Compute

$$\tan\phi = \frac{z_e}{\sqrt{x_e^2+y_e^2}} \cdot \frac{R + h}{(1-e^2)R+h}$$

4°. Compute

$$\sin^2\phi = \frac{\tan^2\phi}{1 + \tan^2\phi}$$

5°. Compute

$$R = \frac{a}{\sqrt{1-e^2\sin^2\phi}}$$

6°. Compute altitude

$$h = \sqrt{x_e^2 + y_e^2} \cdot \sqrt{1 + \tan^2\phi} - R$$

7°. If $|h - h_{old}| \leq \varepsilon$ go to 9°, else go to 8°.

8°. If $k < k_{max}$ go to 2°, else stop and write a message "the algorithm does not converge" or set a flag.

9°. Compute geodetic latitude

$$\phi = \arctan(\tan\phi)$$

10°. Compute geodetic longitude

$$\lambda = \arctan\,(y_e/x_e).$$

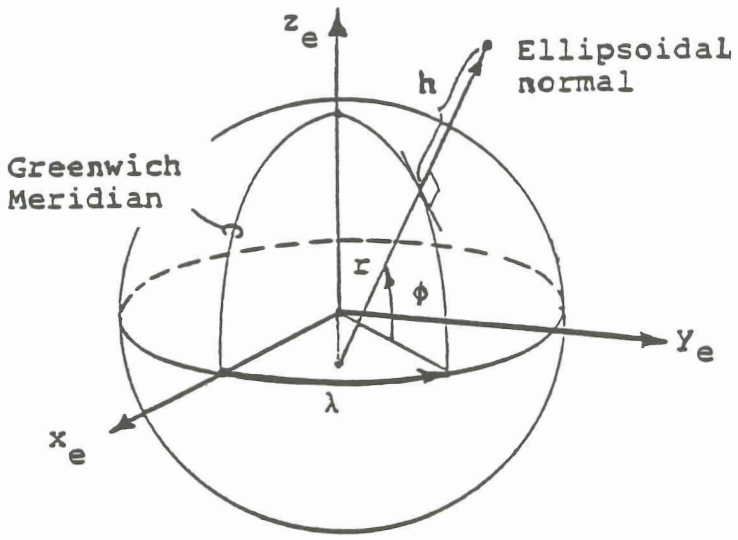

Figure 1. Geodetic parameters and ECEF frame.