Transformation from Cartesian to Geodetic Coordinates Accelerated by Halley's Method

Article *in* Journal of Geodesy · February 2006

DOI: 10.1007/s00190-006-0023-2

CITATIONS
59

READS
4,661

1 author:

Toshio Fukushima
National Astronomical Observatory of Japan
769 PUBLICATIONS **3,681** CITATIONS

SEE PROFILE

Transformation from geocentric rectangular to geodetic coordinates accelerated by Halley's method

Toshio Fukushima

National Astronomical Observatory, Ohsawa, Mitaka, Tokyo 181-8588, Japan

Toshio.Fukushima@nao.ac.jp

Received _____; accepted _____

To be submitted to Journal of Geodesy

ABSTRACT

By using Halley's third-order formula to find a root of a nonlinear equation, we developed a new iterative procedure to solve an irrational form of the latitude equation. By limiting the number of its iteration to one, starting from the zero height solution, and minimizing the number of division by means of the rational form representation of Halley's formula, we obtained a new non-iterative method to transform geocentric rectangular coordinates to geodetic ones. The new method is sufficiently precise in the sense that the maximum error in the transformed latitude is less than 2 μarcsecond. On the other hand, the new method runs roughly twice faster than the well-known Bowring's method (Bowring 1976) and does around 50% faster than our previous method (Fukushima 1999), which was the fastest method to do the geodetic coordinate transformation.

Subject headings: Geodetic coordinate transformation — Halley's method — Latitude equation

1. Introduction

The rise of GPS and other space geodetic techniques has enhanced the importance of the transformation between the classic and convenient geodetic coordinates (φ, λ, h) and the modern and straightforward geocentric rectangular coordinates (x, y, z). The forward transformation from the former to the latter is an easy task. However, the backward one, especially the conversion from the coordinates on a meridional plane $(p \equiv \sqrt{x^2 + y^2}, z)$ to (φ, h), is a complicated problem. Then the issue has been the target of many investigations. See some recent papers on this problem (Pollard 2002; Jones 2002; Vermeille 2002) and other articles listed in their references.

Once we presented an iterative method for the problem by using Newton's second-order formula to solve a nonlinear equation (Fukushima 1999). We numerically showed that it runs faster than the well-known Bowring's method (Bowring 1976) and some other methods known at the time such as Borkowski's closed-form solution of his quartic equation (Borkowski 1989) or the repeated usage of Bowring's formula (Laskowski 1991). Since then, some new methods were proposed (Pollard 2002; Jones 2002; Vermeille 2002). However, all of them are slower than our previous method. See Table 1, which compares an averaged CPU time of various procedures to do the core part of the backward transformation.

Is this the ultimate procedure to do the task? The answer is no. In this article, we report that usage of Halley's formula (Danby 1988, §6.6), which is known as an economic third-order method, significantly accelerates the backward transformation as illustrated in the table.

Table 1. Comparison of methods to transform (p, z) to (φ, h)

Reference	CPU Time	$\delta_{\rm MAX}$ (mas)	Equation Type	Variables	Solution	Convergence
Pollard (2002)	4.413	—	Vector	(h, z_0)	Linear	Linear
Jones (2002)	2.465	—	Trigonometric	ψ	Newton	Quadratic
Laskowski (1991)	1.843	—	Trigonometric	ψ	Newton	Quadratic
Borkowski (1989)	1.567	0.0066	Quartic	t	Ferrari	—
Vermeille (2002)	1.056	—	Quartic	k	Ferrari	—
Bowring (1976)	0.852	2.4	Trigonometric	ψ	Newton, once	Quadratic
Fukushima (1999)	0.667	—	Quartic	t	Newton	Quadratic
This work	0.426	0.0019	Irrational	(S, C)	Halley, once	Cubic

Note. — We compared some methods to transform a given pair of geocentric coordinates on a meridional plane (p, z) to the corresponding geodetic coordinates (φ, h). The items compared were the averaged CPU time, the maximum of transformation error, the type of equation, the choice of variables, and the method of solution. Here the unit of CPU time is μs for the Intel Pentium M 1.3 GHz processor and the unit of errors is milliarcsecond (mas), and the transformation error is defined as $\delta \equiv |\Delta\varphi| + |\Delta h|/(a + h)$. We omitted the errors when they are less than 1 nano arcsecond. The numerical values were obtained from the measurements in the double precision environment for a set of around 9.7×10^8 pairs of equally-spaced grid points in (φ, h), where the latitude was changed in the range $0 \le \varphi \le 90^\circ$ at every 10 arcsecond and the height was varied in the range -10 km $\le h \le 30000$ km at every 100 m. The CPU time used for preparing the constants such as $e_c \equiv \sqrt{1 - e^2}$ are excluded from the measurements. On the other hand, the CPU time used for selecting the starter and its computation was included. The programing language used in the measurements was the Compaq Visual Fortran 6.6. As for the variables adopted, z_0 is the z-coordinate of the foot on the reference ellipsoid, $\psi \equiv \tan^{-1}(e_c \tan\phi)$ is the reduced latitude, $t \equiv \tan(\pi/4 - \psi/2)$, $k \equiv h/N + e_c^2$, and S and C are a pair of quantities such that the tangent of the reduced latitude is expressed as $\tan\psi = S/C$.

2. Method

The main problem of the transformation from the geocentric rectangular coordinates to the geodetic coordinates is the solution of the latitude equation. We write it as an irrational equation[1] with respect to the tangent of the reduced latitude, $T \equiv \tan\psi$, as

$$g(T) \equiv pT - z_c - \frac{cT}{\sqrt{1+T^2}} = 0. \tag{1}$$

Here

$$z_c \equiv e_c|z|, \qquad c \equiv ae^2, \qquad e_c \equiv \sqrt{1-e^2}. \tag{2}$$

where a and e are the semi-major axis and the eccentricity of the reference ellipsoid.

We apply Halley's third-order method to the solution of the above nonlinear equation $g(T) = 0$. The resulting iterative formula becomes

$$T_{n+1} = T_n - \frac{g\left(T_n\right)}{g'\left(T_n\right) - g''\left(T_n\right) g\left(T_n\right) / \left(2g'\left(T_n\right)\right)}, \tag{3}$$

where the first and second derivatives of $g(T)$ are given as

$$g'(T) = p - \frac{c}{\left(\sqrt{1+T^2}\right)^3}, \tag{4}$$

and

$$g''(T) = \frac{3cT}{\left(\sqrt{1+T^2}\right)^5}, \tag{5}$$

respectively.

A naive application of Halley's formula would require several division operations, which are much more time-consuming than other arithmetic operations. Then, in order to minimize the number of division operations, we express the variable T in a fraction form,

---

[1]The derivation was given in Appendix B of Fukushima (1999).

$T = S/C$, and rewrite Halley's formula, which can be regarded as an iterative formula with respect to T, into a pair of iterative formulas in terms of S and C as

$$S_{n+1} = \left(z_c A_n^3 + c S_n^3\right) A_n^3 - B_n S_n, \tag{6}$$

$$C_{n+1} = \left(p A_n^3 - c C_n^3\right) A_n^3 - B_n C_n, \tag{7}$$

where

$$A_n = \sqrt{S_n^2 + C_n^2}, \tag{8}$$

$$B_n = \left(\frac{3}{2}\right) c S_n C_n \left[\left(p S_n - z_c C_n\right) A_n - c S_n C_n\right]. \tag{9}$$

Thus, no division appears during the iteration. Note that thus-computed S_n and C_n are not normalized in the sense $A_n \neq 1$.

As a starter for the iterative procedure, we adopt the solution in case $h = 0$, which is expressed in terms of S and C as

$$S_0 = z, \qquad C_0 = e_c p. \tag{10}$$

In order to accelerate the solution procedure, we limit the number of iterations to one. Then the geodetic coordinates are computed from the values after the first correction, S_1 and C_1, as

$$\varphi = \tan^{-1} \frac{S_1}{C_c}, \tag{11}$$

and

$$h = \frac{p C_c + S_0 S_1 - a\sqrt{e_c^2 S_1^2 + C_c^2}}{\sqrt{S_1^2 + C_c^2}}, \tag{12}$$

where

$$C_c = e_c C_1. \tag{13}$$

Note that one call of `atan`, three calls of `sqrt`, and two division operations are required in the total procedure from (p, z) to (φ, h). This smallness of the number of time-consuming operations contributes to the acceleration of the procedure.

3. Numerical Comparisons

In order to compare the cost and performance of the new method with those of the existing methods, we conducted test conversions from (p, z) to (φ, h) at around a billion of grid points of (p, z) ranging from the deepest sea bottom to well above the orbits of GPS/NAVSTAR and including the equatorial plane, and the polar axis. As for the parameters of test reference ellipsoid, a and $1/f$, we adopted those of the GRS 1980 system. The methods compared are (1) the first vector iteration method of Pollard (2002), which solves a pair of equations with respect to h and z_0, the z-coordinate of the foot on the reference ellipsoid, by using a linear iteration starting from the zero height solution, (2) Jones' method (Jones 2002), which solves a trigonometric equation with respect to the reduced latitude[2] by using Newton's iterative formula from a set of three stable starters, (3) the iteration of Bowring's formula twice at most (Laskowski 1991), (4) Borkowski's non-iterative method (Borkowski 1989), which solves a quartic equation with respect to $t \equiv \tan(\pi/4 - \psi/2)$ by Ferrari's method, (5) Vermeille's non-iterative method (Vermeille 2002), which solves a quartic equation with respect to $k \equiv h/N + e_c^2$ by Ferrari's method, (6) Bowring's method (Bowring 1976), which solves a trigonometric equation with respect to ψ by the Newton method whose application is limited to one, (7) our previous method (Fukushima 1999), which solve a variation of Borkowski's quartic equation by Newton's formula from a set of stable starters, and (8) the new method.

In the table, the methods are listed in the order of CPU time. The new method is the fastest. The slowness of Pollard's method is due to the linear convergence. The approach of Jones' method is essentially the same as our previous method. The difference in choice of the variable to be solved led to a large difference in CPU time. As is seen in the comparison of speed between Borkowski's method and our previous method, the repeated application

[2]He used the notation t.

of Newton's formula from a suitable starter is much faster than the single application of Ferrari's method in solving quartic equations derived from the latitude equation. In general the methods solving trigonometric equations are slower than the others because the evaluation of trigonometric functions are quite time-consuming. On the other hand, the methods solving quartic equations must be faster than those solving irrational equations in principle since the square root and the division run much slower than the evaluation of low order polynomials by Horner's method. However, the results of the new method betrayed this expectation. This owes to limiting the iteration to one and rewriting into a fraction form.

As for the precision, all the procedures but Borkowski's method, Bowring's formula, and the new method are accurate at the machine epsilon level. The reason for the inaccuracy of the last two methods is that they are truncated during the iteration for the sake of speed. On the other hand, Borkowski's method is imprecise because of an inappropriate definition of coefficients of his quartic equation. See discussion in Fukushima (1999). A proper treatment like Vermielle's method leads to errors at the machine epsilon level. From a practical viewpoint, however, both Borkowski's method and the new method are satisfactory in the sense that the maximum latitude error is less than 10 μarcseconds.

4. Conclusion

We presented a new method to compute geodetic coordinates from the given geocentric rectangular coordinates. The method solves an irrational form of the latitude equation with respect to the reduced latitude by a single application of Halley's third-order formula from the solution of zero height approximation. It runs around 50 % faster than our previous method (Fukushima 1999) and roughly twice faster than Bowring's method. Although the obtained solution by the new method is of approximate nature, it is sufficiently precise in

the sense that the maximum error in computed latitude is less than 2 μarcsecond for a very wide range of height covering the GPS/NAVSTAR orbits.

A Fortran code of the new method is presented in Appendix.

REFERENCES

Borkowski, K. M. (1989) Accurate Algorithms to Transform Geocentric to Geodetic Coordinates. Bulletin Géodésique: 63, 50–56

Bowring, B. R. (1976) Transformation from Spatial to Geographical Coordinates. Survey Review: XXIII, 181, 323–327

Danby, J. M. A. (1988) Fundamentals of Celestial Mechanics. Willmann-Bell, Richmond, VA

Fukushima, T. (1999) Fast transform from geocentric to geodetic coordinates. J Geodesy: 73, 603–610

Jones, G. C. (2002) New solutions for the geodetic coordinate transformation. J Geodesy: 76, 437–446

Laskowski, P. (1991) Is Newton's iteration faster than simple iteration for transformation between geocentric and geodetic coordinates? Bulletin Géodésique: 65, 14–17

Pollard, J. (2002) Iterative vector methods for computing geodetic latitude and height from rectangular coordinates. J Geodesy: 76, 36–40

Vermeille, H. (2002) Direct transformation from geocentric coordinates to geodetic coordinates. J Geodesy: 76, 451–454

APPENDIX

This manuscript was prepared with the AAS LaTeX macros v5.2.

A. Fortran code

```
subroutine gconv2(x,y,z,phi,lambda,h)
implicit none
real*8 x,y,z,phi,lambda,h
real*8 a,finv,aeps2,f,e2,ec2,ec,b,c,PIH
real*8 p,s0,zc,c0,c02,c03,s02,s03,a02,a0,a03
real*8 s1,c1,cs0c0,b0,cc,s12,cc2
logical first
data first/.TRUE./
if(first) then
  first=.FALSE.
  a=6378137.d0
  finv=298.257222101d0
  aeps2=a*a*1.d-32
  f=1.d0/finv
  e2=(2.d0-f)*f
  ec2=1.d0-e2
  ec=sqrt(ec2)
  b=a*ec
  c=a*e2
  PIH=2.d0*atan(1.d0)
endif
lambda=atan2(y,x)
s0=abs(z)
p2=x*x+y*y
if(p2.gt.aeps2) then
  p=sqrt(p2)
```

```
  zc=ec*s0
  c0=ec*p
  c02=c0*c0
  c03=c02*c0
  s02=s0*s0
  s03=s02*s0
  a02=c02+s02
  a0=sqrt(a02)
  a03=a02*a0
  s1=zc*a03+c*s03
  c1=p*a03-c*c03
  cs0c0=c*c0*s0
  b0=1.5d0*cs0c0*((p*s0-zc*c0)*a0-cs0c0)
  s1=s1*a03-b0*s0
  cc=ec*(c1*a03-b0*c0)
  phi=atan(s1/cc)
  s12=s1*s1
  cc2=cc*cc
  h=(p*cc+s0*s1-a*sqrt(ec2*s12+cc2))/sqrt(s12+cc2)
else
  phi=PIH
  h=s0-b
endif
if(z.lt.0.d0) then
  phi=-phi
endif
return
end
```