GEODESY AND CARTOGRAPHY
Vol. 60, No 2, 2011, pp. 145-159
DOI No 10.2478/v10277-012-0013-x

Conversion between Cartesian and geodetic coordinates on a rotational ellipsoid by solving a system of nonlinear equations

Marcin Ligas, Piotr Banasik

Department of Geomatics
AGH University of Science and Technology
30 Mickiewicza Al., 30-059 Krakow, Poland
e-mail: ligas@agh.edu.pl, pbanasik@agh.edu.pl

Received: 6 April 2011 /Accepted: 13 October 2011

Abstract: A new method to transform from Cartesian to geodetic coordinates is presented. It is based on the solution of a system of nonlinear equations with respect to the coordinates of the point projected onto the ellipsoid along the normal. Newton's method and a modification of Newton's method were applied to give third-order convergence. The method developed was compared to some well known iterative techniques. All methods were tested on three ellipsoidal height ranges: namely, (-10 – 10 km) (terrestrial), (20 – 1000 km), and (1000 – 36000 km) (satellite). One iteration of the presented method, implemented with the third-order convergence modified Newton's method, is necessary to obtain a satisfactory level of accuracy for the geodetic latitude ($\sigma_\varphi < 0.0004$") and height ($\sigma_h < 10^{-6}$ km, i.e. less than a millimetre) for all the heights tested. The method is slightly slower than the method of Fukushima (2006) and Fukushima's (1999) fast implementation of Bowring's (1976) method.

Keywords: Cartesian and geodetic coordinates, rotational ellipsoid, Newton's method, coordinate transformation

1. Introduction

Transformation from Cartesian coordinates (x, y, z) to geodetic (ellipsoidal) coordinates (φ, λ, h) is one of the basic tasks in computational geodesy. It is used for location of both ground objects as well as moving objects in outer space. The problem of the conversion from Cartesian to geodetic coordinates was discussed by many authors; their list may be found in Featherstone and Claessens (2008) and it still attracts interest. Solutions to the problem are either approximate or exact, characterized by different approaches: from transforming formulae (1) (Heiskanen and Moritz, 1967); introducing additional parameters, e.g. parametric latitude (Borkowski, 1987, 1989; Fukushima, 1999, 2006; Vermeille, 2002, 2004), minimizing the distance between a point in space and its projection on the ellipsoid (Hedgley, 1976) as well as vector methods (Lin and Wang, 1995; Feltens, 2009), and also proposing the use of h-geometry (Zanevicius and Kersys, 2010).

Formulae relating 3D Cartesian coordinates to geodetic coordinates may be expressed as follows.

For points inside/outside the ellipsoid

$$\begin{bmatrix} x_G \\ y_G \\ z_G \end{bmatrix} = \begin{bmatrix} (N+h)\cos\varphi\cos\lambda \\ (N+h)\cos\varphi\sin\lambda \\ \left[\left(1-e^2\right)N+h\right]\sin\varphi \end{bmatrix} \tag{1}$$

while for points on the surface of the ellipsoid

$$\begin{bmatrix} x_E \\ y_E \\ z_E \end{bmatrix} = \begin{bmatrix} N\cos\varphi\cos\lambda \\ N\cos\varphi\sin\lambda \\ \left[\left(1-e^2\right)N\right]\sin\varphi \end{bmatrix} \tag{2}$$

where

x_G, y_G, z_G – Cartesian coordinates of a point outside/inside the ellipsoid,
x_E, y_E, z_E – Cartesian coordinates of a point on the ellipsoid,
φ, λ, h – geodetic coordinates: latitude, longitude and ellipsoidal height, respectively,
a, b – semi-major and semi-minor axes of the ellipsoid, respectively,
N – radius of curvature in the prime vertical

$$N = \frac{a}{\sqrt{1-e^2\sin^2\varphi}} = \frac{a^2}{\sqrt{a^2\cos^2\varphi+b^2\sin^2\varphi}}$$

e^2 – first eccentricity squared

$$e^2 = 1-\left(\frac{b}{a}\right)^2$$

As may be seen from Eqs. (1) and (2) the transformation from geodetic to Cartesian coordinates is straightforward. On the other hand, the inverse transformation, which is the subject of this paper, is not so simple. Geodetic longitude is the exception and may simply be expressed as

$$\lambda = \arctan\frac{y_G}{x_G} \tag{3}$$

However, Vanicek and Krakiwsky (1982) and Vermeille (2004), respectively, give more stable variants for the longitude, i.e.

$$\lambda = 2\arctan\frac{y_G}{x_G+\sqrt{x_G^2+y_G^2}} \tag{4}$$

or

$$\lambda = \begin{cases} \dfrac{\pi}{2} - 2\arctan\dfrac{x_G}{\sqrt{x_G^2 + y_G^2} + y_G}, & \text{for} \quad y_G \geqslant 0 \\ -\dfrac{\pi}{2} + 2\arctan\dfrac{x_G}{\sqrt{x_G^2 + y_G^2} - y_G}, & \text{for} \quad y_G < 0 \end{cases} \tag{5}$$

A method of conversion presented in this paper is based on vector calculus and consists of two steps. The first step is to find the point on the surface of the ellipsoid (or the meridian ellipse) being the projection of a point h distance away (outside/inside/on) from the ellipsoid along the ellipsoidal surface normal (Fig. 1). The second step – computing the geodetic latitude and height – is straightforward. The proposed solution is most similar to that of Lin and Wang's (1995), but in this presentation the problem is solved with respect to coordinates of the point on the ellipsoid (or meridian ellipse that is a section of rotational ellipsoid through its axis of revolution) directly rather than for the parameter m (see Lin and Wang, 1995).

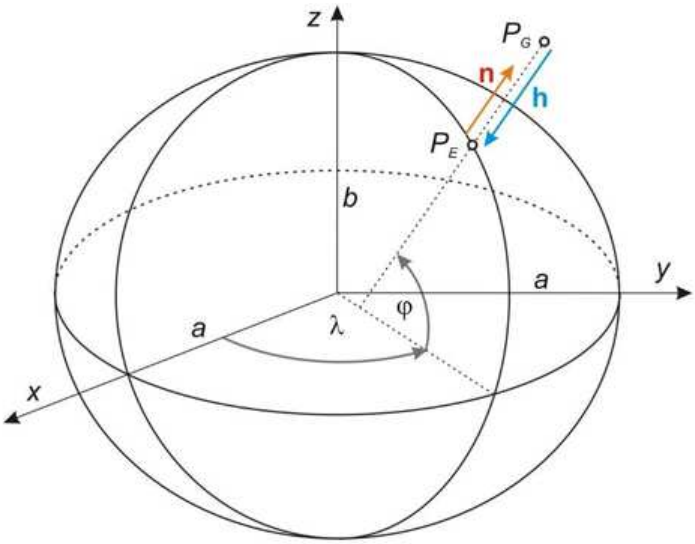

Fig. 1. Solution through co-linear vectors **n** and **h**

2. The first step – the projection of a point onto the ellipsoid

One of the possible solutions to the problem of coordinate transformation may be obtained by finding the projection of an external point $P_G = (x_G, y_G, z_G)$ onto the reference ellipsoid along the ellipsoidal surface normal, i.e. $P_E = (x_E, y_E, z_E)$ (Fig. 1). This may be achieved by constructing two co-linear vectors: the vector **n** normal to the ellipsoid (obtained from the gradient operator) in P_E which may be expressed as

$$\mathbf{n} = [n_1, n_2, n_3] = 2\left[\frac{x_E}{a^2}, \frac{y_E}{a^2}, \frac{z_E}{b^2}\right] \tag{6}$$

and the vector **h** connecting points P_G and P_E (Fig. 1)

$$\mathbf{h} = [h_1, h_2, h_3] = [x_E - x_G, y_E - y_G, z_E - z_G] \tag{7}$$

From the basics of vector calculus it is known that coordinates of the co-linear vectors are proportional with the constant factor k, thus

$$k = \frac{h_1}{n_1} = \frac{h_2}{n_2} = \frac{h_3}{n_3} \Rightarrow k = \frac{x_E - x_G}{Ax_E} = \frac{y_E - y_G}{Ay_E} = \frac{z_E - z_G}{Bz_E} \tag{8}$$

where $A = a^{-2}$ and $B = b^{-2}$.

In addition, the coordinates of P_E should satisfy the equation of the rotational ellipsoid, i.e.

$$\frac{x_E^2}{a^2} + \frac{y_E^2}{a^2} + \frac{z_E^2}{b^2} = 1 \tag{9}$$

Combining Eqs. (8) and (9), the systems of equations to be solved for $P_E = (x_E, y_E, z_E)$ are obtained. Although the above reasoning is strict in mathematical terms and fully applicable to the problem of conversion stated on a triaxial ellipsoid (Ligas, 2011), for the ellipsoid of revolution, it can be simplified by reducing the dimensions and solving the task on a meridian ellipse. Hence, Figure 1 simplifies to

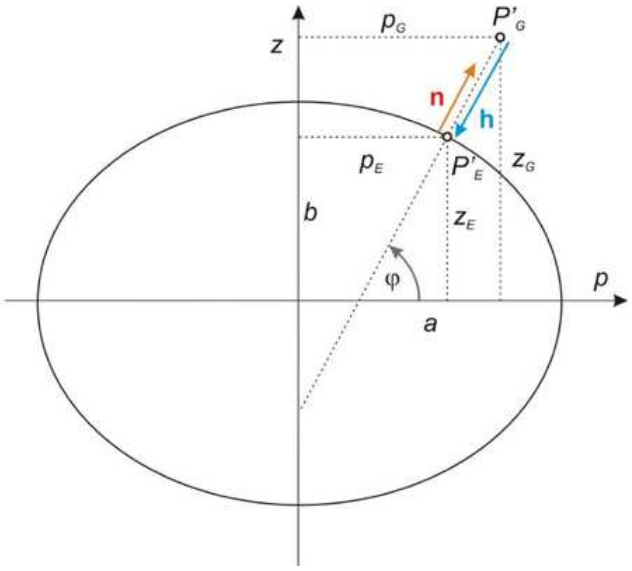

Fig. 2. Simplified conversion problem (meridian ellipse)

where $P'_G = (p_G, z_G)$, $P'_E = (p_E, z_E)$, $p_E = \sqrt{x_E^2 + y_E^2}$, $p_G = \sqrt{x_G^2 + y_G^2}$ and

$$\begin{cases} p_E = N \cos\varphi \\ z_E = \dfrac{b^2}{a^2} N \sin\varphi \end{cases} \tag{10}$$

In this simplified variant, the normal vector to the meridian ellipse **n** in point P'_E is given by

$$\mathbf{n} = [n_1, n_2] = 2\left[\frac{p_E}{a^2}, \frac{z_E}{b^2}\right] = \frac{2}{ab}\left[\frac{b}{a}p_E, \frac{a}{b}z_E\right] = \frac{2}{K}\left[Gp_E, Hz_E\right] \tag{11}$$

where $G = b/a$, $H = a/b$, $K = ab$, and the vector **h** connecting points P'_G and P'_E is

$$\mathbf{h} = [h_1, h_2] = [p_E - p_G, z_E - z_G] \tag{12}$$

The two vectors are required to be co-linear, hence the two-dimensional version of Eq. (8) may be written as

$$k = \frac{h_1}{n_1} = \frac{h_2}{n_2} = \frac{p_E - p_G}{Gp_E} = \frac{z_E - z_G}{Hz_E} \rightarrow Hz_E\,(p_E - p_G) = Gp_E\,(z_E - z_G) \tag{13}$$

and this makes the first equation of the system.

Similarly to the three-dimensional case, here the point P'_E is required to lie on the meridian ellipse, thus to satisfy the meridian ellipse equation

$$\frac{p_E^2}{a^2} + \frac{z_E^2}{b^2} = 1 \tag{14}$$

or with the already introduced constants G, H, K

$$Gp_E^2 + Hz_E^2 = K \tag{15}$$

This makes the second equation. Thus, the final version of the system of nonlinear equations to be solved is of the form

$$\begin{cases} f_1\,(p_E, z_E) = (p_E - p_G)\,Hz_E - (z_E - z_G)\,Gp_E = 0 \\ f_2\,(p_E, z_E) = Gp_E^2 + Hz_E^2 - K = 0 \end{cases} \tag{16}$$

3. Solution to the system of nonlinear equations – methods involved

To solve the system of nonlinear equations of the general form $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, two methods were applied. The first one – generalized Newton's method as a natural extension of classical Newton's method for nonlinear equations with one variable which is known to provide second order convergence. The second – modified Newton's method (Darvishi

and Barati, 2007) which gives third-order convergence for the price of an additional functions' evaluation. The iterative process for the two methods may be summarized as *Newton's method*

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{J}(\mathbf{x}_i)^{-1}\,\mathbf{f}(\mathbf{x}_i) \tag{17}$$

and *modified Newton's method* (Darvishi and Barati, 2007)

$$\begin{aligned} \mathbf{x}^*_{i+1} &= \mathbf{x}_i - \mathbf{J}(\mathbf{x}_i)^{-1}\,\mathbf{f}(\mathbf{x}_i) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i - \mathbf{J}(\mathbf{x}_i)^{-1}\left[\mathbf{f}(\mathbf{x}_i) + \mathbf{f}\left(\mathbf{x}^*_{i+1}\right)\right] \end{aligned} \tag{18}$$

where

J – Jacobian matrix (matrix consisting of the first partial derivatives),
x – the iterated solution to the system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$,
i – iteration number.

The solution based on Eq. (17) will be denoted as I and the one based on Eq. (18) as II. For the system of equations (16), which is a low dimensional one, the explicit form of the Jacobian matrix and its inverse may be expressed as ($x_1 = p_E$ and $x_2 = z_E$)

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} j_{11} & j_{12} \\ j_{21} & j_{22} \end{bmatrix} = \begin{bmatrix} Hz_E - (z_E - z_G)G & (p_E - p_G)H - Gp_E \\ 2Gp_E & 2Hz_E \end{bmatrix} \tag{19}$$

$$\mathbf{J}^{-1} = \frac{1}{|\mathbf{J}|}\begin{bmatrix} j_{22} & -j_{12} \\ -j_{21} & j_{11} \end{bmatrix} \tag{20}$$

$$|\mathbf{J}| = j_{11}j_{22} - j_{12}j_{21} \tag{21}$$

In order to solve the system of equations (16) by the methods defined by Eqs. (17) and (18), the first approximate solution to $\mathbf{x} = (p_E, z_E)$ is set to

$$p^0_E = ap_G r, \quad z^0_E = bz_G r \tag{22}$$

where

$$r = \frac{1}{\sqrt{p^2_G + z^2_G}}$$

For algorithm I, the number of iterations was limited to two and for algorithm II, only one iteration was executed. The consecutive steps in the main algorithm's loop will involve a small number of time-consuming arithmetic operations in favour of additions, subtractions and multiplications, which makes the solution relatively quick compared to evaluation of transcendental function. The possible weak points of the algorithm are the zero value of the determinant (singularity of Jacobian matrix) and the adopted starting point. Numerical tests performed showed the reliability of the algorithm for the majority points in space, with the exception of the region near the geocentre. This kind

of instability occurs in many algorithms, due to the fact that a point within a meridian ellipse (near the geocentre) of which Cartesian coordinates are to be transformed into geodetic coordinates lies on the intersection of two normals, and not suitably chosen initial point for the iteration may cause convergence to the wrong solution or even divergence (e.g. Zhang et al., 2005; Shu and Li, 2010).

4. The second step – computations of the latitude φ and height h

Geodetic latitude φ is obtained from Eqs (10), i.e.

$$\tan\varphi = \frac{\sin\varphi}{\cos\varphi} = \frac{\dfrac{a^2 z_E}{b^2 N}}{\dfrac{p_E}{N}} = \frac{a^2 z_E}{b^2 p_E} = H^2 \frac{z_E}{p_E} \tag{23}$$

and for the geodetic height h, the formula for the Euclidean distance between the points P'_E and P'_G is used

$$h = \sqrt{(p_E - p_G)^2 + (z_E - z_G)^2}, \quad \text{if} \quad (p_G + |z_G|) < (p_E + |z_E|) \quad \text{then} \quad h = -h \tag{24}$$

5. A brief review of compared iterative methods and details of coding

The above method has been compared to six previous iterative methods. They are

***Heiskanen and Moritz* (1967)**

$$\textit{Equation to be solved/iterated}: \begin{cases} h = \dfrac{p_G}{\cos\varphi} - N \\ \tan\varphi = \dfrac{(N+h)\, z_G}{\left(\dfrac{b^2}{a^2} N + h\right) p_G} \end{cases} \tag{25}$$

$$\textit{Initial guess}: \tan\varphi_0 = \frac{a^2 z_G}{b^2 p_G} \tag{26}$$

Method of solution: fixed point iteration

***Lin and Wang* (1995)**

$$\textit{Equation to be solved/iterated}: f(m) = \frac{p_G^2}{\left(a + \dfrac{2m}{a}\right)^2} + \frac{z_G^2}{\left(b + \dfrac{2m}{b}\right)^2} - 1 = 0 \tag{27}$$

$$\textit{Initial guess}: m_0 = \frac{ab\left(a^2 z_G^2 + b^2 p_G^2\right)^{\frac{3}{2}} - a^2 b^2 \left(a^2 z_G^2 + b^2 p_G^2\right)}{2\left(a^4 z_G^2 + b^4 p_G^2\right)} \tag{28}$$

Method of solution: Newton's iteration

Derivation of (φ, h): $\varphi = \arctan\left(\dfrac{a^2 z_E}{b^2 p_E}\right)$ (29)

h is obtained in the same manner as presented in Eq. (24), where

$$p_E = \frac{p_G}{1 + \dfrac{2m}{a^2}}, \quad z_E = \frac{z_G}{1 + \dfrac{2m}{b^2}}$$

Fukushima (1999)

Equation to be solved/iterated: $p_G t^4 + ut^3 + vt - p_G = 0$ (30)

Initial guess: $t = \dfrac{p_G - c + z'}{p_G - c + 2z'}$ (31)

where $t = \tan\left(\dfrac{\pi}{4} - \dfrac{\psi}{2}\right)$, ψ – reduced latitude, $u = 2(z' - c)$, $v = 2(z' + c)$, $c = ae^2$, $z' = \dfrac{b}{a} z_G$

Method of solution: Newton's iteration

Derivation of (φ, h): $\varphi = \arctan\left(\dfrac{a(1 - t^2)}{2bt}\right)$ (32)

$$h = \frac{2\frac{b}{a} p_G t + z_G(1 - t^2) - b(1 + t^2)}{\sqrt{(1 + t^2) - 4e^2 t^2}} \tag{33}$$

Borkowski (1989)

Equation to be solved/iterated: $2\sin(\psi - t) - g \sin 2\psi = 0$ (34)

Initial guess: $\tan\psi_0 = \dfrac{a z_G}{b p_G}$ (35)

where $t = \arctan\left(\dfrac{b z_G}{a p_G}\right)$, $g = \dfrac{a^2 - b^2}{\sqrt{(a p_G)^2 + (b z_G)^2}}$

Method of solution: Newton's iteration

Derivation of (φ, h): $\varphi = \arctan\left(\dfrac{a}{b}\tan\psi\right)$ (36)

$$h = (p_G - a\cos\psi)\cos\varphi + (z_G - b\sin\psi)\sin\varphi \tag{37}$$

Fukushima (2006)

Equation to be solved/iterated: $g(T) = PT - Z - \dfrac{ET}{\sqrt{1 + T^2}}$ (38)

Initial guess: $T_0 = \dfrac{|z_G|}{e_c p_G} = \dfrac{Z}{e_c^2 P}$ (39)

where $T = \tan\psi$, $P = \dfrac{p_G}{a}$, $Z = \dfrac{e_c |z_G|}{a}$, $E = e^2$, $e_c = \sqrt{1 - e^2}$

Method of solution: Halley iteration

Derivation of (φ, h): $\varphi = sign(z)\arctan\left(\frac{T}{e_c}\right)$ (40)

$$h = \frac{e_c p_G + |z_G| T - b\sqrt{1+T^2}}{\sqrt{e_c^2 + T^2}} \tag{41}$$

***Fukushima's fast implementation of Bowring's formula* (1999)**

Equation to be solved/iterated: $T = \frac{z' + cS^3}{p_G - cC^3}$ (42)

Initial guess: $T_0 = \frac{z_G}{e' p_G}$ (43)

where $C = \frac{1}{\sqrt{1+T^2}}$, $S = CT$, $e' = \sqrt{1-e^2}$, $c = ae^2$, $z' = e' z_G$

Method of solution: Newton's iteration

Derivation of (φ, h): $\varphi = \arctan\left(\frac{T}{e'}\right)$ (44)

$$h = \begin{cases} \frac{\sqrt{(1-e^2)+T^2}}{e'}\left(p_G - \frac{a}{\sqrt{1+T^2}}\right) & , \text{if } p_G > z_G \\ \sqrt{(1-e^2)+T^2}\left(\frac{z_G}{T} - \frac{b}{\sqrt{1+T^2}}\right) & \text{otherwise} \end{cases} \tag{45}$$

All presented methods have been coded in Borland Delphi 7 with double precision floating point arithmetic (extended type 10B in size, 19 – 20 significant digits) and run under Windows XP Professional operating system, on HP Pavilion notebook with AMD Athlon(tm) 64X2 Dual – Core Processor TK – 55, 1.80 GHz, 960 MB RAM. All constant values necessary to implement the particular method were declared only once at the beginning of a driver program and had no impact on the time of execution. Table 1 presents the number of most time consuming arithmetic operations used while coding the algorithms (for CPU times for each arithmetic operation, consult Fukushima (1999)).

6. Numerical results

A numerical comparison among the methods was carried out on the meridian ellipse (constant longitude set to $\lambda = 45°$) of the GRS80 reference ellipsoid (Moritz, 1980). The methods have been tested over three geodetic height ranges. The first range of heights varying from -10 km to 10 km (terrestrial) with the step of 0.5 km (73 841 points), the second range of heights from 20 km to 1000 km with the step of 10 km (178 299 points) and the third range of heights from 1000 km to 36 000 km (satellite) with the step of 100 km (632 151 points).

Table 1. The use of arithmetic operations in transformation from (x, y, z) to (φ, h)

Method	Divisions	Square roots	Trigonometric
Heiskanen and Moritz (1967)	$1 + 4i$	$1 + 2i$	1
Borkowski (1989)	$3 + i$	$4 + 2i$	$4 + 2i$
Lin & Wang (1995)	$4 + 5i$	3	1
Fukushima (1999)	$2 + i$	2	1
Fukushima (2006)	$2 + 2i$	$3 + i$	1
Fast Bowring (Fukushima, 1999)	$2 + 2i$	$3 + i$	1
This work I	$2 + i$	3	1
This work II	$2 + i$	3	1

The comparative numerical procedure was performed in two steps. In the first step Cartesian coordinates were generated on the basis of known geodetic coordinates, for each ellipsoidal height with latitude varying from 0° to 90° every 0.05°. In the second step geodetic coordinates were recovered from Cartesian coordinates obtained in the first step. Along with the retransformed coordinates, the time of execution of each method and the obtained accuracy (defined as $\log_{10}$ from maximum absolute difference between initial (known) geodetic coordinates and retransformed coordinates for all latitudes and heights tested within a particular height range) were recorded. The time of execution of a particular method was rescaled (fraction with the time of execution of Fukushima's algorithm as the reference value, in the denominator) to the Fukushima's algorithm denoted in his work as (c) (Fukushima, 2006). The time of execution and the accuracy obtained were checked after each iteration for each of the tested methods. The maximum deviations (interpreted here as errors) of retransformed geodetic coordinates from the initial coordinates are presented in logarithmic scale (base 10). Maximum error in φ is expressed as $\log_{10}\{\text{abs}[\max(\varphi_t\text{-}\varphi_c)]\}$, and maximum error in h is expressed as $\log_{10}\{\text{abs}[\max(h_t - h_c)]\}$, where subscripts t and c denote initial values and recalculated ones, respectively. The iterations were performed until a method reached the "acceptable" level of accuracy; namely, error in $\varphi < 10^{-8}$ dec. deg. (less than ≈ 0.0004") and error in $h < 10^{-6}$ km (less than a millimetre).

The results presented in Table 2 and subsequent tables with respect to the CPU time seem to stay in contradiction with some results presented in Fukushima (2006). It can be explaned in terms of the CPU and the environment, i.e. the authors used Borland Delphi 7 while Fukushima used Compaq Visual Fortran 6.6B, hence the difference in handling floating point arithmetic. For the terrestrial range of heights (Table 2), the algorithm II requires one iteration in order to obtain a satisfactory level of accuracy for all practical purposes. On the other hand, algorithm I requires two iterations to reach an acceptable level of accuracy but this requires extra time which slows down the solution. For the terrestrial range of heights (Table 2), the preference goes to Fukushima's fast implementation of Bowring's formula. Fukushima's (2006)

method is placed right after with almost perfect numerical accuracy obtained with only one iteration. The same level of accuracy reveals the method of Lin and Wang (1995) with a slight increase of the time. Algorithm II is the fourth in order among the tested iterative algorithms.

Table 2. Comparison of the methods with respect of the time of execution and obtained accuracy for the height range -10 km $\leqslant h \leqslant$10 km

Method		No of iterations				
		1	2	3	4	5
***Heiskanen & Moritz* (1967)**	Rescaled CPU time	(0.72)	(1.05)	(1.39)	(1.73)	
	Max. error in φ	-5.88	-8.15	-10.39	-12.61	
	Max. error in h	-1.17	-3.34	-5.51	-7.68	
***Borkowski* (1989)**	Rescaled CPU time	(1.91)				
	Max. error in φ	-11.09				
	Max. error in h	-14.88				
***Lin & Wang* (1995)**	Rescaled CPU time	(1.13)				
	Max. error in φ	-14.88				
	Max. error in h	-14.75				
***Fukushima* (1999)**	Rescaled CPU time	(0.85)	(1.06)	(1.28)	(1.51)	(1.72)
	Max. error in φ	1.20	-0.02	-2.35	-6.91	-14.88
	Max. error in h	2.39	-0.05	-4.70	-13.79	-14.72
***Fukushima* (2006)**	Rescaled CPU time	(1.00)				
	Max. error in φ	-14.88				
	Max. error in h	-14.70				
***Fast Bowring* (Fukushima, 1999)**	Rescaled CPU time	(0.83)				
	Max. error in φ	-11.09				
	Max. error in h	-9.05				
This work I	Rescaled CPU time	(1.04)	(1.43)			
	Max. error in φ	-6.09	-12.71			
	Max. error in h	-2.04	-8.19			
This work II	Rescaled CPU time	(1.20)				
	Max. error in φ	-9.51				
	Max. error in h	-6.76				

For medium heights (Table 3), the preference goes to the Fukushima's (2006) algorithm. Bowring's algorithm requires additional iteration to obtain the acceptable level of accuracy (decrease in accuracy for $h \approx 350$ km) which contributes to the slower execution. Lin and Wang's method still preserves the scaled CPU time from the previous height range but now the decrease in accuracy in calculated h is visible. The two new presented algorithms for this range of heights keep the timing and the accuracy from the terrestrial range (Table 2).

Table 3. Comparison of the methods with respect of the time of execution and obtained accuracy for the height range 20 km $\leqslant h \leqslant$ 1000 km

Method		No of iterations				
		1	2	3	4	5
***Heiskanen & Moritz* (1967)**	Rescaled CPU time	(0.72)	(1.03)	(1.36)	(1.68)	(2.01)
	Max. error in φ	-4.01	-6.34	-8.64	-10.93	-13.20
	Max. error in h	0.83	-1.41	-3.64	-5.88	-8.11
***Borkowski* (1989)**	Rescaled CPU time	(1.85)	(2.33)			
	Max. error in φ	-7.30	-12.53			
	Max. error in h	-14.48	-14.84			
***Lin & Wang* (1995)**	Rescaled CPU time	(1.11)				
	Max. error in φ	-13.62				
	Max. error in h	-9.03				
***Fukushima* (1999)**	Rescaled CPU time	(0.85)	(1.05)	(1.25)	(1.47)	(1.66)
	Max. error in φ	1.2	-0.01	-2.34	-6.89	-14.88
	Max. error in h	2.45	0.02	-4.63	-13.73	-14.70
***Fukushima* (2006)**	Rescaled CPU time	(1.00)				
	Max. error in φ	-10.51				
	Max. error in h	-14.69				
***Fast Bowring* (Fukushima, 1999)**	Rescaled CPU time	(0.82)	(1.08)			
	Max. error in φ	-7.29	-14.88			
	Max. error in h	-5.17	-14.73			
This work I	Rescaled CPU time	(1.02)	(1.39)			
	Max. error in φ	-6.10	-12.73			
	Max. error in h	-2.05	-8.21			
This work II	Rescaled CPU time	(1.17)				
	Max. error in φ	-9.53				
	Max. error in h	-6.78				

For the satellite height range (Table 4), the situation changes slightly. Fukushima's algorithm is still the quickest, but there is a significant loss in the latitude recovery in comparison to the two previous ranges of heights (Tables 2 and 3). Two iterations of the fast implementation of Bowring's method are slightly slower than Fukushima's (2006), but they offer much better accuracy. The next in order are algorithms I and II, which offer the same time as before but reveal some increase in the accuracy in both φ and h. For all height ranges, the remaining methods are slower than Fukushima's (2006) and also slower than algorithm II. Although all the methods presented in the paper use an iterative scheme, for two of them the only one iteration is sufficient to obtain acceptable results for any practical purposes for the range of heights tested; namely, Fukushima's (2006) and the new presented algorithm.

Table 4. Comparison of the methods with respect of the time of execution and obtained accuracy for the height range from 1000 km $\leqslant h \leqslant$ 36000 km

Method		No of iterations				
		1	2	3	4	5
***Heiskanen & Moritz* (1967)**	Rescaled CPU time	(0.73)	(1.05)	(1.38)	(1.70)	(2.03)
	Max. error in φ	-3.68	-6.18	-8.57	-10.90	-13.19
	Max. error in h	2.38	-0.61	-3.31	-5.71	-8.03
***Borkowski* (1989)**	Rescaled CPU time	(1.87)	(2.34)			
	Max. error in φ	-6.49	-11.78			
	Max. error in h	-12.62	-14.05			
***Lin & Wang* (1995)**	Rescaled CPU time	(1.12)	(1.43)			
	Max. error in φ	-11.17	-14.87			
	Max. error in h	-5.85	-14.05			
***Fukushima* (1999)**	Rescaled CPU time	(0.86)	(1.06)	(1.27)	(1.49)	(1.68)
	Max. error in φ	1.21	-0.01	-2.31	-6.83	-14.87
	Max. error in h	3.22	0.81	-3.81	-12.84	-13.85
***Fukushima* (2006)**	Rescaled CPU time	(1.00)				
	Max. error in φ	-8.82				
	Max. error in h	-13.97				
***Fast Bowring* (Fukushima, 1999)**	Rescaled CPU time	(0.83)	(1.09)			
	Max. error in φ	-6.32	-14.87			
	Max. error in h	-3.58	-13.97			
This work I	Rescaled CPU time	(1.03)	(1.40)			
	Max. error in φ	-6.43	-12.75			
	Max. error in h	-2.32	-8.11			
This work II	Rescaled CPU time	(1.20)				
	Max. error in φ	-10.00				
	Max. error in h	-7.25				

7. Conclusions

In the paper, a new method of solving one of the basic tasks in computational geodesy – the conversion from Cartesian to geodetic coordinates has been presented. The method is based on a solution of nonlinear system of equations with respect to coordinates of a point on the surface of the ellipsoid. Numerical tests indicated that the method works accurately and reliably. Although it is not the fastest method among all compared, but it certainly is not the slowest one. In this way, the new method becomes another trace in the part of geodetic literature devoted to the problem of Cartesian to geodetic coordinate transformation.

Acknowledgments

The paper is the result of research on geospatial methods carried out within statutory research No 11.11.150.006 at the Department of Geomatics, AGH University of Science and Technology, Krakow.

References

Borkowski K.M., (1987): *Transformation of Geocentric to Geodetic Coordinates without Approximations*, Astrophys. Space Sci., 139, pp. 1–4.

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

Bowring B.R., (1976): *Transformation from spatial to geographical coordinates*, Survey Review, 23, pp. 323–327.

Darvishi M.T., Barati A., (2007): *A third-order Newton-type method to solve systems of nonlinear equations*, Applied Mathematics and Computation, 187(2), pp. 630–635.

Featherstone W.E., Claessens S.J., (2008): *Closed-form transformation between geodetic and ellipsoidal coordinates*, Studia geophysica et geodaetica, 52, pp. 1–18.

Feltens J., (2009): *Vector methods to compute azimuth, elevation, ellipsoidal normal and the Cartesian (X,Y,Z) to geodetic (φ, λ, h) transformation*, Journal of Geodesy, Vol. 82, pp. 493–504.

Fukushima T., (1999): *Fast transform from geocentric to geodetic coordinates*, Journal of Geodesy, Vol. 73, pp. 603–610.

Fukushima T., (2006) *Transformation from Cartesian to geodetic coordinates accelerated by Halley's method*, Journal of Geodesy, Vol. 79, pp. 689–693.

Hedgley D.R., (1976): *An exact transformation from geocentric to geodetic coordinates for nonzero altitudes*, NASA TR R – 458, Washington.

Heiskanen W.A., Moritz H., (1967): *Physical Geodesy*, W.H. Freeman and Company, San Francisco.

Ligas M., (2011): *Cartesian to geodetic coordinates conversion on a triaxial ellipsoid*, Journal of Geodesy, DOI: 10.1007/s00190-011-0514-7.

Lin K.C., Wang J., (1995): *Transformation from geocentric to geodetic coordinates using Newton's iteration*, Bulletin Géodésique, Vol. 69, pp. 300–303.

Moritz H., (1980): *Geodetic Reference System 1980*, Bulletin Géodésique, Vol. 54, pp. 395–405.

Shu Ch., Li F., (2010): *An iterative algorithm to compute geodetic coordinates*, Computers & Geosciences, Vol. 36, pp. 1145–1149.

Vanicek P., Krakiwsky E.J., (1982): *Geodesy: The concepts*, Elsevier, Amsterdam, The Netherlands.

Vermeille H., (2002): *Direct transformation from geocentric to geodetic coordinates*, Journal of Geodesy, Vol. 76, pp. 451–454.

Vermeille H., (2004): *Computing geodetic coordinates from geocentric coordinates,* Journal of Geodesy, Vol. 78, pp. 94–95.

Zanevicius D., Kersys F., (2010): *Technologies for calculating geodetic coordinates applying h-geometry functions*, Geodezija ir Kartografija, 36(4), Vilnius Gediminas Technical University, pp. 160–163.

Zhang C.D., Hsu H.T., Wu X.P., Li S.S., Wang Q.B., Chai A.Z., Du L., (2005): *An alternative algebraic algorithm to transform Cartesian to geodetic coordinates*, Journal of Geodesy, Vol. 79, pp. 413–420.

Transformacja współrzędnych kartezjańskich na geodezyjne na elipsoidzie obrotowej poprzez rozwiązanie układu równań nieliniowych

Marcin Ligas, Piotr Banasik

Katedra Geomatyki
Wydział Geodezji Górniczej i Ochrony Środowiska
Akademia Górniczo-Hutnicza
al. Mickiewicza 30, 30-059 Kraków
e-mail: ligas@agh.edu.pl, pbanasik@agh.edu.pl

Streszczenie

Artykuł przedstawia nową metodę transformacji między współrzędnymi kartezjańskimi a współrzędnymi geodezyjnymi na elipsoidzie obrotowej. Metoda polega na rozwiązaniu nieliniowego układu równań, w którym niewiadomymi są współrzędne punktu leżącego na powierzchni elipsoidy a będącego rzutem punktu znajdującego się poza elipsoidą wzdłuż normalnej. Tak wyznaczone współrzędne punktu na elipsoidzie są podstawą do obliczenia szerokości i wysokości geodezyjnej. Do rozwiązania układu równań zastosowano metodę Newtona oraz zmodyfikowaną metodę Newtona charakteryzującą się zbieżnością trzeciego rzędu. Nowa metoda została porównana z kilkoma dobrze znanymi rozwiązaniami iteracyjnymi. Wszystkie metody były testowane na trzech zakresach wysokości elipsoidalnych: -10 - 10 km (ziemski), 20 - 1000 km, 1000 - 36000 km (satelitarny). Zastosowanie zmodyfikowanej metody Newtona powoduje, iż jedna iteracja nowej metody wystarczy aby osiągnąć zadowalający poziom dokładności zarówno dla szerokości geodezyjnej, jak i wysokości. Prezentowana metoda jest nieco wolniejsza niż metoda Fukushimy (2006) oraz od szybkiej implementacji metody Bowringa (Fukushima, 1999).