Lukatela H. (2007): Koordinatni sustavi globusnih digitalnih zemljopisa, Ekscentar, no. 10, pp. 108-113

# Koordinatni sustavi globusnih digitalnih zemljopisa

*Hrvoje Lukatela*[*]

***SAŽETAK.*** *Normala na površinu elipsoida kao definicija položaja točke na njegovoj površini, numerički se najčešće izražava kutnom mjerom geografske širine i dužine. Umjesto toga, ista se geometrijska tvorevina numerički može izraziti u vektorskom obliku.*
*Kada se geodetska računanja izvode na digitalnim računalima, vektorski oblik normale ima dvije prednosti u usporedbi sa računanjem uz pomoć kutnih vrijednosti: izbjegava se relativno sporo izračunavanje trigonometrijskih funkcija, a programi postaju jednostavniji, jer polje numeričkih vrijednosti koordinata nema crta diskontinuiteta ili singularnih točaka. Skraćeni oblik takvih koordinata pokazuje vrlo dobar omjer širine podatka u memoriji ili na disku računala i prostorne rezolucije na planetarnoj površini.*

***KLJUČNE RIJEČI:*** *koordinatni sustavi, elipsoidna geometrija, digitalni zemljopisi, koordinatne slike, programiranje geodetskih računanja.*

***KLASIFIKACIJA*** *prema COBISS-u:* ***1.05***

## 1. Geodezija i digitalna računala

Malo se koja klasična tehnička znanost promijenila primjenom digitalnih računala tijekom proteklih nekoliko desetljeća kao geodezija. Tu su promjenu izazvali i nadalje je određuju dva čimbenika: jednoga bi mogli nazvati »vanjskim«, a drugoga »unutarnjim«. Vanjski je čimbenik sve veća potreba da se za informatičke sustave koji uključuju znanje o prostoru izmjeri, organizira i učini brzo pristupačnim stalno rastuća količina podataka o ljudskom okolišu, sa sve većom točnošću i pravodobnošću. Unutarnji je čimbenik ubrzano i stalno usavršavanje izvedbe geodetskih računanja, koje omogućuje neprekidni napredak u elektoničkoj računalnoj opremi.

U postupku je gradnje svakog digitalnog zemljopisa jedna od najvažnijih odluka ona o izboru kanoničkog načina prikazivanja položaja neke točke na Zemljinoj površini pomoću brojeva ili, još točnije, brojevima u obliku u kakvom ih je moguće pohranjivati i obrađivati u digitalnom računalu. Ovaj kratki članak prikazuje jedan od mogućih pristupa tom problemu. Zašto *problemu*? Zato jer izbor tog načina uvelike utječe na učinkovitost računalne opreme, koja pak postaje sve kritičnija povećavanjem količine podataka i brzinom kojom ih sustav mora obrađivati. Nesvrsishodno rješenje tog problema imat će teške posljedice po isplativost sustava, a promijeniti ga naknadno bit će rijetko praktično, često i sasvim nemoguće.

## 2. Koordinatni sustavi digitalnih zemljopisa

Ponajprije vrijedi razmotriti najvažnije korisne osobine koje izabrani način prikazivanja mora imati:

1. Odnos između brojeva koji definiraju položaj i točaka na Zemljinoj površini mora biti jednoznačan i mora pokrivati podjednakim matematičkim osobinama cijeli onaj dio Zemljine površine na kojem se nalaze podaci koje sustav obrađuje.

2. Prostor u memoriji sustava mora biti što bolje iskorišten. Očekujemo da će u idealnom slučaju broj jednoznačnih kombinacija koje se mogu prikazati onom širinom elementa računske memorije koja odgovara pohrani položaja jedne točke (recimo, 64 bita) biti jednak broju elementarnih površinskih čestica zemljine površine, dovoljno malenih da ih možemo, sumjerljivo potrebama točnosti sustava, aproksimirati *točkom* (za razliku od *površine*).

3. Brojevi trebaju biti pohranjeni u takvom obliku da se matematičke operacije računske geodezije mogu izvoditi na bar dva krajnja načina: nešto sporije, ali s maksimalnom točnošću koja će se od sustava zahtijevati, i brže, s minimalnom točnošću, u onim prilikama kada je opravdano smanjiti zahtjeve točnosti u zamjenu za povećanu brzinu obrade.

Vrijedi posebno upozoriti da među tim osobinama nema jedne koja je bila od velike važnosti kad su se problemi računske geodezije rješavali »ljudskom rukom«: potrebom da brojčana vrijednost položajnog podatka, u svojoj kanoničkoj formi, bude neposredno razumljiva korisniku informatičkog sustava. To nije potrebno zato jer će tek iznimno i u vrlo malom postotku točke koje sustav obrađuje biti (u brojčanom obliku, za razliku od njihovog slikovnog uobličenja na zemljovidu) predstavljene ljudskim očima, pa će, u onim rijetkim slučajevima kada to bude potrebno, pretvorba internog koordinatnog podatka (naprimjer, brojeva

[*] Hrvoje Lukatela, dipl. ing. geod., osobni web: http://lukatela.com/hrvoje/

o kakvima će biti riječ dalje u ovom prikazu) u onakav podatak kakav je korisnik navikao vidjeti (naprimjer, niz slova i brojeva koji pismom predstavljaju kutnu mjeru geografske širine i dužine) predstavljati sasvim zanemarivo opterećenje za sustav.

Ovaj prikaz također podrazumijeva da je problem izbora geodetskog datuma (elipsoida i elipsoidnih koordinata točaka osnovne mreže) riješen, pa ćemo, kada govorimo o »koordinatnom sustavu«, pretpostaviti da govorimo tek o načinu kako elipsoidne koordinate predstavljamo u računskom sustavu, a ne uključujemo razmatranja o tome kako se osnovna mreža, koja je sadržana u definiciji »geodetskog datuma«, odnosi spram svojeg utjelovljenja na zemljinoj površini. Ovdje nas zanimaju prvenstveno matematičke osobine elipsoidnih koordinata, stoga ćemo zanemariti pojmnovnu razliku između geodetskih i geografskih koordinata i u daljnjem tekstu upotrebljavati najširem krugu graditelja i korisnika sustava najpristupačniji termin: »geografske koordinate«.

## 3. Domena podataka

U tradicionalnoj je geodetskoj praksi izbor koordinatnog sustava skoro uvijek bio integralni dio projekta početne izmjere neke administrativne jedinice: dakle, strogo omeđene, relativno male (čak i u slučajevima cijelih država, »male« u usporedbi s cijelom Zemljom) površine. Ako se, međutim, problem izbora ne rješava za određeni administrativno-tehnički projekt, već za uporabu u nekom informatičkom sustavu koji će služiti velikom broju različitih digitalnih zemljopisa, na prizvoljno velikim i proizvoljno položenim prostorima, izbor koordinatnog sustava mora zadovoljavati i one slučajeve u kojima se prostorni podaci protežu diljem sveukupne površina planeta (Zemlje, a nerijetko već i njezinog satelita ili planetarnih susjeda). Premda svaka pojedinačna primjena takvog računskog sustava ne mora nužno obuhvatiti cjelokupnu planetarnu površinu, koordinate na osnovu kojih je informatički sustav izgrađen će morati odgovoriti takvom zahtjevu.

U ovom prikazu pretpostavljamo da je domena podataka cijela, neprekinuta planetarna površina i da koordinatni sustav mora biti primjenjiv, kako je prije spomenuto, *...s podjednakim matematičkim osobinama...*, na bilo kojem njezinom dijelu. Prije desetak godina su se počeli pojavljivti prvi veliki napudbeni proizvodi koji poštuju to načelo, a prvi je među njima bio Geodetic DataBlade, Infomix-ovog (danas IBM) Dynamic Server DB sustava. (http://tinyurl.com/ysgcms). Programi prostornih računanja toga sustava zasnovani su na programskoj biblioteci u C-jeziku, koja je sadržavala komponente o kojima je ovdje riječ. Produkt je originalno razvijen specifično za potrebe NASA-e, da bi ubrzo postao prvi komercijalno uspješni sustav za gradnju globusnih ili kontinentalnih prostornih baza podataka. Engleski je nadimak za takve prozvode, koji se sve više udomaćuje u geodetsko-informatičkoj praksi, »round-world GIS«, pa bismo ih mi mogli zvati, u možda nesto opširnijem, ali zato i preciznijem obliku, »globusnim digitalnim zemljopisima«.

## 4. Geografske koordinate

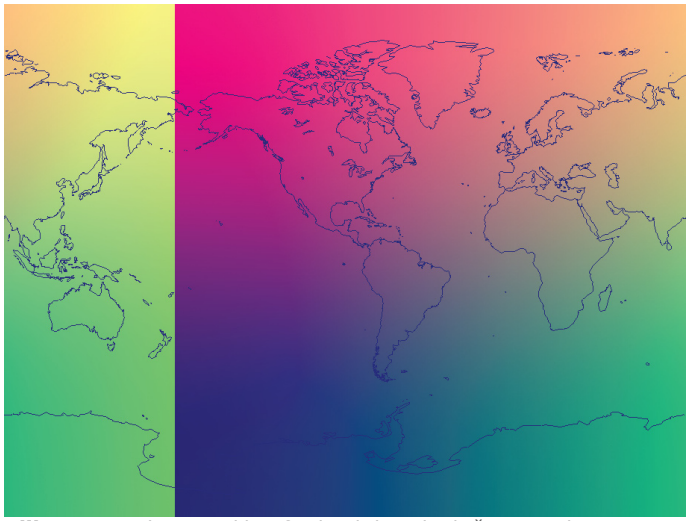
Sustav koordinata geografske dužine i širine vuče svoje korjene iz doba kada je tek počela sazrijevati spoznaja o pravom obliku Zemlje, a duboko je povezan sa prirodom oko nas. Geografska dužina odraz je, u kutnoj mjeri, određenog vremena koje je Zemljina kugla provela u svojoj vrtnji, a širina je naprosto prosječna godišnja mjera komplementa kuta, pod kojom Sunce sredinom dana obasjava dio njezine površine. Ta fundamentalna veza geografskih koordinata s našim poimanjem prirode oko nas čini ih intuitivno razumljivima, ne samo geodetskom stručnjaku - graditelju prostornog informatičkog sustava, nego i svakom njegovom potencijalnom korisniku, pa i »čistim informatičarima« koji u takvoj gradnji sudjeluju. Više nego bilo koja druga mjera ikada upotrebljavana u geodetskoj praksi, neusporedivo više nego lokalne ravninske koordinate različitih načina konformnog preslikavanja elipsoida u ravninu, geografske su koordinate ljudskoj mjeri prilagođen »prirodni način« brojevnog opisa položaja neke točke na Zemaljskoj površini.

Nije, stoga, začudjujuće da je tradicionalna kutna mjera geografske širine i dužine čest izbor koordinatnog sustava digitalnog zemljopisa, kada taj mora pokrivati cijelu Zemljinu površinu. Takav će izbor zadovoljiti jedan dio maloprije postavljenih kriterija, ali će se teško ogriješiti o neke druge. Za uočavanje problema koje sa sobom donosi upotreba kutne mjere geografskih koordinata, a i za razmatranje alternativnih rješenja, bit će korisno poslužiti se slikom, kako slijedi.

## 5. Slika polja koordinatnih promjena

Ako Zemljinu površinu preslikamo u ravninu, pa elementarjoj čestici podloge po kojoj crtamo pridodamo boju koja se intenzitetom svojih komponenti mijenja proporcionalno veličini koordinatnih brojeva odgovarajuće čestice Zemljine površine, dobit ćemo slike polja koordinatnih promjena (koje ćemo u daljnjem tekstu nazivati »koordinatnim slikama«). Prva takva koordinatna slika (Slika 1), u cilindričnoj projekciji, načinjena je tako da se boja mijenja od modrozelene do crvene kako geografska širina raste od -PI/2 do +PI/2, a istovremeno od ljubičaste do žutozelene, kako se geografska dužina mijenja od -PI do +PI. Kombinirana boja mijenja se tako bez ikakvog diskontiuiteta diljem cijelog dvodimenzionalnog polja

**Slika 1.** Koordinatna slika, fi - lambda, cilindrična projekcija 0, 0



**Slika 2.** Koordinatna slika, fi - lambda, cilindrična projekcija 0, -PI

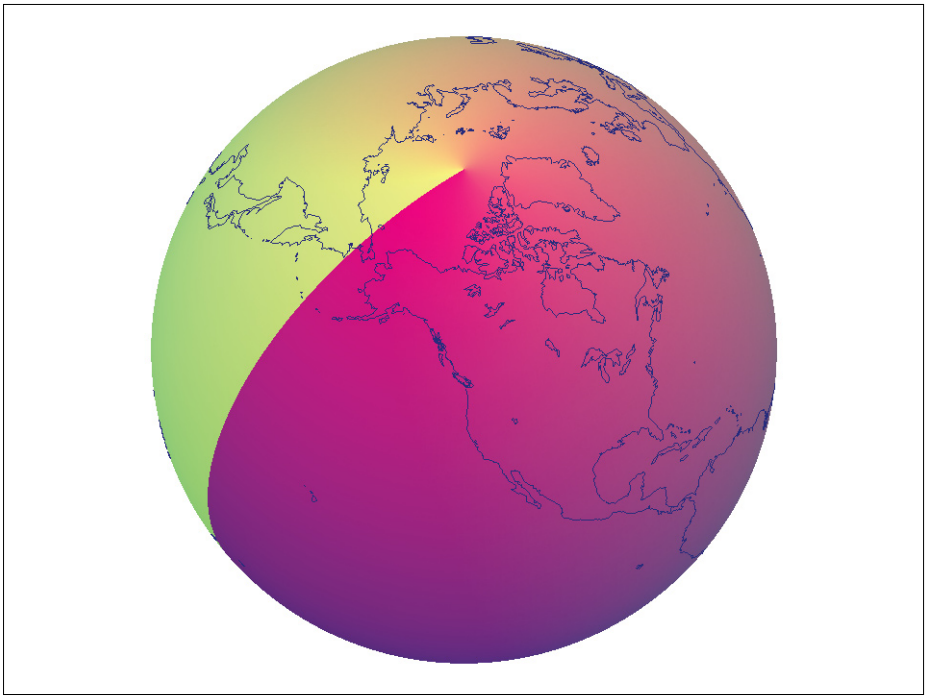
slike, pa bi se na prvi pogled moglo zaključiti da su i numerička svojstva koordinatnog sustava, odražena bojom na njegovoj slici, jednako tako kontinuirana diljem cijele domene podataka.

To, naravno, nije tako: dovoljno je pomaknuti centar cilindrične projekcije za PI/2 prema zapadu od početnog meridijana i ponoviti postupak (Slika 2), da bi se na slici pokazalo, da je meridijan antipodan početnom meridijanu linija diskontinuiteta kutne mjere geografske dužine.

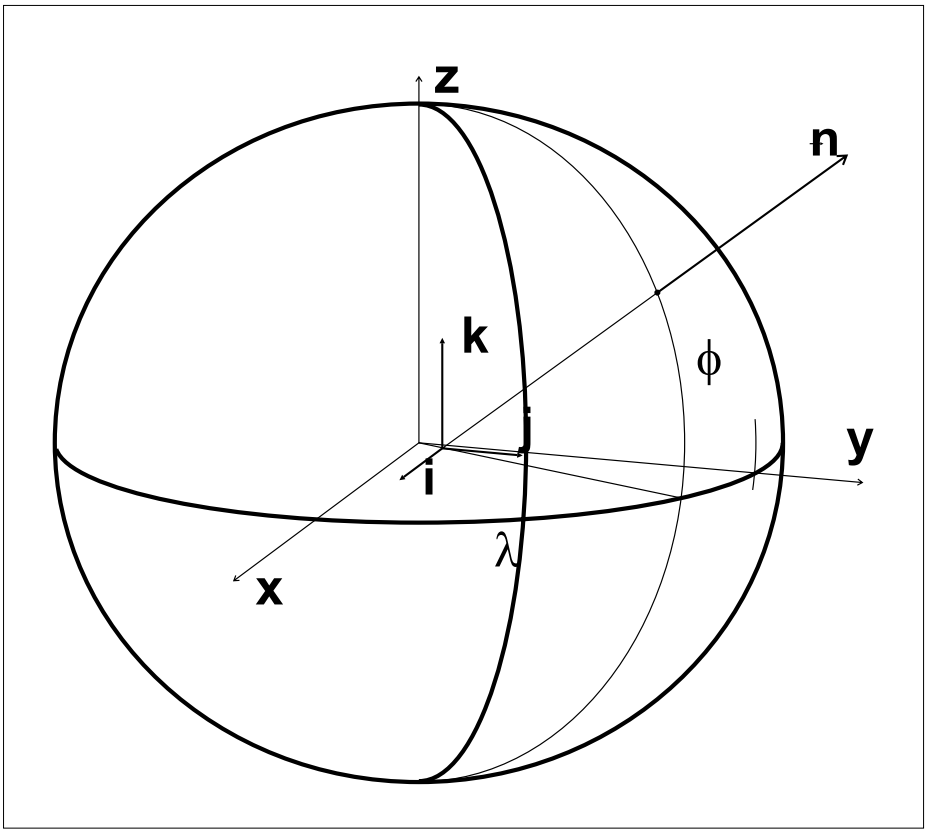
Ako pak koordinatnu sliku kutne mjere geografske širine i dužine izvedemo u ortografskoj projekciji (Slika 3), uočit ćemo nešto mnogo važnije: krajnje točke linije diskontinuiteta (sjeverni i južni pol) singularne su točke numeričke koordinatne domene, premda su te dvije točke u prirodi, po svim osobinama njihove prostorne geometrije, sasvim jednake bilo kojoj drugoj točki na globusu.

Uvesti u sustav singularne točke koje su uzrokovane samo načinom na koji mjerimo koordinate tamo, gdje ih u prirodnoj domeni podataka nema, bila bi velika pogreška: pri računanju bismo onda morali očekivati (i, naravno, riješiti) sve one probleme koji su tipični za numeričke metode koje obrađuju koordinatne brojeve u okolišu singulariteta. Dok će *računaru* (tj., čovjeku koji izvodi računanje) biti odmah očito treba li u kontekstu nekog projekta brinuti o koordinatnim diskontinuitetima i singularitetima, za *računalo* (elektronički uređaj) nema takvoga olakšanja, pa se svaki pojedini program, programska rutina ili algoritam, koji se ikada tijekom radnog vijeka nekog sustava može susresti s podacima u okolišu diskontinuiteta ili singulariteta, mora učiniti otpornim na numeričke nedaće koje se u takvom okolišu s pravom očekuju. Ugradivanje takve otpornosti je proces, koji ne samo da bitno poskupljuje gradnju napudbine, već i - što je često mnogo važnije - čini neusporedivo težim provjeru njezine točnosti i ispravnosti. Svaki dodatni posebni uvjet koji udvostručuje lokalnu lepezu programskog tijeka, doprinosi eksponencijanom povećanju broja globalno mogućih kombinacija programskog tijeka. Takve bi kombinacije trebalo, sve do posljednje, nezavisno ispitati u svakom sustavu koji se želi provjeriti do potpunosti. Već u mnogim jednostavnim programima će tako nešto biti teško izvedivo, dok će u kapitalnim napudbenim produktima to biti potpuno nemoguće, bez obzira na izdašnost proračuna ili raspoloživost vremena.

Uzgred možemo spomenuti još jednu veliku manu računanja kutnim mjerama geografske širine i dužine: skoro svaki računski postupak morat ćemo započeti iznalaženjem trigonometrijskih vrijednosti kut-



**Slika 3.** Koordinatna slika, φ - λ, ortografska projekcija




**Slika 4.** Normala na površinu rotacionog elipsoida

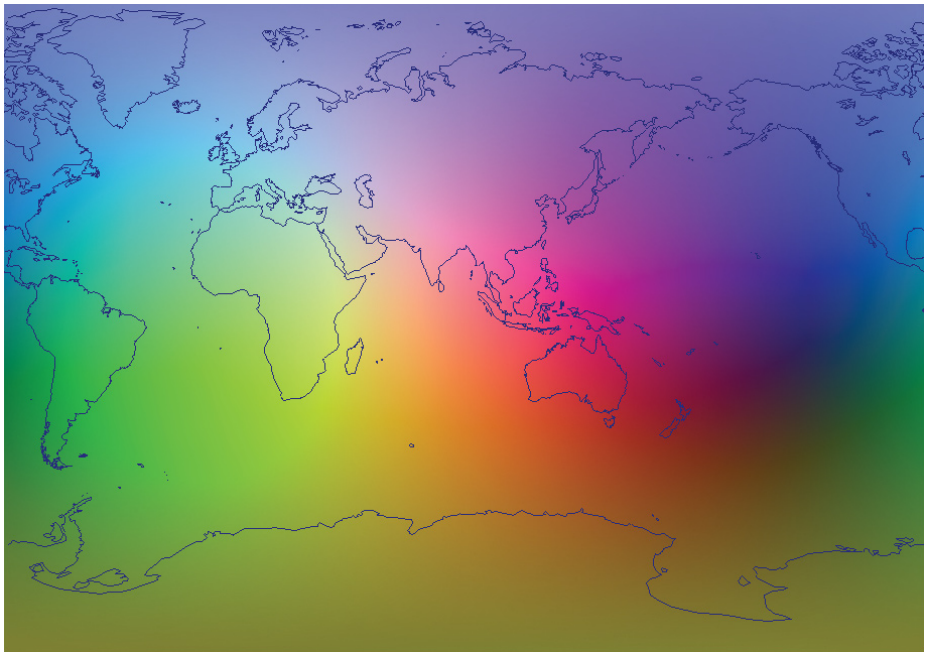
nih kordinata, a završiti računanjem kuta iz vrijednosti neke njegove trigonometrijske mjere. Transcendentalne su funkcije relativno skupa operacija, pa premda postaju, kao i algebarske operacije, sve brže u novijim generacijama digitalnih procesora, *odnos* između brzine algebarskih i trigonometrijskih operacija postaje sveudilj lošiji po ove druge, tako da ih u računanju treba izbjegavati kad god je to moguće.

Kutne geografske koordinate samo su jedan od načina da se brojevima opiše prostorna orjentacija normale koja definira položaj točke na površini rotacionog elipsida: ona može biti (Slika 4) jednako tako jednoznačno određena svojim normaliziranim vektorskim komponentama *i*, *j*, i *k*, kao i kutevima $\varphi$ i $\lambda$. Ako dakle prihvatimo namjesto kutne mjere ta tri broja kao koordinate točke na površini elipsida, riješit ćemo se svih maloprije spomenutih neprilika, uz tek jedan ozbiljan prigovor, kojem se, kako ćemo vidjeti kasnije, može naći prikladno rješenje.

Daleko najvažnija i u gradnji informatičkih sustava (za razliku od »računanja rukom«) najkorisnija osobina takvog koordinatnog sustava vidi se ako pogledamo

**Funkcije pretvaranja geografskih koordinata iz kutnog u vektorski oblik**

```
/* ========================================================================== */
/* Extracted from MLGP, a micro-library of geodetic primitives. hl, 2008/01. */

#include <math.h>

struct sDircos {       /* point on sphere or spheroid, direction cosine form */
   double dcx, dcy, dcz;           /* direction cosines for 3 cartesian axes */
   };
struct sLatLong {    /* point on sphere or spheroid, latitude/longitude form */
   double phi, lambda;                              /* latitude, longitude */
   };
#define GP_FUZZ_SQUARE  2.458e-14

struct cdc64 {           /* CDC (concise direction cosines), 4-byte integers */
   int u, v;                                        /* u: i or j; v: j or k */
   };
#define CDC64_UNDEF    -2147483647
/* ========================================================================== */
/* Stub, expected to be replaced with a compiler- and processor-specific     */
/* implementation that, presumably via assembler code, makes use of the      */
/* simultaneous derivation of sine and cosine offered on most FPU-s.         */
/* For instance, for GNU C compiler (gcc) and Intel 386+ FPU, use:           */
/* void gp_SineCosine(double a, double *s, double *c) {                   */
/*    asm("fsincos" : "=t" (*c), "=u" (*s) : "0" (a));                       */
/*    }                                                                      */
void gp_SineCosine(double angle,                  /* given angle in radians */
                   double *pSin,                        /* returned sine */
                   double *pCos) {                   /* returned cosine */
   *pSin = sin(angle);
   *pCos = cos(angle);
   }
/* ========================================================================== */
/* Given latitude and longitude, return vector form of the normal. note that */
/* input value is assumed to be well-formed, no data checking is performed.  */
void gp_LatLongToDircos(const struct sLatLong *pltln,      /* given lat/long */
                        struct sDircos *pdcos) {          /* returned vector */
   double cosPhi, cosLmbd, sinLmbd;
/* -------------------------------------------------------------------------- */
   gp_SineCosine(pltln->phi, &(pdcos->dcz), &cosPhi);
   gp_SineCosine(pltln->lambda, &sinLmbd, &cosLmbd);
   pdcos->dcx = cosPhi * cosLmbd;
   pdcos->dcy = cosPhi * sinLmbd;
   return;
   }
/* ========================================================================== */
/* Given vector form of the normal, return latitude and longitude. note that */
/* input value is assumed to be well-formed, no data checking is performed.  */
void gp_DircosToLatLong(const struct sDircos *pdcos,         /* given vector */
                        struct sLatLong *pltln) {       /* returned lat/long */
   double aux;
/* -------------------------------------------------------------------------- */
   aux = pdcos->dcx * pdcos->dcx + pdcos->dcy * pdcos->dcy;
   pltln->phi = atan2(pdcos->dcz, sqrt(aux));
   if (aux < GP_FUZZ_SQUARE) pltln->lambda = 0.0;
   else pltln->lambda = atan2(pdcos->dcy, pdcos->dcx);
   return;
   }
/* ========================================================================== */
```



**Slika 5.** Koordinatna slika vektorskog oblika normale, cilindrična projekcija

koordinatne slike normale izražene u obliku vektora: ne samo da je polje boja kontinuirano u cilindričnoj projekciji (Slika 5), nego je vidljivo da je polje boja bez ikakvih diskontinuiteta ili singulariteta, bez obzira kako i odakle promatramo površinu (Slike 6 i 7). Možemo, dakle, reći da je takav koordinatni sustav neprekidan i izometričan do one iste mjere do koje je neprekidna i izometrična domena podataka.

Funkcije koje pretvaraju geografske koordinate iz kutnog u vektorski oblik (i obrnuto) su krajnje jednostavne, pa ih ovdje uključujemo prvenstveno radi definicije programskih struktura i konstanti koje ćemo koristiti i u sljedećem odjeljku.

## 6. Skraćeni oblik vektorskih elipsoidnih koordinata

Prigovor koordinatnom sustavu koji smo izabrali očit je i jednostavan: ako za numerički opis položaja točke trebamo tri umjesto samo dva broja, povećat ćemo u jednakom omjeru količinu računalne memorije koja je potrebna za prikaz nekog određenog skupa prostornih objekata. Rješenje je, međutim, jednostavno, a nudi ga činjenica da je zbroj kvadrata komponenata normaliziranog vektora ravan jedinici. Vađenje drugog korijena na digitalnim procesorima je operacija tek nešto malo »skuplja« od dijeljenja (vremenski najskuplje od četiri osnovne algebarske operacije), a neusporedivo brža od izvođenja tigonometrijskih vrijednosti iz njihovih kuteva. Zato je kao koordinate točke svrsishodno zadržati samo dvije od tri vektorske komponente normale, a onu treću, kada je potrebna za izvođenje računanja, izvesti iz druge dvije. Taj će postupak biti donekle sličan izvođenju trigonometrijskih vrijednosti pri početku računanja s kutnom mjerom, ali s tom razlikom da je od takvog izvođenja neusporedivo brži. Jasno je, također, i koju od tri komponente treba »odbaciti« kada točku pohranjujemo: onu, čija će promjena u nekom određenom iznosu imati najmanji utjecaj na položaj točke, dakle onu koja ima najveću apsolutnu vrijednost. Koordinatna slika takvog »skraćenog« vektorskog oblika normale u cilindričnoj projekciji vidi se na slici 8, a u ortografskoj projekciji na slici 9. Ta su dva prikaza uključena samo radi boljeg poimanja skraćenog oblika vektorskih elipsoidnih koordinata: linije diskontinuiteta tog sustava nemaju utjecaja na računanja, jer se ona vrše u punom, »trokomponentnom« koordinatnom obliku.

Elemente skraćenog oblika vektorskih koordinata moguće je pohraniti u memoriji s različitom širinom zapisa, pa prema tome i s različitom rezolucijom na Zemljinoj površini. Ako, na primjer, pohranimo

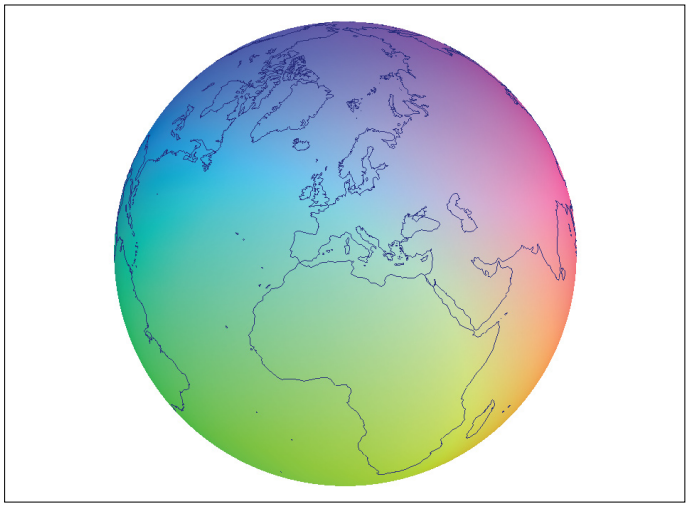
svaki od dva broja s onoliko znamenki koliko ih je moguće (uz potrebu da se zapiše i to koji je od tri broja ispušten i kojeg je on bio predznaka) pohraniti u 32-bitnom cjelobrojniku, zapisom točke u 64 bita ćemo postići prostornu rezoluciju od oko 15 milimetara u prirodi. Takav vrlo kompaktan oblik zapisa koordinata točaka neće dostajati za podatke geodetskih mreža ili detalja koji je određen terestičkom izmjerom visoke točnosti, ali će biti više nego dostatan za objekte kojima su koordinate određene, naprimjer GPS-om (uključujući i diferencijalni GPS), fotogrametrijskim metodama ili daljinskim opažanjima. Primjer programskog koda dviju C funkcija, koje u oba smjera pretvaraju elemente normale u takav zapis (vidi programski kod).

Zamjena klasičnih geodetskih računanja koja uptrebljavaju kutnu mjeru geografskih koordinata, s njihovim inačicama koje upotrebljavaju vektorski oblik normale na ellipsoid, nadasve je zanimljivo područje računske geodezije, ali prelazi granice ovog članka. Spomenut ćemo samo za usporedbu da je većina klasičnih računanja zasnovana na nekom postulatu diferencijalne geometrije na elipsoidnoj površini, koji se onda razvojem u red po rastućim potencijama elipsoidnog eksentriciteta dovodi do numeričkog rješenja s točnošću primjerenom problemu koji se rješava. Kada se pak računa normalama u vektorskom obliku, umjesto razvoja u redove je primjerenije koristiti iterativne algoritme, koji u svojem najčešćem obliku, u programskoj petlji, naizmjence u malim iznosima variraju položaj normale u točki u kojoj je ona najbliža koordinatnom ishodištu i onoga na elipsidnoj površini, a iteracija se zaustavlja kada su geometrijski uvjeti koji definiraju problem zadovoljeni - i opet - onakvom točnošću koja je primjerena problemu koji se rješava. No, dok smo u slučaju klasičnih računanja morali odabrati kriterij točnosti već pri razvoju programa, u slučaju iterativnih algoritama možemo

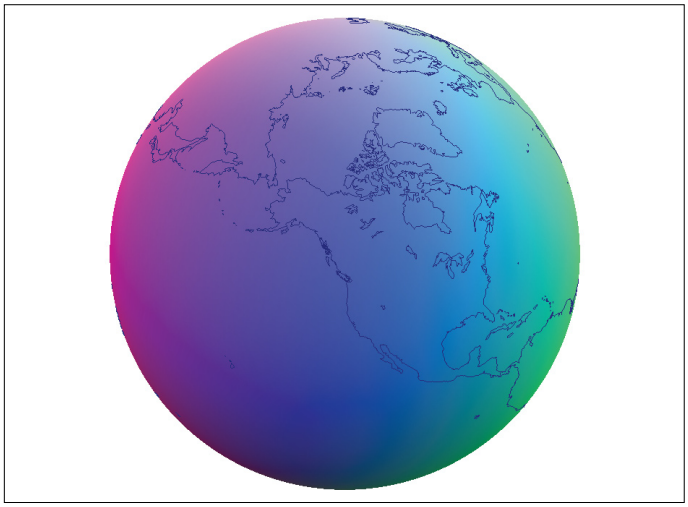
**Programski kod dviju C funkcija koje pretvaraju elemente normale**

```
/* ========================================================================== */
/* Extracted from MLGP, a micro-library of geodetic primitives. hl, 2008/01. */
/* ========================================================================== */
#define CDC64_SCALE 2147483640.0

#define M_BIT_1 0x80000000                                  /* high-order bit */
#define N_BIT_1 0x40000000                                  /* one next to it */
#define BITS_OA 0x3fffffff     /* all other: to force vacated high bits to 0 */

#define TRUNC_INT(a) (BITS_OA & (((int)(a * CDC64_SCALE))>>2)) /* int is I4! */
#define RSTR_DBL(i)  ((double)(((i)<<2) | 2) / CDC64_SCALE)

/* Given three-component vector form of spherical or spheroidal coordinates, */
/* return "concise" form, by dropping the component with the greatest        */
/* numerical value, scaling the magnitude of the other two so that they      */
/* will fit into a 32-bit integer each. Use the most-significant bits to     */
/* encode the order (i,j or k) and the sign of the dropped third component.  */
void gp_DircosToCdc64(const struct sDircos *vct, /* given vector coordinates */
                      struct cdc64 *cdc) {        /* returned "concise" form */
   double adi, adj, adk;
/* ------------------------------------------------------------------------- */
   adi = fabs(vct->dcx);
   adj = fabs(vct->dcy);
   adk = fabs(vct->dcz);

   if ((adi > adj) && (adi > adk)) {                                /* drop i */
      if (vct->dcx > 0.0) {
         cdc->u = TRUNC_INT(vct->dcy);
         cdc->v = TRUNC_INT(vct->dcz);
         }
      else {                                                /* swap to keep xOy RH */
         cdc->u = TRUNC_INT(-vct->dcy) | N_BIT_1;
         cdc->v = TRUNC_INT(vct->dcz);
         }
      }
   else if (adj > adk) {                                           /* drop j */
      if (vct->dcy > 0.0) {
         cdc->u = TRUNC_INT(-vct->dcx);
         cdc->v = TRUNC_INT(vct->dcz) | N_BIT_1;
         }
      else {                                                /* swap to keep xOy RH */
         cdc->u = TRUNC_INT(vct->dcx) | N_BIT_1;
         cdc->v = TRUNC_INT(vct->dcz) | N_BIT_1;
         }
      }
   else {                                                          /* drop k */
      if (vct->dcz > 0.0) {
         cdc->u = TRUNC_INT(vct->dcy);
         cdc->v = TRUNC_INT(-vct->dcx) | M_BIT_1;
         }
      else {                                                /* swap to keep xOy RH */
         cdc->u = TRUNC_INT(vct->dcy) | N_BIT_1;
         cdc->v = TRUNC_INT(vct->dcx) | M_BIT_1;
         }
      }
   return;
   }
```

*** nastavak na sljedećoj stranici ***



**Slika 6.** Koordinatna slika vektorskog oblika normale, ortografska projekcija (a)



**Slika 7.** Koordinatna slika vektorskog oblika normale, ortografska projekcija (b)

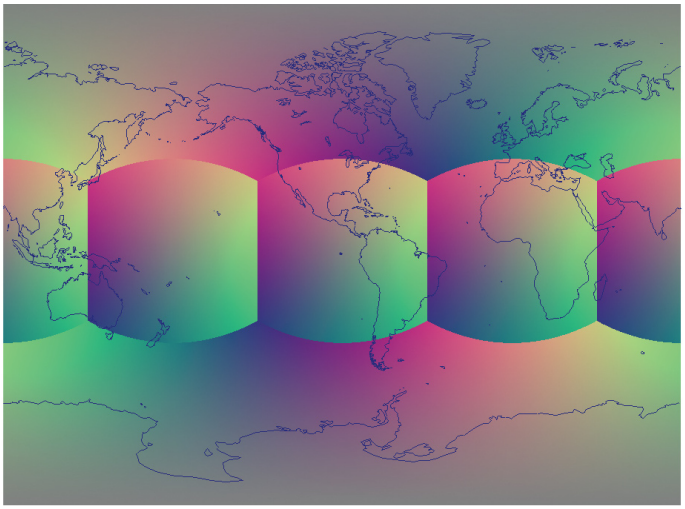
**Programski kod dviju C funkcija koje pretvaraju elemente normale**

```
/* ------------------------------------------------------------------------ */
/* Given concise form of spherical or spheroidal coordinates, return their  */
/* full vector form, suitable for computations. This funcitin is the inverse */
/* of gp_DircosToCdc64().                                                   */
void gp_Cdc64ToDircos(const struct cdc64 *cdc,         /* given concise form */
                      struct sDircos *vct) {  /* returned vector coordinates */
   double du, dv, dw;
/* ------------------------------------------------------------------------ */
   du = RSTR_DBL(cdc->u);
   dv = RSTR_DBL(cdc->v);
   dw = 1.0 - du * du - dv * dv;
   if (dw < GP_FUZZ_SQUARE) dw = 0.0;
   else dw = sqrt(dw);
   if (cdc->v & M_BIT_1) {                                   /* restoring k */
      if (cdc->u & N_BIT_1) {                /* flip kept, reverse computed */
         vct->dcx = dv;
         vct->dcy = du;
         vct->dcz = -dw;
         }
      else {
         vct->dcx = -dv;
         vct->dcy = du;
         vct->dcz = dw;
         }
      }
   else if (cdc->v & N_BIT_1) {                              /* restoring j */
      if (cdc->u & N_BIT_1) {                /* flip kept, reverse computed */
         vct->dcx = du;
         vct->dcy = -dw;
         vct->dcz = dv;
         }
      else {
         vct->dcx = -du;
         vct->dcy = dw;
         vct->dcz = dv;
         }
      }
   else {                                                    /* restoring i */
      if (cdc->u & N_BIT_1) {                /* flip kept, reverse computed */
         vct->dcx = -dw;
         vct->dcy = -du;
         vct->dcz = dv;
         }
      else {
         vct->dcx = dw;
         vct->dcy = du;
         vct->dcz = dv;
         }
      }
   return;
   }
/* ======================================================================== */
```

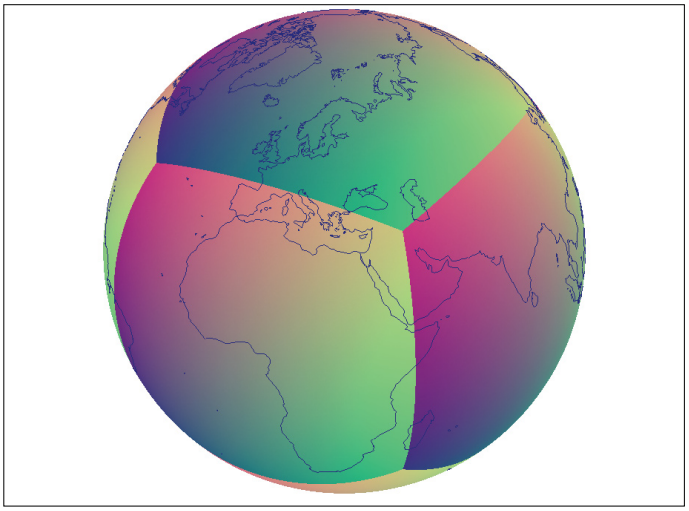
tu odluku odgoditi sve do trenutka izvođenja programa, kada je mnogo lakše odrediti svakom pojedinm problemu primjerenu ravnotežu između potrebne točnosti i vremena kojeg će provedba računa zahtijevati. Interesantno je spomenuti, da su već u doba prvih početaka primjene digitalnih računala u geodeziji, poneki autori upozoravali (npr. [Bomford, str. 593], već 1975.!) da formule koje rabe vektorski oblik normale »...za razliku od geografske širine i dužine barataju sa tri umjesto sa samo dva broja, ali su zato simetričnog oblika i pogodnije za računanja.« (citat u slobodnom prijevodu autora). Dosta je vremena međutim trebalo, da bi se takva zapažanja iskoristila i u svakodnevnoj geodetskoj praksi.

## O AUTORU

*Hrvoje Lukatela diplomirao je na Geodetskom fakultetu u Zagrebu 1971. godine. Nakon nekoliko godina terenskog rada zapošljava se u IBM-u kao sistemski inženjer u području znanstvenih i tehničkih aplikacija. Sredinom 70-tih sudjeluje u projektu gradnje napudbine za izjednačenje mreža i pohranu koordinata za geodetsku upravu kanadske provincije Ontario, a u isto vrijeme astronomskim mjerenjima uspostavlja orijentaciju kampusa Sveučilista Kralja Abdulaziza prema Meki. Početkom 80-tih u Calgaryju rukovodi razvojem tehničkih aplikacija za projekt Alaska Highway Natural Gas Pipeline i sudjeluje u preuzimanju sustava obrade rezultata Olimpijade u Sarajevu za Calgary. 1991. osniva u Calgaryju tvrtku Geodyssey Limted, čiji je glavni proizvod u obliku programske biblioteke geodetskih računanja, između ostalog, bio i fundamentalna komponenta sustava globusnih baza podataka Informixa i IBM-a. Nakon preuzimanja proizvoda te tvrtke od strane konzorcija korisnika, djeluje kao samostalni konzultant u Zagrebu.*



**Slika 8.** Koordinatna slika skraćenog vektorskog oblika normale, cilindrična projekcija



**Slika 9.** Koordinatna slika skraćenog vektorskog oblika normale, ortografska projekcija