Térinformatikai Iránytű I.

Dr. **Halmai** Ákos, Ph.D.

Tudományos Munkatárs, Pécsi Tudományegyetem Természettudományi Kar Földrajzi és Földtudományi Intézet Térképészeti és Geoinformatikai Tanszék

Lektorálták:

- Dr. habil. Gyenizse Péter, Ph.D.;
  Pécsi Tudományegyetem, Földrajzi és Földtudományi Intézet

- Németh J. András;
  ESRI Magyarország Kft.

Pécsi Tudományegyetem

Ver.: 1.0.
DOI: https://doi.org/10.15170/TERINF-I-TTK-2024
ISBN: 978-963-626-346-1 (nyomtatott)
ISBN: 978-963-626-347-8 (elektronikus)
Pécs, 2024.11.18.

Tartalomjegyzék

1 Geo- vs. térinformatika

A geoinformatika a földrajz egy ágazata. A földrajz a jelenségek térbeliségével és térrel kialakított kapcsolatrendszerével foglalkozik[1].

Ezen alapdefiníció szerint a geoinformatika a földrajzi adatok gyűjtésével, szakszerű tárolásával, elemzésével, modellezésével és megjelenítésével foglalkozó tudomány.

A térinformatika ennél egy bővebb kategóriát képez: itt a térbeli adatok gyűjtésével, tárolásával, elemzésével, modellezésével és megjelenítésével foglalkozunk – legyen az bármilyen tér: épített, virtuális, orvosi, földtudományi vagy extraterresztriális.

Mivel a két szakterület alapjai és módszerei meglehetősen azonosak, a továbbiakban összefoglalóan térinformatikával foglalkozunk – földrajzi példákkal.

A térinformatika célja, hogy az általa kezelt adatokból információt nyerjen ki, vagy olyan formára alakítsa az adatot, hogy abból egy másodlagos elemzés során, lehetőségünk legyen információt nyerni.

1.1 Adat és Információ

A térinformatika nevében hordozza az „információ" szót. A fenti definícióban viszont az „adat" szerepel, mint ahogy a következő fejezet is az „Adattárolás". Mi a különbség az adat és az információ között?

Talán úgy lehetne a legjobban elkülöníteni, ha azt mondjuk, ahogy az információ az, amire vágyunk; információ alapján cselekszünk, információkra támaszkodva hozunk döntést. Információnak tekintjük azt az adatot, amely számunkra valami hasznos tartalmat hordoz.

De mi az „adat"? Az adat valamilyen állapot, vagy tényállás rögzített reprezentációja; egy pillanatnyi állapotot tükröző bejegyzés, amely valamilyen tulajdonságot ábrázol, annak mért, vagy becsült mennyisége.

Képzeljük el, hogy csoportképeket nézegetünk, mert kíváncsiak vagyunk, hogy valaki – nevezzük például Lajosnak – jelen volt-e, vagy sem. A csoportkép legyen egy teljesen közönséges, 22 megapixeles kép, melyen huszonkét-millió pixel – négyzetes képelem – van. A pixel azt írja le, hogy a fényképezőgép által látott kis térrész milyen színű és milyen megvilágítottságú volt.

De mi mire vagyunk kíváncsiak? Arra, hogy Lajos jelen volt-e, vagy sem? El tudjuk dönteni? Igen. Ránézünk a képre, és ha Lajos látszik, akkor ott volt, ha nem, akkor nem biztos, hogy ott volt. Ez

[1] A definíció ötletét Trócsányi (2010) valamint Pirisi, Trócsányi, & Hajnal (2011) adta.

egy információ. Annak is a legelemibb egysége: egy darab igen, vagy nem. Csúnyán fogalmazva egy darab elemi információ, egy darab logikai érték, egy darab igaz/hamis. Ez az információ.

De ehhez, hogy ezt el tudjuk dönteni, huszonkét-millió pixelt le kellett jegyezni. Ez az adat. És ezen adatnak csak egy kicsi, átalakított és kielemzett része az információ. A képen rengeteg adat van: a háttérben fonnyadnak a szobanövények, valaki leette magát és a sliccébe csípte a nyakkendőjét, kőporcelán a padló és LED-világítás van.

De ebből minkent csak egy valami érdekel. Lajos ott volt-e, vagy sem. A huszonkét-millió adatpixelből csak pár száz pixel információ, amin Lajos látszik. Azt sem nyersen: mentális munkával fel kell dolgozni és fel kell ismerni.

Ezért minden információ technikailag adat, de az adatoknak csak egy része információ – és ráadásul az információ a kérdéstől függ. Bármilyen adat lehet információ – egy bizonyos nézőpontból.

Így először az adat tárolásával, majd az adat információ kinyerés céljából történő elemzésével foglalkozunk.

2 Adattárolás

A térinformatika roppant „adatintenzív” szakma. Nagymennyiségű adatot gyűjt, tárol, kezel, megjelenít és elemez – lényegében egyetlen célból: azért, hogy információt nyerjen ki ezekből az adatsorokból. A teljes soron végig menni, így az adatgyűjtésen, -tároláson, -kezelésen, -megjelenítésen és -elemzésen nagy falat lenne, ezért kezdjük a térinformatika alapjával – az adattárolással.

Bármit is teszünk, az adatot vagy kiolvasnunk-, vagy eltárolnunk kell, így az adatkezeléstől és tárolástól műszaki korlátok miatt nem szabadulhatunk.

2.1 A bináris adattárolás

Az térinformatikai adattárolás – a hagyományos, informatikai adattároláshoz hasonlóan – bináris. Ez a gyakorlatban azt jelenti, hogy minden jelenség, az összes raszter, műholdfelvétel szintvonal, társadalomstatisztikai adatsor binárisan kódolt. Ez mára lassan közhely, de mégis – kevesen értik, ezért nézzünk egy példát! A Föld legmagasabb hegycsúcsa a Csomolungma (Mount Everest), melynek magassága ~8 848,86 m, melyet az egyszerűség kedvéért kerekítsünk 8 849 méterre. Hogy lehet ezt a számot binárisan – kettes számrendszerben – eltárolni? Egyszerűen átvájtuk kettes számrendszerbe:

8849|1
4424|0
2212|0
1106|0
553|1
276|0
138|0
69|1
34|0
17|1
8|0
4|0
2|0
1|1
0|0

Először a 8 849-et elosztjuk kettővel: megvan 4 424-szer, maradt az egy. A 4 424-et a 8 849 aláírjuk, a maradékot a vonaltól jobbra. A 4 424-ben a kettő megvan 2 212-ször (a 4 424 alá írjuk), marad a nulla (a vonaltól jobbra írjuk). Az maradékos osztást addig folytatjuk, míg a sor végén 0|0-t nem kapunk.

Ekkor a vonaltól jobbra eső, egyeseket és nullásokat tartalmazó számsort „ledöntjük”, és ezzel meg is kaptuk az eredményt: 010001010010001_2, melyből az első vezető nulla elhagyható így $8\,849_{10} \rightarrow 10001010010001_2$ (az alsó indexek a számrendszert jelölik, hogy ne kíséreljük meg a kettes számrendszerben felírt számot a tízes számrendszerhez hasonlóan kiolvasni, hogy tízbillió…).

A kettes számrendszerben felírt számok számjegyeit (értéküktől függetlenül) biteknek nevezzük. Egy bit két értéket vehet fel ‘1’ vagy ‘0’; vagy nézőpontunktól függően logikai ‘Igaz’ vagy logikai ‘Hamis’.

Azért, hogy a kettes számrendszerben tárolt adatsorokban hatékonyabban lehessen navigálni, a kettes számrendszerben felírt számokból nyolcelemű csoportokat formálunk, és az így képzett nyolc bitből álló csoportot byte-nak, vagy magyarosan bájtnak nevezzük. Így a 10001010010001_2 felírható úgy is (két vezető nulla visszahelyezésével),

hogy $00100010\ 10010001_2$. Az elektronikai rendszerek fizikai felépítésből fakadóan a digitális adattároló rendszerek nyolcbites bájtokban számolnak és ez egyben a legkisebb lefoglalható egység is. Tehát a $8\,849_{10}$ digitális lejegyzéséhez két darab bájtra van szükség: $00100010\ 10010001_2$, melyből a '00100010' az első bájt, míg a '10010001', melyben csoportonként nyolc darab-, és összesen 16 db bit szerepel.

Ez a '8 bit $\overset{\text{def}}{=}$ 1 byte' rendszer egyben az adat mennyiségének mérésére is alkalmas: például a 9_{10} binárisan felírva 00001001_2. Ez nyolc darab bit, tehát egy byte. A $6\,875\,987_{10}$ binárisan $01101000\ 11101011\ 01010011_2$, ami három byte, tehát 24 bit. Így függetlenül attól, hogy a bináris számjegyek mit takarnak számosságukkal mérhető a tárolandó adat mennyisége is.

Íme a teljes sor: 8 bit $\overset{\text{def}}{=}$ 1 byte; 1 024 byte $\overset{\text{def}}{=}$ 1 kilobyte; 1 024 kilobyte $\overset{\text{def}}{=}$ 1 megabyte; 1 024 megabyte $\overset{\text{def}}{=}$ 1 gigakilobyte; 1 024 gigabyte $\overset{\text{def}}{=}$ 1 terrabyte; 1 024 terrabyte $\overset{\text{def}}{=}$ 1 petabyte.

Ezeket a mértékegységeket rendszerint rövidítve írjuk: bit → 'b'; byte → 'B'; → 'b'; kilobyte → 'kB'; megabyte → 'MB'; gigabyte → 'GB'; terrabyte → 'TB'; petabyte → 'PB'.

Jogosan merül fel a kérdés, hogy a váltószám miért 1 024 és nem 1 000. Ennek is oka a kettes számrendszerhez való igazodás, mert 2^{10} = 1 024, így a váltás binárisan „kerek" számhoz igazodik ($1\,000_{10} \rightarrow 001111101000_2$; $1\,024_{10} \rightarrow 010000000000_2$).

Később felmerült a probléma, hogy az 1 024-es váltószám nem illeszkedik az SI (Système International d'Unités) mértékrendszerhez, mely alapján a távolság-, térfogat-, sebesség stb. méréseink zajlik a mindennapi életben, – mivel az SI előírja az ezres váltószámot. Ezért, hogy megkülönböztessék az 1 024-es váltószámmal felírt mennyiségeket az SI mértékegységektől, betoldottak középre egy 'i' betűt: kiB, MiB, GiB, TiB… Ezen mértékegységek kiolvasása a szokásos „kilobyte", „megabyte", stb. helyett „kibibyte", „mebibyte" stb. formában vagy ritkábban a „kilo-bi-byte", „mega-bi-byte" stb. alakban történik.

A gyakorlatban azonban ezzel szinte senki sem törődik: az 'i'-s mértékegységek csak igényesebb informatikai írásokban, és szabadszoftveres környezetekben fordulnak elő, így ha pontosan akarunk számolni, mindig meg kell győződnünk arról, hogy az 1 MB az 8 388 608 bitet, vagy 8 000 000 bitet jelent-e. Az operációs rendszerek gyártói az előbbit, a háttértároló- és memóriagyártók az utóbbit favorizálják. Ennek megfelelően a „kibi-", „mebi-" stb. előtagok kiolvasását is hanyagolják és az eredeti, „kilo-", „mega-" alakokat használják.

2.2 Bájt vagy byte?

A bájt eredeti alakja a „byte", eredeti, amerikai angol kiejtésében /baɪ't/$_{US}$, amely egy szándékos elírás/átírás eredménye.

Az adat legkisebb egysége a „bit” /bɪ't/$_{US}$, amely körülbelül egy kis darabkát jelent: egy kis darabka adat. Maga a „bit” szó nem az informatikából származik, széles körben használják az olyan hétköznapi szószerkezetekben, mint a „little bit of…” (egy kis…).

Ha ezeket a kis „darabkákat”, vagy „szemeket” összefűzzük (nyolcasával), akkor egy „falatot”, vagy „falatkát” kapunk. Egy falat adatot. A falat angolul „bite” kiejtésében /baɪ't/$_{US}$. Bár a bit és a bite két külön dolgot jelöl, kiejtésük nagyon eltérő (/bɪ't/ vs. /baɪ't/), írásmódjuk meglehetősen hasonló (egy ’e’ a különbség), ezért elgépelhető és az elgépelés ebben az esetben értelemzavaró is egyben.

Ezen elgépelhetőség elkerülése érdekében az IBM-nél (International Business Machines) a bite írásmódját szándékosan „byte”-ra módosították (IEEE Computer Society, 2024), mert az angol helyesírás szerint ez továbbra is /baɪ't/-nak olvasható, – de mivel két betű is különbözik – elgépelni már sokkal nehezebb.

A byte eredeti alakja a „byte”, de ahogy az informatika egyre jobban meghonosodott a magyar nyelvterületen is, megjelent a byte fonetikusan átírt magyarosított megfelelője a „bájt” is. Jelenleg mindkettő elfogadottan használható. A byte toldalékolása kötőjeles (byte-ot, byte-ok…) a bájté nem (bájtot, bájtok…).

2.3 Bináris? Digitális?

Mind a „digitális” mind a „bináris” kifejezés elterjedten használt az informatikában. A kettő közül a „digitális” a nagyobb, gyűjtőfogalom. Szótári jelentése szerint az olyan adatreprezentációt takar, ahol az adatok diszkrét értékekkel, egész számokkal kerülnek lejegyzésre. Így a digitális adattárolás a számokat, a képeket, a szövegeket, a hangokat, a videókat és minden más adattípust is számokkal jegyez le. Ennek egyik alesete a „bináris”, ahol az adat lejegyzésére használt számok a kettes számrendszerből valók. Ezzel a jelenlegi informatikánk digitális, azon belül bináris.

A genetika is digitális, ellenben nem bináris: a DNS-ben négy diszkrét érték, négy különböző vegyület, négy nukleobázis található: az adenin, a timin, a guanin és a citozin. Ezek a nukleobázisok hármas csoportokba szerveződnek és így tárolják a faj- és egyedspecifikus adatokat.

2.4 Szövegábrázolás

A szövegek tárolása az informatika egyik klasszikus feladata. De hogyan lehet szövegeket tárolni, amikor a betűk nem számok és pláne nem két darab van belőlük. Pedig előbbiek alapján tudjuk, hogy az adatreprezentációnak binárisnak kell lenni.

2.4.1 A Morse–kód

A régebbi adattovábbítási eljárások közül a Morse–kód digitális és bináris is egyben: akár még jó is lehet szövegtárolásra. A Morse–kód összesen két jelet használ teljes szöveges állományok továbbítására: a rövid ‘ti’ (nyomtatva ‘•’) és a hosszú ‘tá’ (‘–’) jelet.

Működése nagyon egyszerű: az ábécé betűihez hozzárendel egy egyedi, csak az adott betűre jellemző jelkombinációt, és például az ‘a’ kombinációja ‘• –’, a ‘b’-é ‘– • • •’, a ‘c’-é ‘– • – •’. Így az egész ábécé kódolásának ismeretében teljes szövegek továbbíthatók pusztán két jel használatával – és az már csak szimbolika kérdése, hogy az egyik állapotot ‘•’-nak, vagy ‘0’-nak jelöljük, míg a másikat ‘–’-nak, vagy ‘1’-nek.

Leghíresebb Morse–kód a ‘• • • – – – • • •’, vagy ismertebb formájában az ‘$\overline{\text{SOS}}$’. (A felsővonás az üzenet speciális mivoltát jelöli: az ‘$\overline{\text{SOS}}$’ nem egy szó, és túlmutat a benne lévő betűk jelentésén – ez volt a régen a nemzetközi segélykérés Morse–kódja.) Ha klasszikusan, a kettes számrendszer számjegyeiben gondolkodunk, akkor az ‘$\overline{\text{SOS}}$’-t felírhatjuk így is: ‘000111000_2’.

A Morse–kódnak azonban van egy korlátja, amely megnehezítené a modern informatikai használatát: a betűkhöz rendelt kódszekvenciák különböző hosszúságúak (1. táblázat). Itt például az ‘e’ betű egy bit, még az ‘y’ négy. Praktikusabb lenne egy olyan megoldást választani, ahol a kódok egyforma hosszúságúak.

1. táblázat. Az angol ábécé betűinek Morse–kódja – nemzetközi változat *(Snodgrass & Camp, 1922)*.

Betű	**Kód**	**Betű**	**Kód**
A	• –	N	– •
B	– • • •	O	– – –
C	– • – •	P	• – – •
D	– • •	Q	– – • –
E	•	R	• – •
F	• • – •	S	• • •
G	– – •	T	–
H	• • • •	U	• • –
I	• •	V	• • • –
J	• – – –	W	• – –
K	– • –	X	– • • –
L	• – • •	Y	– • – –
M	– –	Z	– – • •

2.4.2 A telex

Az egyforma kódhossz problémájára a megoldás még az informatika kora előtt megszületett: megjelentek a telexek, a géptávírók és ezek kódolási rendszerét a modern informatika is örökölte. A telexek olyan billentyűzettel rendelkező telekommunikációs eszközök voltak, amelyekkel szöveget lehetet nagy távolságokra továbbítani. A kezelő egyszerűen leütött egy betűt és ezt az ezzel párba kapcsolt készülék a távolban papírra vetette.

A telexnél az egyes betűkhöz azonos hosszúságú bináris számokat rendelünk – még akkor is, ha vezető nullákat kell hozzáadnunk a kódhoz:

… Itt most csak három betű szerepel mintaként, de látható, hogy
'**a**' → 97_{10} → 01100001_2 egy táblázat segítségével a szövegírás is megoldható binárisan,
'**b**' → 98_{10} → 01100010_2 csak meg kell szerkesztenünk egy olyan listát, vagy táblázatot,
'**c**' → 99_{10} → 01100011_2 amely megfelelteti az egyes betűket egy-egy mellérendelt, de
… mindig azonos hosszúságú számnak. Ha például szeretnénk leírni azt, hogy „abba", akkor a háttértárolóba, vagy a memóriába azt kell bejegyezni binárisan, hogy '$01100001011000100110001001100011_2$'. Ebből a bináris adatfolyamból az első nyolcdarab bináris számjegy (1 byte; '01100001_2') reprezentálja az 'a' betűt, a '01100010' reprezentálja a 'b' betűt (kétszer egymás után), majd az utolsó nyolc '01100011' számjegy ismét az 'a' betűt. Ebből az átírásból látható, hogy a digitális, bináris számítógépek számára a betűk semmilyen kitűntetett jelentéssel nem bírnak, ugyan úgy binárisan kódoltak, mint a számok. A szövegfájlok képernyőre nyomtatása egyszerűen úgy zajlik, hogyha a gép észlel a bináris adatfolyamban egy '01100010_2' szekvenciát, akkor kirajzol egy függőleges vonalat, melynek a jobb oldalán lapított „hasa" van. És ezt a felhasználók egyszerűen 'b'-nek fogják látni. Ezt a hozzárendelési táblázatot *kódtáblának*, vagy *karaktertáblának*, néha *kódlapnak* nevezzük.

A kódtáblák természetesen nem csak ezt a három betűt írhatják le: nyolc biten legfeljebb $2^8 = 256$ féle betűt, jelet és egyéb karaktert tárolhatunk el.

Mára csak történeti szempontból érdekes, hogy a telexek eredetileg csak 7 bitet használtak, így 128 féle karakter továbbítására voltak képesek; a 8. bitet csak később fűzték hozzá, ami további 128 karakter tárolását tette lehetővé. Az eredeti 7-bites szabvány neve ASCII – American Standard Code for Interchange Interface (Amerikai, Szabványos Kód-csere Formátum), míg a 8-bit változat az Extended ASCII – a Kiterjesztett ASCII.

A 8-bites karaktertábla három részre osztható:

1. Első szelete a tízes számrendszerben $0–31_{10}$ közé eső tartomány plusz a 127_{10}-es karakter, melyeket vezérlőkaraktereknek nevezünk.
2. A második szegmens a nyomtatható karakterek csoportja 32_{10} és 126_{10} között, ahol az angol ábécé kis- és nagybetűi szerepelnek a központozási jelekkel és néhány egyéb gyakori karakterrel együtt.
3. Majd a harmadik tartomány ($128_{10}–255_{10}$) a kódlap kiterjesztett része, ahol egy pár új vezérlő-, néhány különleges- és néhány gyakori ékezetes karakter kapott helyet.

2.4.3 A vezérlőkarakterek

A vezérlő karakterek nem kódolnak nyomtatható elemeket, hanem a telex fizikai működését szabályozzák (2. táblázat):

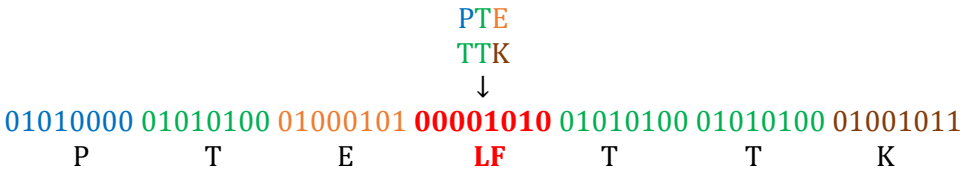
2. táblázat. A vezérlőkarakterek (nem nyomtatható karakterek, Oualline [1997]):

№	Bináris reprezentáció	Rövidítés	Angol megnevezés
0	00000000_2	NUL	Null
1	00000001_2	SOH	Start of Header
2	00000010_2	STX	Start of Text
3	00000011_2	ETX	End of Text
4	00000100_2	EOT	End of Transmission
5	00000101_2	ENQ	Enquiry
6	00000110_2	ACK	Acknowledge
7	00000111_2	BEL	Bell
8	00001000_2	BS	Backspace
9	00001001_2	HT	Horizontal Tab
10	00001010_2	LF	Line Feed
11	00001011_2	VT	Vertical Tab
12	00001100_2	FF	Form Feed
13	00001101_2	CR	Carriage Return
14	00001110_2	SO	Shift Out
15	00001111_2	SI	Shift In
16	00010000_2	DLE	Data Link Escape
17	00010001_2	DC1	Device Control 1
18	00010010_2	DC2	Device Control 2

19	00010011_2	DC3	Device Control 3
20	00010100_2	DC4	Device Control 4
21	00010101_2	NAK	Negative Acknowledge
22	00010110_2	SYN	Synchronize
23	00010111_2	ETB	End of Transmission Block
24	00011000_2	CAN	Cancel
25	00011001_2	EM	End of Medium
26	00011010_2	SUB	Substitute
27	00011011_2	ESC	Escape
28	00011100_2	FS	File Separator
29	00011101_2	GS	Group Separator
30	00011110_2	RS	Record Separator
31	00011111_2	US	Unit Separator
32	00100000_2	space	Space
Nyomtatható karakterek tartománya (l.; 4. táblázat) 33–126.			
127	01111111_2	DEL	Delete

Ezen vezérlőkarakterekből több átöröklődött a modern informatikába is. Ezek közül leggyakrabban a Space-el, a sima szóközzel találkozunk, amely bár a képernyőn üres helynek látszódik, valójában binárisan eltárolt karakter, amely kirajzoláskor egy üres helyet hagy.

Gyakori a „CR” – Carriage Return (kocsi vissza) és az „LF” – Line Feed (laptovábbítás) karakter is: ezeket a karaktereket használjuk a sortörés megjelölésére. A Windows® a CR+LF kombinációt, a Unix-szerű operációs rendszerek (pl.: Linux) pedig csak az „LF”-et a használják. Így akárhányszor entert ütünk egy szövegfájlban, annyiszor egy-, vagy két karakter fűződik hozzá a fájlhoz, amely a szöveg kirajzolásának pillanatában azt eredményezi, hogy a felirat a képernyőn egy új sorban jelenik meg. Annak ellenére, hogy vizuálisan a szöveg megtörik, magában a fájlban a byte-ok teljesen folytonosan (sorfolytonosan) jelennek meg (1. ábra).



1. ábra. A sortörés vezérlőkarakter bináris megjelenése (középen, dőlt betűkkel). Hat darab nyomtatható- és egy darab vezérlőkarakter („LF"): összesen 7 byte.

Később, további vezérlőkaraktereket fűztek a kódlaphoz a 128–159-es tartományban, de ezek használata kevésbé elterjedt.

2.4.4 A nyomtatható karakterek és a kiterjesztett ASCII

Az ASCII kódtábla nyomtatható karakterei tartalmazzák a számkaraktereket, az angol ábécé kis- és nagybetűit, relációs- és központozási jeleket, a főbb záró- és idézőjel-típusokat, valamint a matematikai operátorokat.

Ezzel a karakterkészlettel a legtöbb angolnyelvű szöveg lejegyezhető. Az informatika nemzetközivé válásával azonban felmerült az igény, hogy ne csak angolnyelvű szövegek karaktereit jegyezhessük le.

A 8-bites, kiterjesztett ASCII kódtáblát szánták első körben ennek orvoslására. Ez a kódlap ugyan tartalmazott néhány ékezetes karaktert, például az 'ö'-t, de nem tartalmazta az 'ő'-t. Ugyanígy szerepelt benne egy pár karakter a francia-, spanyol- és germán nyelvterületekről – de ezekből sem az összes –, és egyáltalán nem voltak a táblán a szláv nyelvek karakterei. A korlátozottság abból fakadt, hogy az elvi 256 helyből már csak nagyon kevés kódpont maradt: ide kellett bezsúfolni azokat a kerektereket, amelyeket a legfontosabbnak tartottak.

2.4.5 A nemzeti kódlapok

A kiterjesztett ASCII csak korlátozott számban tartalmazott ékezetes karaktereket, melyek megléte elengedhetetlen a legtöbb nyelv számára. Erre született a nemzeti kódlapok használatának gondolata. A koncepció lényege, hogy ugyan sok ékezetes- és speciális karakterre van szükség – de rendszerint nem egyszerre. Így például, ha magyarul írunk – ami ékezetes betűben gazdag nyelv – rendszerint nincs szükségünk a francia 'è', sem a svéd 'å' jelére. Ezen gondolatmenet alapján az az ötlet született, hogy az adott nyelv igényinek megfelelően cserélgessük a felső 128 db, nagy számértékekkel leírt karaktert a 256 elemű, 8-bites kódtáblában.

Ez azt jelenti, az alsó (alacsony számértékű), 0–127-es tartományt (mely 7 biten fér el) változatlanul hagyjuk és a felső (nagy számokkal lejegyzett), 128–255-ös tartomány karaktereit cserélgetjük a nyelvi igényeknek megfelelően. Ez a szabvány az ISO 8859-es azonosítószámot kapta és 15 különböző kódlapot definiáltak, melyeknek a felső 128 karaktere (többségében) más és más, így például az ISO 8859–1 írja le a nyugat-európai nyelvek ékezetes karaktereit, vagy az ISO 8859–2 a közép-európai ékezetes karaktereket (a magyar is ide tartozik).

Az eljárás előnye, hogy a kódtába továbbra is 8-bites, az angol ábécé betűi változatlanul elérhetőek (mert azok az alsó 128 elemhez tartoznak); de hatalmas hátránya, hogy a szöveg megjelenítéséhez szükséges kódlapot is *pontosan ismerni kell*. És ha a felhasználó nem ismeri a táblák közötti különbséget, akkor legfeljebb krikszkrakszokat olvasgathat.

Így például az ISO 8859 világában szembe találkozhatunk az alábbi karakterláncolattal: „ٴrvẓtr tkrfrgیp” Vajon mire gondolhatott a költő? Ebből a kódsorból elég nehéz rájönni, de egy kis próbálgatással kideríthető, hogy az eredeti szöveg „*Árvíztűrő tükörfúrógép*” volt. És mi itt a probléma? Az „Árvíztűrő tükörfúrógép”-et eredetileg ISO 8859–2-es kódlapon írták, de véletlenül ISO 8859–6-os lapon (arab) jelenítjük meg. Mindeközben a fájl bináris tartalma *semmit sem változott* – pusztán rossz megjelenítési kódlapot használunk és erről semmi sem tájékoztat minket. (Árvíztűrő tükörfúrógépek nincsenek: ez egy konstruált mintamondat, melyben az összes magyar ékezetes karakter szerepel.)

Ebben a szituációban mutatkozik meg a karaktertáblák igazi viselkedése: a kódlapok csak hozzárendelési táblázatok, amelyek valamilyen vizuális formát rendelnek egy adott bináris szekvenciához a képernyőn. A karakterek csak lejegyzett számok, melyek értékük és a kódlap alapján valamilyen formát vesznek fel a képernyőn, melyet a felhasználó betűnek, központozási jelenek, szóköznek, vagy más egyéb jelenek lát – valamilyen tanult megszokás alapján.

Ezen korlátozások ellenére az ISO 8859-es kódlapokat széleskörben használták (Magyarországon az ISO 8859–2-est).

2.4.6 UTF-8

A nemzeti kódlapok a karaktertábla felső 128 elemében elég sokféle karaktert képesek lefedni, de közel sem az összeset.

Alapítsunk egy pénzváltót Tibetben és nevezzük el a Föld legmagasabb hegycsúcsáról, a Csomolungmáról! A hegymászók és a turisták a világ sok tájáról érkeznek, és sok nyelven beszélnek. A pénzváltónknak honlapot is szerkesztünk, ahova több nyelven is ki szeretnénk írni a nevünket, elérhetőségünket és az árfolyamokat.

Először kiírjuk angolul, hogy „Mt. Everest”. Ezzel nem is lehet semmi baj, hiszen a benne szereplő betűk az ASCII kódtáblán is rajta vannak. Írjuk ki tibetiül is: „ཇོ་མོ་གླང་མ”! Itt már nem ilyen egyszerű a helyzet, de egy nemzeti kódlappal még megoldhatjuk. Persze ki kellene írni kínaiul („珠穆朗玛峰”), japánul („エベレスト”), hindiül, dévanágari írással („एवरेस्ट पर्वत”), nepáliul, a dévanágari egyik módosított változatával („सगरमाथा”). És ha ennyivel megelégszünk, még akkor is ki kell írni a pénznemeket, népszerű jeleikkel együtt: $, €, £, ₽, ฿ és még mindig csak a leggyakoribbaknál tartunk. Itt már nem támaszkodhatunk nemzeti kódlapokra: egy sincs, amely az összes karakterünket lefedné. Egyszerűen új kódolási eljárásra van szükség…

Ezen kihívásra született meg az UTF-8 (8-bit Unicode Transformation Format). Az UTF-8 szakít a 8-bites korlátozással és a speciális karaktereket több bájton írja le. A kódtábla ASCII (alsó 127 karakter) része változatlan, de a felett egy új kódolási eljárást vezettek be.

Először írjuk le az ‘a’ betűt! Ennek bináris reprezentációja 01100001_2 – akárcsak az ASCII kódtáblában: egy byte, nyolc bit. Most próbálkozzunk meg az ‘á’ betűvel! Az ‘á’ UTF–8 bináris reprezentációja 11000011 10100001_2. Ez kettő byte, tizenhat bit. Lépjünk eggyel tovább és jöhet az Oṃ jele az indiai eredetű vallásokból: ‘ॐ’. Ennek a jelnek az UTF-8 bináris reprezentációja 11100000 10100101 10010000_2, ami három byte hosszú. Négy bájt hosszúságúak az igazán ritkán használt karakterek, mint például a 𠜎-jel, vagy, az emodzsik.

Az UTF-8 kódolás ötletét a 3. táblázat mutatja be:

3. táblázat. Az UTF-8 karakterkódolás bináris mintázatai. A ‘v’ betűk az értékbiteket jelölik (.NET Foundation, 2024).

Byte [db]	Értékbitek [db]	UTF-8 reprezentáció
1	7	$0vvvvvvv_2$
2	11	$110vvvvv_2$ $10vvvvvv_2$
3	16	$1110vvvv_2$ $10vvvvvv_2$ $10vvvvvv_2$
4	21	$11110vvv_2$ $10vvvvvv_2$ 10vvvvvv $10vvvvvv_2$

Itt látható, hogy azok a karakterek, amelyek az eredeti ASCII táblázaton is szerepeltek hét értékbittel jelennek meg (128 karakter), miközben az első számjegy *mindig* nulla. Ha az UTF-8 feldolgozó algoritmus olyan byte-ot talál, melynek elő számjegye nulla, akkor tudja, hogy a karakter a 0–128-as tartományba esik.

Abban az esetben, ha az első három számjegy 110_2, akkor a dekóder tudja, hogy még egy byte-ot be kell olvasni. Ekkor a második byte első két számjegye 10_2, azért, hogy ne lehessen összekeverni a nyitóbyte-al. Ha a feldolgozó 1110_2 talál a byte elején, akkor tudja, hogy még további két byte-ot be kell olvasni. Ha a nyitóbyte 11110_2-val kezdődik, akkor további hármat. Ha figyelmen kívül hagyjuk a formailag kötelező egyeseket és nullásokat a byte-ok elejéről, akkor szabadon felhasználható értékbiteket kapunk: ezeket jelöltük 'v'-vel a 3. táblázatban.

Ezzel az eljárással 2^{21} = 2 097 152 db önálló karaktert lehet kódolni, amely még az ázsiai nyelvek teljes karakterkészletének eltárolására is elegendő. (Jelenleg az UTF-8-ban 1 112 064 db érvényes karaktert tartanak nyilván a latin betűktől kezdve, az összetett írásrendszerű nyelveken keresztül, a ritka- és holtnyelvek, valamit a jelkarakterekkel bezárólag.)

Az eljárás hátránya, hogy egy kicsit komplikált, illetve a kötelező formális bitek miatt egy kevés helyet elpazarol; de előnye, hogy cserébe egy kódlappal az összes használatban lévő karakterünk elkódolható és az alsó 128-as tartományban kompatibilis az eredeti, ASCII kódlappal. Ezen előnyei miatt ez lett a modern informatikai adattárolás elfogadottan használt kódlapja. Használjuk ezt az eljárást, ahol csak tehetjük!

Az UTF-8 speciális, a felhasználót nem érintő programozás-technikai hátránya, hogy egy szövegben a karakterszám ismeretében nem határozható meg, hogy hány bájtra lesz szükség az adat lejegyzésére, mivel a karakterek változó bájtszámon kódoltak. Így ahhoz, hogy például egy négybetűs szöveg esetében meg tudjuk mondani, hogy hány bájt kell, ahhoz a szöveget *ténylegesen* fel kell dolgoznunk. Például, a csak ASCII karaktereket tartalmazó „fura" szöveg pontosan négy bájt, a kínai „生辰八字" viszont 12, mert minden karakter három bájtos.

Az UTF-8 karaktereket tartalmazó szövegek jól tömöríthetők, mert az egyes karakterek jelölőbitjei azonosak és hosszú szövegek esetén gyakran ismétlődnek, így speciális algoritmusok képesek összevontan, hatékonyan tárolni őket.

Az UTF-8 képes kezelni a jobbról-balra és a balról-jobbra író nyelveket egyaránt – akár vegyes formában is –, köztes, nem nyomtatható irányjelző karakterek beszúrásával.

2.4.7 UTF-16, UTF-32 – és más egzotikumok

A fent bemutatott UTF-8 csak egy – a legelterjedtebb – a több-bájtos karakterkódolási eljárások közül. Vannak olyan megoldások, melyek alapegysége nem az (egy) bájt, hanem minden karakter lejegyzésére legalább két bájtot használ. Ilyen eljárás az UTF-16, ennek négybájtos, 32-bites változata az UTF-32. Több operációs rendszer belső szövegtárolására az UTF-16-ot használja. Ha

tehetjük, kerüljük ezen kódlapok (és más mostanra kihalt kódlap, pl. az UCS-2, UCS-2-LE-BOM) használatát! Az UTF-16-ot – megtévesztően – „Unicode”-nak is nevezik.

2.4.8 A földrajzban használt gyakori jelek

Ahhoz, hogy a földrajzi-, térinformatikai környezetben megfelelően tudjuk szöveget tárolni, íme néhány fontosabb karakter az UTF-8-as karaktertábláról:

- µ mikro-jel, olyan összetételekben, mint a µm (mikrométer). Vizuálisan gyakran, de technikailag nem egyezik a görög, kis µ-vel (kis mű-vel).
- Ω Ohm-jel. Vizuálisan gyakran, de technikailag nem egyezik a görög nagy Ω-val (nagy ómegával).
- ° fok-jel, pl.: °C, vagy északi szélesség 46°.
- Hőmérsékleti fokjelek: °C; °F. (a fokjel és az azt követő betű egy karakterként szerepel.)
- ´ perc-jel, pl.: 46°18´. (Nem egyezik az aposztrófval: ' vs. ´.)
- ″ másodperc-jel, pl.: 46°18´27″. (Nem egyezik a sima idézőjellel, sem a nyomdai, magyar nyelvterületen alkalmazott jobb oldali idézőjellel: ″ vs. ", illetve ”.)
- ′ láb-jel: 1′ $\overset{\text{def}}{=}$ 0,3048 m. (Nem egyezik a perc jellel: ′ vs. ´.)
- ″ hüvelyk- / inch- / zoll- jel: 1″ $\overset{\text{def}}{=}$ 2,54 cm.
- ∶ arányjel, pl.: 1∶100 000. Itt az egy és a százezer között arány jel szerepel, sima kettőspont helyett (∶ vs. :). Az arányjel megjelenése betűtípustól függő, de rendszerint magasabban ül, mint a kettőspont.
- Égitestek jelei: Nap: ☉; Merkúr: ☿; Vénusz: ♀; Föld: ⊕; Mars: ♂; Jupiter: ♃; Szaturnusz: ♄; Uránusz: ♅; Neptunusz: ♆. Hold az első negyedben: ☽; Hold az utolsó negyedben: ☾; Telihold: ○; Újhold: ●. Plútó törpebolygó: ♇.
- Állatövi jegyek: Kos: ♈; Bika: ♉; Ikrek: ♊; Rák: ♋; Oroszlán: ♌; Szűz: ♍; Mérleg: ♎; Skorpió: ♏; Nyilas: ♐; Bak: ♑; Vízöntő: ♒; Halak: ♓; Kígyótartó: ⛎ (az asztrológiában nem használt, 13. állatövi jegy).
- „Bizonyítás vége” jel: ■.
- Római számok: I; II; III; IV; V; VI; VII; VIII; IX; X; XI; XII; L; C; D; M. Ezen jelek érdekessége, hogy még az összetett számjegyek is egy karaktert alkotnak, pl.: a római XI (11) egy darab karakterrel és nem két karakterrel (egy nagy X-el és egy nagy I-vel) ábrázolt.
- Derékszög: ∟. Műszaki változata: ⌞.
- Geodéziai alappont: ⧋.
- Végtelen: ∞.
- Plusz-mínusz jel: ±.

- Abszolút-érték jel: |…|; alsó egészrész: ⌊…⌋; felső egészrész: ⌈…⌉ (a '…' helyére kell a számnak, kifejezésnek kerülni).
- Gyakoribb, egy karakterrel ábrázolt törtek: ¼; ½; ¾; ¾; ⅕; ⅝.
- „Copyright"- és „Trademark"-jel: ©; ™.
- Számhalmazok jelei: ℕ; ℝ; ℤ. (Természetes-, valós- és egészszámok.)
- Matematikai halmaz eleme- és nem eleme jel: ∈; ∉.
- Egyenlőségek és relációk: = egyenlő; ≝ definíció szerint egyenlő; ≈ megközelítőleg egyenlő; ≤ kisebb, vagy egyenlő; ≥ nagyobb, vagy egyenlő; ≔ legyen egyenlő.
- Matematikai összegzés: ∑. Nem azonos a nagy görög nyomtatott szigmával: ∑ vs. Σ. Ritkán használt párja a produktum: ∏. Nem azonos a görög, nagy, nyomtatott pível: ∏ vs. Π.
- A görög kisbetűk: α; β; γ; δ; ε; ζ; η; θ; ι; κ; λ; μ; ν; ξ; ο; π; ρ; σ; τ; υ; φ; χ; ψ; ω.
- A görög nagybetűk: Α; Β; Γ; Δ; Ε; Ζ; Η; Θ; Ι; Κ; Λ; Μ; Ν; Ξ; Ο; Π; Ρ; Σ; Τ; ϒ; Φ; Χ; Ψ; Ω.
- Numero jel: №. (Például típus, vagy sorszám megjelölésére.)
- Százalék- és ezrelék jel: %; ‰.
- Meteorológiai piktogramok: ⛈; ☀; ☁; 🌫; ◔; ◑; ◕;●.
- Az UTF-8 támogatja a felső- és alsó indexek használatát is, melyek karakterekként elérhetők: $^{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ +\ -}$. Ezek segítségével természetföldrajzi környezetben, sima szövegfájlokban is eltárolhatók olyan, tudományosan helyes megírások, mint a ^{235}U, vagy a Fe^{3+}. Ugyanígy az UTF-8 világában szövegszerkesztő nélkül sem vagyunk rászorulva a m2-re a helyes m^2 (négyzetméter) helyett.
- Hasonlóan elérhetők az alsó indexek is: $_{0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ +\ -}$.
- Bekezdés („Pilcrow") jel: ¶, paragrafus jel: §.
- Pénznemek: ₠; ₡; ₢; ₣; ₤; ₥; ₦; Pts; Rs; ₩; ₪; ₫; €; ₭; ₮; ₯; ₰; ₱; ₲; ₳; ₴; ₵; ₶; ₷; ₸; ₹; ₺; ₻; ₼; ₽; ₾; ₿.

2.4.9 Az emodzsik 😎

Az emodzsik, vagy angolosan emoji-k, ritkábban az emotikonok az UTF-8 kódtábla részei. Piktogram szerű megjelenésük ellenére ugyanolyan karakterek, mint a karaktertábla bármely más eleme: 🌍 🤔😬🐞⛿ 💳 ⚰ 🌈 💫 🍌🍩 … Használhatjuk őket akár térképszerkesztés során, például ikonként, piktogramként. Az emodzsik között elérhetők az országkódok is, zászló-, vagy karakteres formában: HU.

2.4.10 Speciális karakterek a szövegszerkesztésben

Az UTF-8 számos olyan karaktert tartalmaz, amely vizuális megjelenésében nem hordoz semmilyen különleges információt, de ha szövegszerkesztőben használjuk, akkor módosítja a szöveg tördelését. Vegyük magát az „UTF-8”-at. Az „UTF-8”-ban egy kötőjel szerepel. A kötőjelek mentén a szöveg elválasztható. Így például, ha egy hosszú sort írunk, melynek a végére az „UTF-8” kerülné, akkor *technikailag* megfelelő lenne úgy elválasztani, hogy „UTF-” <sortörés> „8”.

Ez az ábrázolási mód ugyan műszakilag helyes, de vizuálisan förtelmes és a megértést is zavarja. Mit keres a sor elején egy szimpla „8”-as?

Az ilyen problémák elkerülésére az UTF-8 tartalmaz egy „nem törhető kötőjelet”, amely vizuálisan pont úgy néz ki, mint a sima kötőjel, de e mentén a szövegszerkesztő nem törhet, így az UTF-8 mindig együtt marad. Az olyan kötőjeles szerkezetek esetében, ahol nem szeretnénk, hogy egy sortörés megjelenjen, minden esetben a nem törhető kötőjelet kell használnunk. Ilyenek például a telefonszámok is: a +36/72/503-600. Itt az 503 és a 600 között egy nem törhető kötőjel szerepel, mert senki sem szeretné úgy látni egy telefonszámot, hogy a vége letört a következő sorba.

Hasonló probléma jelentkezik a szóközöknél is: két, szóközzel elválasztott szó két sorba törhető. Vegyük például az alábbi kifejezést: 16 cm. Itt a 16-ot és a cm-et szóköz választja el. Itt a szövegszerkesztő simán megtehetné, hogy a 16-ot a sor végére, a cm-et pedig a következő sor elejére teszi. Ez erősen értelemzavaró, ezért itt is szerepel egy „nem törhető szóköz”: 16 cm. Vagy ennek egy rövidebb – szintén nem törhető – változata, a keskeny szóköz: 16 cm.

2.4.11 A speciális karakterek támogatottsága

Mára igen nagyszámú betűtípus terjedt el. Ezek közül néhány csak esztétikai- de néhány szakmai célokat szolgál. Az UTF-8 rengeteg kódpontot tartalmaz, így, ha maradék nélkül, azonos tipográfiai stílusban az összes karakter képét el szeretnénk készíteni az irgalmatlan munka lenne.

Ezért a betűtípusok csak az adott nyelvi-, használati környezetben gyakori karaktereket készítették el. Ezért, ha egyes karakterek helyén ‘□’ jeleket látunk, az nem feltétlenül karakterkódolási hiba. Lehet, hogy a karakter tökéletesen lejegyzett UTF-8 kódpont, csak megjelenítését az adott betűtípus nem támogatja.

2.5 Számábrázolás

Az informatika azonban nem csak karakterekkel foglalkozik: a bináris rendszer közvetlenül is alkalmas számok tárolására. A karakterek tarolásai módszertanának vizsgálata során csak egy

számtípussal foglalkoztunk, mégpedig az egész számokkal, azoknak is az egy byte-on ábrázolható változataival (vagy ezek egymás után fűzésével az UTF-8 esetében), tehát praktikusan a 0–255-ös tartománnyal. De mi van akkor, ha azzal a feladattal találjuk magunkat szembe, hogy le kell binárisan jegyeznünk, hogy −6 548? Vagy 0,16ė? Vagy
79 228 162 514 264 337 593 543 950 335? És ezt nem szövegként (egymást követő számkarakterekként), hanem *valódi* számként szeretnénk megtenni.

Ebben az esetben a byte nem lesz elegendő, több byte összefűzésével kell a problémát megoldani. Első lépésben a számokat két családra kell osztani: egész számokra és lebegőpontosokra. Ha megismerjük a számábrázolás módszereit, akkor erre az informatika, így a térinformatika egészét visszavezethetjük, foglalkozzunk akár képekkel, videókkal, hangfelvételekkel, avagy koordinátákkal.

2.5.1 Egész számok

Az egész számok tárolása az informatikában nem különösebb kihívás: annyi bitet fűzünk egymás után, hogy a lejegyezni kívánt szám „elférjen”. Praktikus okokból az alábbi számtípusokat használjuk elterjedten:

- **byte**: 8-bit; tartomány: $0–256_{10}$. Ez technikailag a legkisebb egység, kisebb foglalására műszaki okokból nincs mód.
- **sbyte**: 8-bit; tartomány: $-128_{10}–+127_{10}$. Az „sbyte” egy előjel tárolására alkalmas számtípus, ahol a szó elején az „s” az angol „signed” (jelölt / előjeles) rövidítése.
  Ekkor a rendelkezésre álló 8 db bitből egyet lefoglalunk egy egyszerű kapcsolónak: ha az első bit 1_2, akkor a szám negatív, ha 0_2, akkor pozitív megjelölést kap.
  Az előjel-bites számok speciális érdekessége, hogy az aktuális kezdőponttól mérik a távolságot, amelyet az előjelbit szabályoz, így a $+1_{10}$ bináris reprezentációja $0000\ 0001_2$, mert ha az első-, előjelbit nulla / hamis, akkor a kezdőpont a nulla. Ettől a $+1_{10}$ távolsága egy, így a bináris reprezentáció is 1_2 ($0000\ 0001_2$).
  Viszont abban az esetben, ha az első bit egy (tehát a szám negatívnak jelölt), akkor a kezdőpont a -128_{10}, ezért a -128_{10} bináris reprezentációja 10000000_2 mert a -128_{10}-nak a -128_{10}-tól mért távolsága nulla, ezért az utolsó hét számjegy is nulla.
  Ebből következően a -1_{10} bináris reprezentációja $1111\ 1111_2$, mert az -1_{10}-nek -128_{10}-tól mért távolsága 127_{10} ($-1-[-128] = 127_{10}$), a 127_{10} bináris reprezentációja $111\ 1111_2$, és az előjelbitet az elejére helyezve pedig $1111\ 1111_2$.
  Az eljárás ugyan roppant komplikáltnak hathat, de a mögöttes gondolat nagyszerű: ha 11111111_2-hez (-1_{10}) hozzáadunk egyet, akkor a számsor „körbefordul” és elérjük a 00000000_2, ami nulla – minden számrendszerben.

Ezzel az előjelbit kezeléséhez nem kell speciális áramköröket készíteni, a meglévő teljesen megfelelő, a processzor szempontjából sima byte-nak tekinthető, és csak a felhasználón múlik, hogy az értéket miként értelmezi.
Vigyázat! A byte és az sbyte között semmilyen informatikai különbség sincs. Nekünk kell nyilvántartani, hogy milyen számtípussal dolgozunk, így a $1111\ 1111_2$ lehet -1_{10}, de 256_{10} is, attól függően, hogy a lejegyzett nyolc bitet minek *tekintjük*.

- **short**: 16-bit; tartomány: $-32\ 768_{10}$–$+32\ 768_{10}$. A short két egymás után fűzött byte, előjelbittel, az sbyte-nél leírt kezelési módszerrel. Itt a -1_{10} binárisan $11111111\ 11111111_2$: kétszer nyolc bit, csupa egyessel kitöltve. Ha az előjel bit beállított, akkor a referencia érték $-32\ 768_{10}$ ($10000000\ 00000000_2$). A short ritkán használt párja az ushort, amely az „unsigned", tehát jelöletlen/előjelbit nélküli változatát jelöli. Az ushort tartománya: 0–$65\ 535_{10}$.
  A short technikailag semmiben sem különbözik két, egymás után írt bájttól.
- **integer**: 32-bit; tartomány: −2 147 483 648–+2 147 483 647. Az integer négy egymás után fűzött byte, előjel-bittel, az sbyte-nél leírt módszer szerint. Jelöletlen változata az uint.
- **long**: 64-bit; tartomány: −9 223 372 036 854 775 808–+9 223 372 036 854 775 808. A long nyolc egymás után fűzött byte, előjel-bittel, az sbyte-nél leírt módszer szerint. Jelöletlen változata az ulong (unsigned long).

Az itt leírt nevek (byte, short, integer, long) csak tájékoztató jellegűek – és bár a legtöbb rendszer ezen megnevezéseket használja – egyes régebbi- és/vagy speciális operációs rendszeren, szoftverben, vagy fejlesztő-környezetben ettől eltérő megnevezések is szerepelhetnek más-más bitszámmal, így más minimum- és maximum értékekkel. Gyakori megnevezések még:

- **char**: 8-bit, *általában* a fenti byte-al egyenértékű. Neve kissé megtévesztő, mert szöveggel nem összefüggő környezetekben is előfordul, de csak arra utal, hogy az adategység hossza egyezik egy (8-bites) ASCII karakter hosszával. Több térinformatikai program a raszter bitmélységét unsigned char-nak jelöli.
- **word**: 16-bit, *általában* a fenti short-tal egyenértékű.
- **dword**: 32-bit, „double word" (duplaszó), *általában* a fenti integer-rel egyenértékű.
- **qword**: 64-bit, „quad word" (négyes-szó), *általában* a fenti long-gal egyenértékű.

Speciális, különösen rasztertárolási esetekben előfordulhatnak eltérő bitszámú változók is. Például a térinformatikai rendszerek támogatják az egybites és a négybites változókat is, melynek elsődleges oka a tárhelyigény csökkentése, olyan esetekben, ahol több adatra nincs szükség, pl. egy hét kategóriát tartalmazó felszínborítás osztályozás négy biten is eltárolható.

2.5.2 Lebegőpontos számok

A „lebegőpontos számok” egy érdekes informatikai megnevezés, amely a tárolás módszerére utal.
De milyen számok és számtípusok lejegyzésére alkalmasak? A lebegőpontos számtípusok segítségével törteket, azon belül tizedes törteket jegyezhetünk le. Az informatikában használt bináris számrendszer azonban önmagában nem alkalmas törtek lejegyzésére: a bináris rendszer csak biteket tud tárolni – amelyek akár nagy egészszámokká fűzhetők össze –, de tört jeleket, vagy arányokat nem.

Ezért a tört számok tárolását egyszerűen az egészszámok tárolására vezették vissza. De hogyan lehet törtet egész számként tárolni? A megoldás a tizedes pont „lebegtetése”:

$$1{,}41421 = \mathbf{141421} \cdot 10^{\mathbf{-5}}$$

Itt, az 1,41421-et leírhatjuk úgy is, hogy 141421 – amely egy egészszám – szorozva 10^{-5}-el. Ha azt mondjuk, hogy a 10-et elhagyjuk, mert úgyis mindig ezt használjuk, akkor marad a -5 – ami szintén egy egész szám. Tehát egy tizedes tört két egészszámra bontható: *mantisszára* (141421) és *kitevőre* (-5). Ezzel a kitevő értéke alapján jobbra–balra tologathatjuk „lebegtethetjük” a tizedes pontot. Ezen eljárásról kapta a lebegőpontos, vagy angolul *floating point* számábrázolás a nevét.

Az informatikában három lebegőpontos számábrázolás terjedt el:

1. A single, mely 32-biten tárolja a lebegőpontos számokat az alábbi képlet szerint: $(-1)^s \cdot m \cdot 2^k$, ahol ‘s’ az előjelbit (egy vagy nulla), ‘m’ a mantissza, ‘k’ a kitevő. A kitevővel való hatványozás alapja a kettő. A 32 bitből egy az előjelbit, nyolc a kitevő, 23 a mantissza.
2. A double, mely a single dupla hosszúságú, 64-bites változata, tárolásának képlete a single-évelmegegyező: $(-1)^s \cdot m \cdot 2^k$. A 64 bitből egy az előjelbit, 11 a kitevő, 53 a mantissza.
3. A decimal egy 128-bites, sepciális lebegőpontos számtípus, melynek tárolási képlete $(-1)^s \cdot m \cdot 10^k$. Jelentős különbség, hogy a hatványozás alapja 10 és nem kettő.

Mind a single, mind a double képes speciális, számjegyekkel nem leírható értékeket eltárolni: ezek az értékek a pozitív- és negatív végtelen (képernyőre nyomtatva ∞ és −∞-ként, vagy INF és −INF-ként jelenik meg) és a „Nem szám / Not-a-Number”, amelyet gyakran „nan”-nak, vagy „NaN”-nak jelölnek.

2.5.2.1 A Single

A single legkisebb értéke −3,4028235E+38, a legnagyobb 3,4028235E+38. A hasznos számjegyek száma kb. 6–9 db – függetlenül attól, hogy azok a tizedesjeltől jobbra-, vagy balra helyezkednek el.

Ez rengeteg veszélyt hordoz magában: Vegyük például az 1230,00089. Itt a tizedes jegyek száma 5 db, a vesszőtől balra eső számjegyek száma 4 db – összesen 9 db. Ha ezt a számot megpróbáljuk eltárolni single típusú lebegőpontos számként, akkor (főleg a régebbi rendszerekben) semmilyen figyelmeztetést sem kapunk, a gép egyszerűen átalakítja 1230,00085-re, mert ez a legközelebbi szám, melyet a single korlátozott számú bitjein ábrázolni képes.

Itt úgy történik adatvesztés, hogy arról semmilyen figyelmeztetést sem kapunk. A lebegőpontos számok mindig valós számok (két egész hányadosaként előállíthatók), amelyek – mivel a rendelkezésre álló bitek száma korlátozott – ábrázolási hibával is terheltek lehetnek.

A probléma teljesen jelentéktelennek tűnik, de vegyünk egy triviális példát! Próbáljuk meg 32-bites, single lebegőpontos számon eltárolni az $^1/_3 = 0,\dot{3}$-ot, ahol a 0,3333… egy végtelen tizedes tört. Ez single esetében – mivel a szájegyek száma korlátozott – 0,333333343-ként jelenik meg, mert ez a legközelebbi ábrázolható érték.

Bár a gyakorlati életben ez rendszerint elfogadható, láthatjuk, hogy a lebegőpontos számokkal végzett műveletek *eredendően pontatlanok* lehetnek.

Nézzünk egy informatikai példát! Írjuk fel a 0,1-et single lebegőpontos számként. Itt szerencsénk van: a 0,1 ábrázolható 0,1-ként single-ként is. Adjuk össze a 0,1-et tízszer (0,1 + 0,1 + 0,1…)! Eredménynek egyet várunk. De arra várhatunk, mert összeadás „valódi” eredménye lebegőpontos számkezelés esetén 1,00000012!

Ugyanígy, lebegőpontos, informatikai környezetben szorozzuk meg a 0,1-ed 10-el! Eredménynek egyet várunk. És itt igazunk is lesz, mert az így kapott eredmény valóban egy – lebegőpontos környezetben is. Bár az eltérés csak a hetedik tizedesjegynél jelentkezik, a probléma nyilvánvaló: az 1 és a 1,00000012 nem egyenlő!

Nézzünk egy földrajzi példát is! A Föld alakját közelítő WGS84 ellipszoid (5. fejezet) fél kistengelyének hossza 6 356 752,314245 m. Ez már a single tárolóképességnek határán túl van, több mint hét számjeggyel, ezért tizedes jegyeket már csak korlátozottan tudnánk eltárolni, például ez a számérték „csak” a 6 356 752,5 m lesz, mert ez a legközelebbi ábrázolható érték az adott bitszámon. Az itt fellépő hiba már emberileg is értelmezhető nagyságrendű: közel 20 cm.

Itt az első probléma (0,1 + 0,1 + 0,1 … ≠ 1) abból fakad, hogy a tárolás mintája $(-1)^s \cdot m \cdot 2^k$, így már eleve nem minden szám ábrázolható hiba nélkül; a második probléma pedig a tizedesjegyek korlátozott számából fakad.

2.5.2.2 A Double

Azért, hogy csökkentsék a single-nél fellépő kerekítési hibákat, bevezették a hosszabb, 64-bites dupla pontosságú „double" típust. A double legkisebb értéke −1,7976931348623157e+308, a legnagyobb értéke 1,7976931348623157e+308. A hasznos számjegyek száma kb. 15–17 db.

A térinformatikai koordináta-geometriában szinte kizárólag ezt a típust használjuk.

2.5.2.3 A Decimal

A decimális számok – eredti típusmegjelölésükben: „decimal" – a térinformatika tárgykörében ritkán bukkannak fel, de a pénzügy területén széles körben elterjedtek. Nevük egy kicsit megtévesztő: hatványalapjuk a tíz, de ők is csak lebegőpontos számok néhány speciális tulajdonsággal.

A decimális számok 128-bitesek, 28–29 számjegyig pontosak és a $\pm 7{,}9 \cdot 10^{28}$ tartományon használhatók.

Ellentétben más lebegőpontos megoldásokkal, a $(-1)^s \cdot m \cdot 10^k$ tárolási képlet miatt a decimal ábrázolási hiba nélkül tárolja a tizedes törteket (legfeljebb a 28–29. számjegyig), így például a 0,1 hiba nélkül 0,1-ként tárolható és ha az egytizedet tízszer összeadjuk akkor *pontosan* egyet kapunk.

A decimal speciális érdekessége, hogy a többi ábrázolással ellentétben a 0,5-lefelé, nullára kerekíti.

2.5.2.4 A lebegőpontos számábrázolás veszélyei

A lebegő pontos számok eredendően pontatlan számábrázolások, mint ahogy az előbb is láttuk. Ennek azonban a térinformatikában számos negatív következménye is van.

Vegyünk egy egyszerű gyakorlati példát! Adott egy terepmodell, pl. egy GEBCO (General Bathymetric Chart of the Oceans). Ez egy alacsony felbontású raszter, amely 15″-es felbontásban ábrázolja a tengerfeneket és a szárazföldeket egyaránt. A legtöbb térinformatikai program a terepmodelleket, minimum–maximum közé feszített lineáris hozzárendelési függvény szerint ábrázolja, magyarul a mélyen fekvő részek sötétek, a kiemelkedőek pedig világosak lesznek. A jelmagyarázatban rendszerint megjelenik egy érték a minimumra és a maximumra egyaránt. Ez a két érték most a példa kedvéért legyen −106,991 m és +85,391 m. Szeretnénk a legmélyebb pontra tenni egy jelölőt, például egy piros rajzszeget. Ehhez le kellene válogatnunk a legalacsonyabban

fekvő pixelt, tehát keressük a lebegőpontosan ábrázolt pixelek sokaságából az a pixelt, melynek értéke −106,991.

Ezt a kísérletet sokszor megtehetjük egymás után, de az eredmény mindig ugyanaz: *semmi*. Tehát nincs olyan pixel, amelyik értéke egyenlő lenne −106,991-el. Mi a probléma? A szoftverekben a lebegőpontos számok –kényelmi okokból– kerekítetten jelennek meg. Itt a valódi érték −106,991 helyett −106,99099731445. És természetesen igaz, hogy −106,99099731445 ≈ −106,991, de −106,99099731445 ≠ −106,991! Így ennek megfelelően nincs is találat.

Fentebb azt is láttuk, hogyha a 0,1-et tízszerszer összeadjuk az eredmény nem egy a $(-1)^s \cdot m \cdot 2^k$ tárolási képletű lebegőpontosok esetében. Double esetében az eredmény 0,9999999999999999.

Mi a két problémában a közös? Az, hogy tökéletes tárolást / egyenlőséget feltételezünk a matematika szabályai szerint, de a kerekítési- és számábrázolási hibák miatt a lebegőpontos rendszerben nem a várt eredményt kapjuk.

Hogyan lehet ettől a problémától megszabadulni? A megoldás egy kicsit kiábrándító: sehogy. Azokat a lebegőpontos számokat, melyeken bármilyen matematikai műveletet végeztünk (kerekítés, alapműveletek, hatványozás, szögfüggvények, logaritmus stb.) *nem lehet egyenlőségre tesztelni*.

Helyette azt kell vizsgálni, hogy beleesik-e egy előre maghatározott pontosság-kritériummal rendelkező tartományba. Tételezzük fel, hogy a relatív toleranciánk 64-bites lebegőpontos számok esetében $1 \cdot 10^{-9}$. Ekkor a következő kiértékelést kell elvégezni: „$|a-b| \leq 1 \cdot 10^{-9} \cdot \max(|a|, |b|)$”, ahol ‘a’ & ‘b’ az összehasonlítandó számok, ‘|’ az abszolútérték jele, a ‘max’ függvény pedig a két argumentuma közül a nagyobbat választja.

Ha behelyettesítjük az 1-et és a 0,9999999999999999-et, akkor az eredmény igaz, mert a reláció bal oldalán 1,1102230246251565E-16, a jobb oldalán pedig 1e-09 lesz, és ez esetben az egyenlőtlenség igaz, mert 1,1102230246251565E-16 kisebb, mint 1E-09.

Ugyanezt a tesztet elvégezhetjük a domborzatmodelles példánk esetében is. Itt az ‘a’ = −106,991, a ‘b’ pedig egy változó, mely az aktuális pixel értékét tartalmazza.

Mivel a −106,991 három tizedesjegyre kerekített, ezért itt a $1 \cdot 10^{-9}$-es tolerancia értékét $1 \cdot 10^{-3}$, de inkább $0{,}5 \cdot 10^{-4}$-re kell állítani. (Mivel domborzatmodellről beszélünk a két számérték közötti különbség gyakorlatilag lényegtelen.) Ekkor már a két számérték egyenlőnek *tekinthető*.

Természetesen, ha ismerjük a legalacsonyabban fekvő pixel egzakt, *kerekítetlen* értékét konstansként (−106,99099731445), vagy ezt pontosan le tudjuk kérdezni, akkor továbbra is tesztelhetünk egyenlőségre.

Bár az előbb leírt képlet [$|a-b| \leq 1 \cdot 10^{-9} \cdot \max(|a|, |b|)$] meglehetősen bonyolultnak hat, az esetek többségében a szoftverek és programnyelvek kínálnak beépített alternatívát (pl. a Python™-ban a math.isclose() függvény), térinformatikai környezetekben pedig a raszteralgebra jelent gyors megoldást (pl.: minCell = (r == r.minimum), ahol 'r' az aktuális raszter, a 'minimum' pedig a raszter beépített tulajdonsága a valós, kerekítés nélküli érékkel).

Ezen bekezdés tartalmának ismerete nem feltétlenül szükséges a térinformatika tárgykörében, de segítséget nyújthat megérteni az okokat, ha például egy vetületi átszámítás után váratlan eredményeket kapunk. Lássuk a szabályokat:

- A lebegőpontos számkezelésben +0 és −0 is létezik. Ezek egyenlők egymással (+0 = −0);
- $\frac{1}{+0} = +\infty$; $\frac{1}{-0} = -\infty$ (ezért van szükség mindkét nullára);
- $+\infty = +\infty$; $-\infty = -\infty$; $+\infty \neq -\infty$; $+\infty > -\infty$;
- $\frac{1}{+\infty} = 0 = \frac{1}{-\infty} = \frac{-1}{-\infty} = \frac{-1}{+\infty}$;
- $\infty-\infty$; $-\infty+\infty$; $0\times\infty$; $0\div0$; $\infty\div\infty \Rightarrow$ NaN (not-a-number);
- $x + \infty = \infty$
- NaN ≠ NaN A NaN nem egyenlő semmivel – még önmagával sem. A NaN nem kisebb és nem nagyobb semminél sem, beleértve a pozitív- és a negatív nullát, valamint a pozitív- és a negatív végtelent, valamint önmagát is.
- Ha egy műveleti sorba NaN kerül, az eredmény mindig NaN.
- A double legnagyobb felvehető értéke 1,7976931348623157E+308. Ha ehhez bármilyen pozitív számot hozzáadunk az eredmény 1,7976931348623157E+308 (önmaga). Ha a 1,7976931348623157E+308-t bármilyen egynél nagyobb számmal szorozzuk, az eredmény pozitív végtelen.
- Néhány programnyelv és processzor különbséget tesz kétféle NaN között és fentart egy Signaling NaN-t (SNaN), melynek felbukkanása hibaüzenethez vezet és egy Quiet NaN-t (QNaN), mely hiba nélkül fut végig a számítási soron (persze az eredmény NaN lesz).

2.5.3 Számírási megoldások a programozásban

Az informatikában, főleg a programozás területén négy főbb számírási megoldás terjedt el:

1. A hagyományos, tízes számrendszerbeli lejegyzési mód, a legelterjedtebb ábrázolásai forma. Itt a számokat a szokásos módon írjuk, programnyelvekben *minden esetben tizedesponttal*.
   A normálalakú számokat az 'E', vagy 'e' karakter közbeiktatásával írjuk, pl.: $1.4 \cdot 10^5$ = 1.4e5. Az 'e' jelölés után csak egész számok következhetnek, akár vezető nullával és előjellel. Így az 1.45e005, az 1.45e+5 és az 1.45E+005 is érvényes ábrázolási forma.

Szükség esetén a „tiszta” normálalaktól eltérhetünk és írhatunk 0.5e-4-et az 5e-5 helyett. A programnyelvekben a jobb átláthatóság végett a számjegyek tagolhatók alsóaláhúzással: 4096 helyett írhatunk 4_096-ot, de 3.141592653589793 helyett 003.141_592_653_589_793e000-t is. A tagolás az ‘e’ után is használható: 1e1_024.

2. A hexadecimális, tizenhatos számrendszerbeli lejegyzési mód a második leggyakoribb lejegyzési forma. Igen nagy számok kompakt formában, kevés helyiértéken írhatók le. Igen népszerű memóriacímekben és hibaüzenetekben. Írásukat mindig a ‘0x’ előtag vezeti be, így különíthetők el a hagyományos tízes számrendszerbeli számoktól.
   Érvényes hexadecimális szám például a 0x1FB (507_{10}). Az előjelet a ‘0x’ előtag elé írjuk: -0xFF.
   A hexadecimális számrendszerben alkalmazott betűket formázó számkarakterek kis és nagybetűs változataikban is leírhatók, akár vegyesen is, pl.: +0x7Fa.
   A hexadecimális számok is tagolhatók alsóaláhúzással. Sok programnyelv fordítóprogramja csak egész számokat támogatja hexadecimális formában, míg néhány rendszer a lebegőpontos számokat is elfogadja egy speciális formátumban pl.: 3.141592653589793 = 0x1.921fb54442d18p+1, ahol a szám végén, a kitevőnél ‘p’ szerepel.
   A hexadecimális számrendszer érdekessége, hogy a bájt legnagyobb lejegyezhető 255-ös értékét 0xFF-ként ábrázolja, ezért alkalmas bájtsorok (pl.: színkódok) rövid lejegyzésére.
3. Ritkább, de használt megoldás a bináris megírás is. Itt a ‘0b’ bevezető karakterek után a bitek következenek, pl.: 0b101010 (42_{10}). A bináris megírás – a számrendszer jellegéből fakadóan – csak az egész számokat támogatja.
4. Igen ritka megoldás a nyolcas, oktális számrendszerbeli jelölés használata. Ekkor a ‘0o’ (nulla, majd ‘o’ betű), pl.: 0o47 (39_{10}). Csak az egész számokat támogatja.

Az exponensírás a hexadecimális, oktális és binárisan lejegyzett számokkal nem kombinálható.

2.5.4 Szöveg → szám konverzió

Idáig áttekintettük a karakter- és a számábrázolást egyaránt. Az első esetben a lejegyzett binárisan tárolt bájtok karaktereket reprezentálnak, mint például ennek a tankönyvnek a betűit, második esetben ugyanezen bináris bájtok egész- vagy lebegőpontos számokat jelölnek. A különbségtétel csak szemléletbeli: az eredmény attól függ, hogy a felhasználó vagy a szoftver minek látja az adott bájthalmazt.

Ebből a kettőségből fakadóan a számokat akár szövegesen, karakterre kódoltan is tárolhatjuk. Hogyan lehet egy szövegként tárolt számot valódi számmá alakítani? Vegyünk például egy egyszerű egészszámot, a 3-at, melyet tároljunk el szövegként és jegyezzük le bájtként a memóriába!

Ehhez először el kell döntenünk, hogy milyen kódlapot használunk. Az egyszerűség kedvéért használjuk az ASCII-t! A kódlapokhoz mindig tartozik egy táblázat, hogy mely vizuálisan megjelenő karakterhez milyen számot kell rendelni. Az ASCII táblázata itt látható (4. táblázat):

4. táblázat. ASCII karakterek és a hozzájuk tartozó bájt értékek decimális (№) és bináris formában a [33; 126]-os tartományon. A 0–32-es tartomány a 5. *táblázatban látható (Oualline, 1997).*

№	Bináris reprezentáció	Szimbólum	Angol megnevezés
33	00100001_2	!	Exclamation mark
34	00100010_2	"	Double quotes (or speech marks)
35	00100011_2	#	Number sign
36	00100100_2	$	Dollar
37	00100101_2	%	Per cent sign
38	00100110_2	&	Ampersand
39	00100111_2	'	Single quote
40	00101000_2	(	Open parenthesis (or open bracket)
41	00101001_2	)	Close parenthesis (or close bracket)
42	00101010_2	*	Asterisk
43	00101011_2	+	Plus
44	00101100_2	,	Comma
45	00101101_2	-	Hyphen-minus
46	00101110_2	.	Period, dot, or full stop
47	00101111_2	/	Slash or divide
48	00110000_2	0	Zero
49	00110001_2	1	One
50	00110010_2	2	Two
51	00110011_2	3	Three
52	00110100_2	4	Four
53	00110101_2	5	Five

54	00110110_2	6	Six
55	00110111_2	7	Seven
56	00111000_2	8	Eight
57	00111001_2	9	Nine
58	00111010_2	:	Colon
59	00111011_2	;	Semicolon
60	00111100_2	<	Less than (or open angled bracket)
61	00111101_2	=	Equals
62	00111110_2	>	Greater than (or close angled bracket)
63	00111111_2	?	Question mark
64	01000000_2	@	At sign
65	01000001_2	A	Uppercase A
66	01000010_2	B	Uppercase B
67	01000011_2	C	Uppercase C
68	01000100_2	D	Uppercase D
69	01000101_2	E	Uppercase E
70	01000110_2	F	Uppercase F
71	01000111_2	G	Uppercase G
72	01001000_2	H	Uppercase H
73	01001001_2	I	Uppercase I
74	01001010_2	J	Uppercase J
75	01001011_2	K	Uppercase K
76	01001100_2	L	Uppercase L
77	01001101_2	M	Uppercase M
78	01001110_2	N	Uppercase N
79	01001111_2	O	Uppercase O

80	01010000_2	P	Uppercase P
81	01010001_2	Q	Uppercase Q
82	01010010_2	R	Uppercase R
83	01010011_2	S	Uppercase S
84	01010100_2	T	Uppercase T
85	01010101_2	U	Uppercase U
86	01010110_2	V	Uppercase V
87	01010111_2	W	Uppercase W
88	01011000_2	X	Uppercase X
89	01011001_2	Y	Uppercase Y
90	01011010_2	Z	Uppercase Z
91	01011011_2	[	Opening bracket
92	01011100_2	\	Backslash
93	01011101_2	]	Closing bracket
94	01011110_2	^	Caret - circumflex
95	01011111_2	_	Underscore
96	01100000_2	`	Grave accent
97	01100001_2	a	Lowercase a
98	01100010_2	b	Lowercase b
99	01100011_2	c	Lowercase c
100	01100100_2	d	Lowercase d
101	01100101_2	e	Lowercase e
102	01100110_2	f	Lowercase f
103	01100111_2	g	Lowercase g
104	01101000_2	h	Lowercase h
105	01101001_2	i	Lowercase i

106	01101010_2	j	Lowercase j
107	01101011_2	k	Lowercase k
108	01101100_2	l	Lowercase l
109	01101101_2	m	Lowercase m
110	01101110_2	n	Lowercase n
111	01101111_2	o	Lowercase o
112	01110000_2	p	Lowercase p
113	01110001_2	q	Lowercase q
114	01110010_2	r	Lowercase r
115	01110011_2	s	Lowercase s
116	01110100_2	t	Lowercase t
117	01110101_2	u	Lowercase u
118	01110110_2	v	Lowercase v
119	01110111_2	w	Lowercase w
120	01111000_2	x	Lowercase x
121	01111001_2	y	Lowercase y
122	01111010_2	z	Lowercase z
123	01111011_2	{	Opening brace
124	01111100_2	|	Vertical bar
125	01111101_2	}	Closing brace
126	01111110_2	~	Equivalency sign - tilde

A fentebbi táblázatból látható, hogy a karakterként ábrázolt ‘3’ értéke 51 a tízes számrendszerben. Tehát ha van egy karakteresen kódolt, szövegként ábrázolt ‘3’-as értékünk, akkor gyakorlatilag egy 51-es számértéket tartalmazó bájtunk van.

Ugorjunk vissza az eredeti kérdéshez: „Hogyan lehet a szövegként tárolt számot valódi számmá alakítani?” Az eljárás egyszerű. Mivel az ASCII kódtáblán a számkarakterek egymás után növekvő sorrendben következnek ezért a konverzióhoz meg kell határoznunk a nullás számkarakter értékét! Ez esetben 48. Így, ha a kezdő számkarakter értékét kivonjuk az aktuális számkarakter értékéből, akkor megkapjuk azt a tízes számrendszerbeli számot, amelyet a karakter ábrázol. Nézzük meg a 3-as példánkra: 51 – 48 = 3.

Nézzük meg egy bonyolultabb számra, például az 561-re! Ekkor ki kell keresni a 0-s számkarakter értékét (48), az 1-es számkarakter értékét (49), a 6-os számkarakter értékékét (54), végül az 5-ös számkarakter értékét (53). Ha az értékek megvannak, jobbról-balra helyiértékek szerint el kell végezni az alábbi műveletet:

$$(49 - 48) \cdot 10^0 + (54 - 48) \cdot 10^1 + (53 - 48) \cdot 10^2$$

$$1 \cdot 1 + 6 \cdot 10 + 5 \cdot 100$$

$$561$$

Az eljárást tisztán binárisan szemlélve a következő transzformációt hajtjuk végre: 00110101_00110110_00110001 → 00000010_00110001.

Az eljárás lebegőpontos számokra is általánosítható, annyi specialitással, hogy itt a számrendszer alapjának kitevőjét a tizedesponttól jobbra negatív értékkel kell figyelembe venni, például az 56,1-ed az alábbi minta szerint kell átalakítani:

$$(49 - 48) \cdot 10^{-1} + (54 - 48) \cdot 10^0 + (53 - 48) \cdot 10^1$$

$$1 \cdot 0{,}1 + 6 \cdot 1 + 5 \cdot 10$$

$$56{,}1$$

Szövegként ábrázolt tizedes törtek szöveg→szám konverziója esetében vegyük figyelembe, hogy a különböző nyelvek különböző tizedesjeleket használnak, például a magyar vesszőt, de az angol pontot. Az átalakításért felelős, fent bemutatott eljárás az adott nyelvi beállításoknak megfelelő tizedesjelet fogja keresni. Ha ezt nem találja, a kapott szám vagy hibás lesz, vagy hibaüzenetet kapunk. A gyakorlatban ez azt jeleni, hogy az 56,1 (magyar tizedesjellel) konverziója sikertelen lesz angol nyelvi beállításokkal, mert a számban nem található meg a tizedespont.

2.5.5 Bájtsorrend, az „Endianness”

Ha lejegyzünk egy számot, akkor azokat a számokat, melyek nagyobbak, mint 255_{10}, több mint egy bájton kell ábrázolnunk. Vegyük például az 1648_{10}-at. Az 1648_{10} bináris reprezentációja két

bájton $00000110\_01110000_2$. A megoldás kézenfekvő: a helyiértékek jobbról balra növekednek, ahogy a „hagyományos” számírásunknál is megszokhattuk. Ilyenkor a legjelentősebb bájt a bináris számban a bal oldalon van, a legkevésbé jelentős bit pedig a jobb oldalon, mert ha a bal bájt bal oldali bitjét cseréljük ki nulláról egyre, akkor az eredmény 34 416_{10}-lesz, míg, ha a jobb oldali bájt jobb bitjét, akkor 1649_{10}. Így látható, hogy a bal oldali jelentősebb, mint a jobb oldali.

Ez az eljárás azonban nem szükségszerű! Mint ahogy vannak jobbról-balra és balról-jobbra haladó írásrendszerek, így a bájtokat is leírhatjuk jobbról-balra és balról-jobbra egyaránt.

Az előbb bemutatott példaszám a $00000110\_01110000_2$ egy „big-endian” ábrázolás, mert a bal „végén” van a legjelentősebb bájt.

Az eljárás ellentettje a „little-endian”, amikor ugyanezt a számot így jegyezzük le: $01110000\_00000110_2$. Ekkor a bal „végén” található a legkevésbé jelentős szám (csak a bájtok sorrendét cseréltük fel, nem a bitekét).

Két eljárás alkalmazása egy és ugyanazon feladatra meglehetősen feleslegesnek és veszélyesnek tűnik, de sajnos történetileg úgy alakult, hogy a jelenleg használt processzorok egy része az egyik, más részük a másik megoldást használja – miközben a két eljárás között sem gyakorlati, sem elvi különbség nincs.

- Little-endian architektúrák: x86; x87; x64; ARM®; RISC-V®; Apple® silicon.
- Big-endian architektúrák: PowerPC™; Motorola®; hálózati céleszközök.

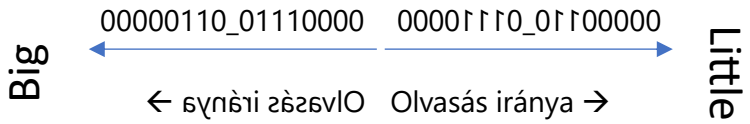
A processzor-architektúráktól függetlenül a hálózati infrastruktúra mindig a big-endian bájtsorrendet használja.

A bájtsorrend hatással van a számábrázolásra, a több-bájtos UTF karakterkódolásra, számos térinformatikai raszterformátumra (pl. TIFF – Tagged Image File Format) és vektoros adatcsereformátumra (ESRI Shapefile – Environmental System Research Institute Shapefile) is. Gyakori probléma, hogy a hálózati infrastruktúra felől érkező több-bájtos számformátumok big-endian, míg a mobiltelefonjaink és számítógépeink nagy része pedig little-endian bájtsorrendet követ.

A két típus között egyszerűen a bájtok sorrendjének megfordításával lehet váltani, amely rendszerint beépített szolgáltatás a legtöbb programnyelvben, de a több-bájtos UTF-8 kódpontok miatt az eszköz a fejlettebb szövegszerkesztőkben is megtalálható.

2.5.5.1 Bitsorrend

A bitsorrend a gyakorlatban alacsony jelentőségű, de segít megérteni a bájtsorrend okát. Ha a bitsorrendre is alkalmazzuk a little-/big-endian szemléletet, akkor a számok lejegyezve így néznek ki (mindkettő az előbbi 1648 bináris reprezentációja, 2. ábra):



2. ábra. Az 1648_{10} big- és little-endian reprezentációja.

Tehát a big-endian esetében a számokat jobbról balra írjuk és a helyiértékek balra nőnek, még a little-endian esetében jobbról balra írunk és a helyiértékek jobbra nőnek. A két személet között szemmel láthatóan nincs érdemi különbség, csak a növekmény iránya.

A little-endian architektúrák rendszerint egyszerre little-endian bit- és bájtsorrendűek, akár csak a big-endian rendszerek egyszerre big-endian bit- és bájtsorrendűek.

Jelezvén a probléma vérre menő, de komikus jelentéktelenségét a két bájtsorrend elnevezése Swift Gulliver utazásaiból származik, ahol a tojást a vastag végén törők és a hegyes végén törők rendszeresen irtották egymást.

A helyzetet tovább bonyolítja, hogy az informatikai rendszerek a little-endian bitsorrendű bájtokat is big-endian-ként nyomatják a képernyőre, hogy alkalmazkodjanak a nyugati számírási szokásokhoz.

2.5.6 Műveletek bináris számokon

A bináris számok jellegéből fakadóan, hogy számjegyeik egyszerű igen/nem kapcsolók olyan műveleteket is végezhetünk melyek a tízes számrendszerben nem megszokottak.

Az első ilyen művelet a bitek csúsztatása. Vegyünk egy egyszerű bájtot, melynek értéke legyen 17_{10}, melynek bináris reprezentációja $0001\ 0001_2$. A bináris számokat a processzorok jobbra- illetve balra el tudják csúsztatni, amely egy igen gyors és hatékony művelet.

A balra történő csúsztatás operátora a „<<”. Próbáljuk meg a $0001\ 0001_2$-t egyel balra csúsztatni! $0001\ 0001_2 << 1_2 = 0010\ 0010_2$. Itt a bal oldali kifejezés a csúsztatandó szám, a jobb oldali pedig a csúsztatás mértéke. Tízes számrendszerben kifejezve $17_{10} << 1_{10} = 34_{10}$. (Előjel nélküli egészeknél a balra csúsztatás a kettővel való szorzással egyenértékű.)

A bitcsúsztatás során legalább egy bit valamelyik oldalról „bejön”, ennek az értéke minden esetben ‘0’, mint ahogy a $0001\ 0001_2$ mögé is egy nulla fűződött és $0010\ 0010_2$-t (34_{10}) kaptunk. A másik oldalon a „felesleges” bitek „leesnek” és eltűnnek. A csúsztatás mértéke logikailag korlátozott, rendszerint az adott változó bitszámánál eggyel kisebb számig terjed, például egy 32-bites

egészszámot legfeljebb 31-el lehet csúsztatni – de ez processzor típustól függ. A balra csúsztatással rokon művelet a jobbra csúsztatás, melynek jele „>>”, mely minden tekintetben úgy zajlik, mint a balra csúsztatás, csak a bitek az ellenkező irányba mozognak.

A második család a bitenkénti logikai műveletek:

- Bitenkénti „nem”. Jele „~”. Ez a művelet minden bitet az ellenkezőjére fordít, pl.: $\sim 1010\ 1010_2 = 0101\ 0101_2$. Angol megnevezése: „NOT”.
- Bitenkénti „és”. Jele „&”. Ez a művelet két bináris szám bitjein páronként logikai „és” műveletet végez, pl.: 0110_2 & $0100_2 = 0100_2$ (mert csak egy helyen volt igaz, hogy mindkét bit ‘1’). Gyakran használják több-bájtos UTF-8 karakterek felismerésére, mert a több-bájtos karaktereket $110\ldots_2$, $1110\ldots_2$, $11110\ldots_2$ előtaggal vezetik be. Így például, ha igaz, hogy 1100_0000_2 & 1100_0011_2 egyenlő 1100_0000_2-vel, akkor biztos, hogy egy kétbájtos UTF-8 karaktert olvasunk, és biztos, hogy az aktuális bájt után kell jönni még egy bájtnak. A példa egyébként a magyar ‘á’ (1100_0000_2 1010_0001_2) betű UTF-8 kódolt első bájtja. Angol megnevezése: „AND”.
- Bitenkénti „vagy”. Jele: „|”. Ez a művelet két bináris szám bitjein páronként logikai „vagy” műveletet végez, pl.: 1010_2 | $0101_2 = 1111_2$. Angol megnevezése: „OR”.
- Bitenkénti „kizáró vagy”. Jele „^”. Ez a művelet két bináris szám bitjein páronként logikai „kizáró vagy” műveletet végez, pl.: $1111\ 1000_2$ ^ $0001\ 1100_2 = 1110\ 0100_2$. (Az azonosak ‘0’-ra váltanak, másutt logikai „vagy”-ként működik.) Angol megnevezése „XOR”. A XOR egyik gyakori használata, hogy segítségével a változók nullázhatók, mert ha egy tetszőleges változót önmagával XOR-ozzuk, akkor az eredmény mindig nulla. Például 101010_2 ^ $101010_2 = 0_2$

Az itt bemutatott bináris műveletek meglehetősen öncélúnak tűnnek, de számos gyakorlati hasznuk van az informatikában. Vegyük például azt a gyakori tesztet, hogy egy adott szám páros-, vagy páratlan-e. Ezt a tesztet rendszerint maradékos osztással végeznénk. Ha van maradék, akkor páratlan, ha nincs, akkor páros. De az osztás egy számítás igényes művelet ezért jó lenne elkerülni. A páratlan számok bináris reprezentációjában az utolsó számjegy mindig 1_2. Ebből következik, hogy ha n & 1_2 nulla, akkor a szám páros, ellenkező esetben páratlan. Nézzük meg a gyakorlatban! Binárisan: 0011_2 & $0001_2 = 0001_2$; ugyanez tízes számrendszerben: 3_{10} & $1_{10} = 1_{10}$ – tehát a három páratlan, mert az eredmény nem nulla.

Az ASCII karakterek bájtokkal reprezentáltak. Ebből fakadóan néhány karakter-manipulációs műveletet könnyen elvégezhetünk bitenkénti műveletekkel.

Például, ha bármely nagybetűnek keressük a kicsi párját, akkor csak egy „|” műveletet kell elvégeznünk rajta a 32_{10}-es konstanssal (amely a szóköz értéke). Így például 'H' | ' ' = 'h'. Ugyanez

tízes számrendszerben: 72_{10} | 32_{10} = 104_{10}. És binárisan: $0100\ 1000_2$ | $0010\ 0000_2$ = $0110\ 1000_2$.

2.5.7 Számítások lebegőpontos koordinátákkal

Ha koordinátákkal összefüggő, például vetületi-, vagy transzformációs számításoknál lebegőpontos számok használatára szorulunk – akkor minden esetben (legalább) 64-bites számokkal kell dolgozni. (Az egészszám alapú számítási megoldást a „Koordináta-ábrázolás II." mutatja be).

Ez egy meglehetősen sarkos kijelentés, ezért vizsgájuk meg az igazságtartalmát!

Az északi szélesség 45. fokán állunk. Számoljuk ki csak és kizárólag 32-bites lebegőpontos számok segítségével a 45. foknak megfelelő, metrikus Web Mercator vetület szerinti 'y' értéket!

Itt $y = 6378137{,}0\ m \cdot ln\ (tan[45° + \frac{\varphi}{2}])$, ahol a 6 378 137,0 m a WGS84 ellipszoid fél nagytengelyének hossza, így behelyettesítve $= 6\ 378\ 137{,}0\ m \cdot ln\ (tan[67{,}5°])$. Kizárólag 32-bites lebegőpontos függvényeket használva az eredmény 5 621 522 méter kereken. Az érték reális, mert a 32-bites single kb. 6–9 számjegy tárolására képes, miközben az eredmény egészrésze önmagában hét számjegyű.

Ha ugyanezt a számítást elvégezzük 64-bites double-el is, akkor az eredmény 5 621 521,4862 m, így a hiba több mint fél méter.

A keletkezett hiba oka kettős: egyrészt a fok-radián konverzió során elveszítjük a hasznos tizedesjegyek egy részét, így a hiba tovább terjed a szögfüggvény felé, ahol ismét kerekítési hiba képződik, mely tovább lép a logaritmus felé. Az így kapott számot felszorozzuk egy milliós nagyságrendű számmal, miközben az értékes tizedesjegyek ismét kiszorulnak.

Ennek analógiájaként a WGS84 ellipszoid ~40 075 016,685 egyenlítői kerületét figyelembe véve keleti hosszúság 100., a szélesség nulladik fokán állva a százat követő első reprezentálható 32-bites lebegő pontos érték a 100,000008°. A két pont közel 85 centiméterre van egymástól, ezért a single az esetek többségében nem alkalmas ennél nagyobb pontosságigényű koordináta-számításokra.

A megfelelő számtípus kiválasztása esetén is fel kell készülnünk néhány gyakori, gyakorlati hibára.

A lebegőpontos számok binárisan, mantissza és kitevő formában tároltak, az $m \cdot 2^k$ alakban, ahol az 'm' a mantissza, 'k' a kitevő. A lebegőpontos számok mantisszájának tárolása miatt a számegyenes felosztása nem lineáris, a nagyobb helyiértékű számok „kiszorítják" az alacsonyabb helyiértékűeket.

Ez a gyakorlatban azt jeleni, hogy két, azonos nagyságrendű szám esetén az eredmény a naivan várt érték, például 1E-12 + 1E-12 = 2E-12 – ahogy azt elvárnánk. De abban az esetben, ha a Föld alakját közelítő WGS84 ellipszoid fél nagytengelyének méterben kifejezett hosszához akarjuk az 1E-12-t hozzáadni, akkor az eredmény meglepő: 6 378 137 + 1E-12 = 6 378 137 (pontosan, nem kerekítve), mivel a használható 15–17 db értékes számjegy után az 1E-12 kiszorul, így pontosságvesztés történik. Ez a jelenség a skála-probléma. A skála-probléma akkor lép fel, ha két, különböző nagyságrendű (különböző skálájú) számon végzünk matematikai műveletet.

Ebből fakadó rokon probléma a műveletek felcserélhetősége: ha 64-bites lebegőpontos számokkal dolgozunk és elvégezzük a „6 378 137 + 4e-10 + 4e-10” műveletet, akkor az előbbi példa analógiájaként 6 378 137-t kapunk, mert nem volt elegendő számjegy az eredmény ábrázolására. A hiba kétszer ismétlődik: az első műveletnél 6 378 137-t, majd a másodiknál szintén 6 378 137-t kapunk. De írjuk át a műveletet „6 378 137 + (4e-10 + 4e-10)”-re! Ekkor az eredmény 6 378 137,000000001. A matematikában ugyan ezen műveletek felcserélhetők, de a számábrázolás gyakorlati hiányosságai miatt ez csak akkor igaz, ha műveletek matematikailag felcserélhetők *és azonos nagyságrendűek* is egyben. Hasonló összegzési problémák kezelésére segítséget nyújthat a Kahan–Babuška-algoritmus. Abban az esetben ha sok, különböző nagysárendű komponenst kell összegeznünk, akkor – a matematikai felcserélhetőség végett – rendezzük őket növekvő sorrendbe és úgy összegezzük, hogy a legnagyobb elem a műveleti sor végére kerüljön.

2.5.8 Casting és numerikus re-interpretáció

A binárisan tárolt számok típusa csak metaadatnak minősül: egy 64-bites egészszám és egy 64-bites lebegőpontos szám között mindössze annyi a különbség, hogy másként tekintünk rá, a műveleteket másképp végezzük el rajtuk, de a memóriában reprezentációjuk azonos. Ebből kifolyólag kétféleképpen is egymásba alakíthatók.

Érték szerinti átalakítás (casting): ennek során az egyik számtípust a másikba alakítjuk, úgy, hogy kísérletet teszünk az eredeti számérték megőrzésére. Ez az áttérés veszteségesen és veszteségmentesen is megtörténhet. Nézzük először a veszteségmentes forgatókönyveket!

- Első és leggyakoribb veszteségmentes érték szerinti konverzió az egészszám→egészszám átalakítás. Ekkor egy olyan nagyobb bitszámú egészszám típusból alakítunk át egy kisebb bitszámú egészszám típusba, hogy a kisebb bitszámú számba az érték „belefér”. Például, ha 16-bites, előjel nélküli egészszámként ábrázoljuk a ‘6’-ot, akkor az veszteségmentesen átalakítható bájttá (mert a bájt $[0; 255] \in \mathbb{Z}^+$ számok ábrázolására alkalmas). Fontos, hogy a céloldali egészszám típusba „beleférjünk”, ellenkező esetben vad dolgok

történnek: próbáljuk meg a 16-biten ábrázolt 255-öt bájttá alakítani. Mivel a célformátumba „beleférünk” ezért nem történik veszteség, az eredmény 255, immáron bájtként. De tegyünk kísérletet a 256 16-biten tárolt egészszám bájttá alakítására. Az eredmény 0, bájton ábrázolva.
Mi történhetett? Próbáljuk meg a 257-et is! Az eredmény 1, bájton ábrázolva. Ebből már látjuk, hogy binárisan az alacsony helyiértékű 8 elemű bitcsoportot kapjuk meg bájtként. Numerikusan kifejezve:

$^{16\text{-bit}}255_{10}$ ($^{16\text{-bit}}00000000\ 11111111_2$) → $^{8\text{-bit}}255_{10}$ ($^{8\text{-bit}}11111111_2$);
$^{16\text{-bit}}256_{10}$ ($^{16\text{-bit}}00000001\ 00000000_2$) → $^{8\text{-bit}}0_{10}$ ($^{8\text{-bit}}00000000_2$);
$^{16\text{-bit}}257_{10}$ ($^{16\text{-bit}}00000001\ 00000001_2$) → $^{8\text{-bit}}1_{10}$ ($^{8\text{-bit}}00000001_2$).

Vannak program nyelvek, melyek ilyen veszteség esetén figyelmeztetik a felhasználót, de vannak olyanok is, melyek elhallgatják a konverziós veszteséget, ezért fontos, hogyha számátalakítási kétségeink vannak, akkor járjunk utána, hogy az adott szoftver figyelmeztet, vagy hibaüzenet nélkül hagyja el az értékes biteket. Az itt bemutatott jelenség nem csak a 16-bit→8-bit, hanem a 32→16, 64→32, 32→8, 64→16 stb. konverziókra is érvényes. Tehát létezik veszteségmentes egész–egész átalakítás, de csak akkor, ha forrásérték „belefér” a célformátum keretei közé.

- Speciális eset, ha előjeles egészszámból előjel nélküli egészszámtípusra térünk át. Itt is, mindaddig, amíg a célformátumba „beleférünk” addig a konverzió veszteségmentes, de mihelyst ezt elkerüljük, fura dolgok történnek.
  Ha például az előjel nélküli $^{16\text{-bit}}42_{10}$-t át akarjuk alakítani előjeles bájttá nem történik semmi: az eredmény 42. De ha az előjeles, negatív $^{16\text{-bit}}-42_{10}$-t akarjuk átalakítani előjel nélküli 16-bites számmá az eredmény 65 494, mert a −42-t binárisan, 16-biten így ábrázoljuk: 1111111111010110_2. Ez egy teljesen érvényes, előjel nélküli egészszám, mely közvetlenül 65 494-nek értelmezhető.
- Külön kategóriát képeznek az egészszám→lebegőpontos szám konverziók. Ezek veszteségmentesek, vagy szélső esetekben alacsony veszteségűek.
- Az informatikában a legtöbb problémát a lebegőpontos→egészszám konverziók okozzák, ilyenkor, ha vannak tizedes jegyek, akkor azok csonkolódnak. Így például a 3,14 egészszámmá alakítva 3, a −3,14 pedig −3 lesz. De fontos tudni, hogy a 3,99 is 3, a −3,99 is −3 lesz, tehát a tizedes jegyek csonkolódnak (nem kerekítünk), úgy, hogy mindig a nullához kerülünk közelebb.

Természetesen lehetőség van a csonkolásnál intelligenseb átalakításra is (például kerekíthetünk, úgy, hogy a nullától távolodjunk, úgy, hogy a nullához közeledjünk; úgy hogy a pozitív-, vagy negatív végtelenhez közeledjünk).

Másik, ritkán használt átalakítási forma a re-interpretáció. Ilyenkor kísérletet sem teszünk érték szerinti átalakításra, egyszerűen a biteket csak másként értelmezzük. Az eljárás gondolata furcsa, de informatikailag praktikus, például az adatbázis-kezelésben. Itt, ha lebegőpontos számokat kell sorba rendezni, akkor elegendő, hogyha binárisan, egészszámként értelmezzük őket és az egészszámokat rendezzük sorba. Például a 32-bites, 7,2 egészszámként re-interpretált értéke 1088841318, a 7,3-é 1089051034. A kapott számok értelmezhetetlen nagyságrendűek, de arányosak, és így sorba rendezhetők – anélkül, hogy speciális, lebegőpontos algoritmusokat használnánk. A re-interpretáció csak azonos bitszámú változók között történhet.

2.6 Adattárolási modellek

Az informatikában és így a térinformatikában is az adatok tárolása kettes számrendszerben, binárisan történik: az adatokat bitre kódoltan tároljuk el – legyen az szöveg, egész-, vagy lebegőpontos szám – tehát egy egyszerű eszközt használunk többféle célra, miközben a háttérben működő adattároló és -feldolgozó rendszer azonos.

Tehát egy byte-on ábrázolt ASCII karaktert tekinthetünk karakternek, például egy ‘t’ betűnek, de tekinthetjük 116_{10}-nak is a tízes számrendszerben, hiszen mindkettő binárisan kifejezve 1110100_2. Ez csak rajtunk, az adat szemlélőin múlik, hogy ugyanazt a bináris számsort hogyan látjuk, vagy láttatjuk a képernyőn. A számítógép számára beolvasni, vagy feldolgozni egy byte-ot pont ugyanaz- és pont ugyanolyan munka, mint feldolgozni egy ASCII karaktert. A különbséget csak mi látjuk, mert egy speciális függvényen, szűrőn, „szemüvegen”, vagy „tudatállapoton” keresztül nézzük az adatokat.

Innentől kezdve minden elvont informatikai- és térinformatikai adatábrázolásunk az előbb megismert eszközökre vezethető vissza, így az egészszám, a lebegőpontos szám és a szöveg ábrázolására, ezek közül pedig az összes végül visszavezethető a bináris számábrázolásra.

A térinformatikai rendszerekben szükségünk van koordináták tárolására, amely végső soron egy számpár, legyen az egész-, vagy lebegőpontos szám. Ezzel ismét egy látszólagos modellt alkottunk: megfogtunk két teljesen közönséges, például 32-bites lebegőpontos számot és azt mondtuk rá, hogy ezek innentől kezdve összetartoznak és az első tagot tekintsük ‘X’-nek a másodikat pedig ‘Y’-nak. A háttérben persze továbbra is csak két közönséges számot tárolunk, de innentől kezdve már összetartozónak tekintjük őket, ezzel létrehozva az első komplex térinformatikai adattípusunkat a *koordinátát* (Koordináta-ábrázolás I.).

Egy koordináta „pontnak” tekinthető, de mit tegyünk, ha szeretnénk modellezni egy gázvezeték nyomvonalát?

Egy nyomvonal felfogható úgy, mint szakaszok láncolata: adott egy kezdő- és egy végpont, közte pedig töréspontok sokasága szerepel. A kezdő-, a vég- és a töréspontok koordinátapontoknak tekinthetők, melyeket előbb definiáltunk. Így a modellünkben tovább építkeztünk és létrehoztunk egy új térinformatikai adattárolási eljárást a „vektoros adattárolást”. A vektoros adattárolás minden esetben koordinátákra vezethető vissza és az ábrázolási igényeknek megfelelően lehet pont, vonallánc és poligon (Vektoros adatábrázolás I. & Vektoros adatábrázolás II.: a „Z” és az „M” kiterjesztés).

Néhány adattípus, például a domborzatmodellek nem ábrázolhatók hatékonyan vektoros formában. Az olyan eseteben, amikor a teret fel kell osztani rés- és átfedés mentes cellákra, akkor *raszteres adattárolást* használunk.

A rasztereket úgy is fel lehet fogni, mint táblázatokat: a táblázat egyes celláiba beírunk egy földrajzi tulajdonságot, például a tengerszint feletti magasságot, majd a táblázat – a raszter – egyik sarokpontját koordinátákkal látjuk el, hogy tudjuk, hogy hol helyezkedik el a földrajzi térben. A raszteres adatábrázolás is csak egy modell, egy felhasználói látásmód: az adat csak binárisan lejegyzett számok sokasága, amelyet a számítógép eltárol – egyszerűen egymás mögé írva. Azt már nekünk kell tudni, hogy ezt a számhalmazt hogyan kell úgy megjeleníteni a képernyőn, hogy az értelmet nyerjen.

3 Vektoros adattárolás

3.1 Koordináta-ábrázolás I.

Az egész- és a lebegőpontos számok ábrázolásának megismerése után a koordináta-ábrázolás már könnyű feladat: egyszerűen csoportosítsunk lebegőpontos-, vagy egészszámokat és máris tetszőleges koordinátákat jegyezhetünk le. A térinformatikában három, fő koordináta-rendszert használunk:

1. **Földrajzi koordináta-rendszerek**. A földrajzi koordináta-rendszerek poláris koordináta-rendszerek, amelyek két referencia irányhoz képest mért szög alapján adják meg egy pont helyét a Föld felszínén.
   A komponenseket szélességnek és hosszúságnak nevezzük és ‘φ’-vel és ‘λ’-val jelöljük. Kiegészülhet egy harmadik, tengerszint-, vagy az ellipszoid feletti magasságot leíró értékkel. Mindhárom koordinátát rendszerint 64-bites lebegőpontos számokkal jelöljük. Ezen koordináta-rendszerrel – tárolás szempontjából – teljesen megegyeznek az **ellipszoidi koordináta-rendszerek**.
2. A **vetített-, vagy síkkoordináta koordináta-rendszerek**. A vetített koordináta-rendszerek Descartes–féle, derékszögű, síkkoordináta-rendszerek. A vetített koordináta-rendszereket úgy hozhatjuk létre, hogy a gömbbel, vagy ellipszoiddal közelíthető Föld felszínét valamilyen vetítési eljárással síkba fejtjük. Az így kapott síkon kijelölünk egy origót és az ettől mért távolságokat egymásra merőleges irányokban feljegyezzük. A komponenseket ‘x’-el és ‘y’-nal jelöljük. Az ‘x’ és az ‘y’ kiegészíthető egy harmadik, ‘z’ értékkel is, amelyik a tengerszint-, vagy az ellipszoid feletti magasságot adja meg. Mindhárom koordinátát rendszerint 64-bites lebegőpontos számokkal jelöljük.
   Az idáig bemutatott két koordináta-ábrázolást követi a legtöbb egyszerű adatcsere-formátum, például az ESRI Shape-file, WKT, vagy a GeoJSON a földrajzi-, az ellipszoidi- és a síkkoordináta-rendszerekben felvett adatok tárolására.
3. **Térbeli-, derékszögű koordináta-rendszerek**. A térbeli-, derékszögű koordináta-rendszerek Descartes–féle, derékszögű- háromtengelyű térkoordináta-rendszerek, melyek origója a Föld belsejében rögzített és a három, egymásra merőleges tengely mentén mért érték alapján adjuk meg egy pont pozícióját. A komponenseket ‘x’-el, ‘y’-nal és ‘z’-vel jelöljük, ahol a ‘z’ értéke független a tengerszint-, vagy ellipszoid feletti magasságtól, egyszerűen a ‘z’ tengely mentén mért távolságot jelöli. Mindhárom koordinátát rendszerint 64-bites lebegőpontos számokkal jelöljük. A térbeli-, derékszögű koordináta-rendszereknél a ‘z’ érték megléte nem opcionális: ha elhagyjuk, a koordináta használhatatlanná válik. Térinformatikai környezetben ezen koordináta-rendszerek közvetlen használata

ritka, a geometria feldolgozásáért felelős algoritmusok a felhasználó elől rejtve, a háttérben használják.

3.2 Koordináta-ábrázolás II.

A koordináta-ábrázolás fentebb leírt lebegőpontos megoldása roppant egyszerű és hatékony is egyben: a koordinátákat egyszerűen lebegőpontos számokkal lejegyezzük.

Ez tökéletesen működik mindaddig, amíg egyszerű pontokat, vagy kisszámú vonalat, vagy sokszögeket tárolunk. Abban az esetben, ha ezen elemek számossága nagy, vagy vizsgálnunk kell az egyes elemek egymáshoz viszonyított helyzetét (például, hogy egy pont ráesik-e egy adott vonal vég-, vagy kezdőpontjára), akkor ennél hatékonyabb módszerhez kell folyamodnunk.

A térinformatikában a koordináta-rendszerek érvényességi területének kiterjedése véges – mind a sík-, mind az ellipszoidi-, mind a földrajzi koordináta-rendszerek esetében.

A koordináta-rendszerek többsége egy jól körülhatárolható területegységet, például egy országot vagy egy országrészt fed le és az ezen belül található tereptárgyak pozíciójának számszerű megadására használják.

Így például az Egységes Országos Vetület (EOV) Magyarország területén és annak közvetlen környezetében érvényes. De a koordináta-rendszerek akkor is véges kiterjedésűek, ha nem kötődnek egy területegységhez, hanem az egész Földön használhatók: például az ellipszoidi koordináták még itt is csak a [−90°; +90°] ∈ ℝ-es, illetve a [−180°; +180°] ∈ ℝ-es intervallumon mozognak, miközben az elvárt pontosságnak sem kell 0,1 milliméternél jobbnak lenni.

Ebből a két korlátozásból kifolyólag nem feltétlenül szükséges lebegőpontos számokat használnunk: a koordináták egészszámokkal is lejegyezhetők. Fogjuk meg a vizsgálati területünket és illesszünk rá egy rácshálót, úgy, hogy a sorok és az oszlopok száma azonos legyen, miközben a rács lépésköze mindkét irányban legyen azonos és kicsi, például egytized milliméter. Ha a rácsháló egyik sarkának koordinátáját feljegyezzük, akkor minden koordinátát rögzíthetünk sor- és oszlopszámként – egészszám formájában:

$$x = \frac{c}{s} + x_0,$$
$$y = \frac{r}{s} + y_0,$$

ahol ‘x_0’ és ‘y_0’ a rácsháló origójának (bal alsó sarkának) lebegőpontos koordinátája; ‘s’ a lépésköz reciproka (lebegőpontos formában); ‘c’ az oszlop száma (egészszám); ‘r’ a sor száma (egészszám). Ebből az ‘x_0’; az ‘y_0’ és s az adott térinformatikai állományra jellemző állandó.

Vegyünk egy koordináta-rendszert, ahol $x_0 = 320\,000\ m$ és $y_0 = 32\,000\ m$, a lépésköz pedig 0,0001 m, tehát $s = \frac{1}{0{,}0001} = 10\,000\ \frac{1}{m}$.

Próbáljuk meg az $x = 522\,600\ m$ és az $y = 86\,400\ m$ jellemzett pont sor- és oszlopszámát meghatározni!

Az előző képletpár sor- és oszlopszámra átrendezve:

$$c = s \cdot (x - x_0)$$
$$r = s \cdot (y - y_0)$$

Behelyettesítve:

$$c = 10\,000\ \frac{1}{\cancel{m}} \cdot (522\,600\ \cancel{m} - 320\,000\ \cancel{m})$$
$$\underline{c = \ 2\,026\,000\,000}$$
$$r = 10\,000\ \frac{1}{\cancel{m}} \cdot (86\,400\ \cancel{m} - 32\,000\ \cancel{m})$$
$$\underline{r = \ 544\,000\,000}$$

Ezzel a módszerrel bármikor áttérhetünk lebegőpontos számokról egészszámokra, mellyel a szomszédsági műveletek hatékonysága növelhető miközben a kerekítési hibák nagyrésze is kiküszöbölhető.

A sokszög- és vonalábrázolás esetén a kezdőpontot követő pontok esetében elég a sorszám és oszlopszám különbségét eltárolni, például, ha a következő töréspont a kezdőponttól kereken egy méterre van *x* irányban, akkor a második pont oszlopszám-különbségeként elégséges feljegyezni, hogy 10 000. A geoadatbázisok többsége – a lebegőpontos tárolás helyett – ezt a sémát használja a koordináták tárolására.

3.3 Vektoros adatábrázolás I.

Ha ismerjük a koordináták informatikai ábrázolásának módszereit, akkor ebből a tudásból már komplex rendszereket építhetünk. Ehhez azonban először meg kell vizsgálnunk, hogy hogyan lehet a valóságot felbontani egyszerű elemekre, melyek koordináta-ábrázolással leírhatók:

- Első és legegyszerűbb elemünk a *pont*. A pont egyetlen egy X és Y, vagy λ és φ párral jellemzett érték, praktikusan két lebegőpontos szám. A ponttal entitások (valóságban létező dolgok) térbeli helyzetét jegyezhetjük le, melyek pontszerűek, vagy az adott vizsgálat szempontjából pontszerűnek tekinthetők. Ilyenek például egy közműtérképen a villanyoszlopok.

- Második elemünk a *vonallánc,* vagy *linestring*. A vonallánc egymás után összefűzött, irányított, összetartozó pontok sokasága. Vonallánccal ábrázolunk vonalszerű entitásokat, mint például az autópályákat. A vonalláncoknak két kötelező kitüntetett pontja van: ezek a kezdő- és a végpontok, melyek ábrázolásukat tekintve teljesen megegyeznek a közönséges pontok ábrázolásával. Ha a vonalláncunknak csak két pontja (egy kezdő- és egy végpontja) van, akkor azt szakasznak nevezzük. A vonalláncoknak azonban lehetnek köztes pontjai is: gondoljunk például egy folyóra! A folyónak van egy kezdő- (forrás-) és egy záró (torkolati-) pontja, közben számos alkalommal irányt vált: ezek a pontok a köztes pontok. A vonallánc pontjait (beleértve a kezdő- és végpontokat is) összefoglalóan *vertex*-nek nevezzük, így a vonallánc vertexek sorozata.
- A harmadik elemünk a *poligon*, azaz *sokszög*. A poligon zárt területegységek ábrázolására, például egy tó partvonalának leírására használjuk. A poligon a vonallánc elvi kiterjesztésének tekinthető: a poligon technikailag egy olyan speciális vonallánc, melynek kezdő- és végpontja egybe esik, és a körülkerített terület az adott objektumhoz tartozik, ezzel egy felületet ír le. A poligonok legalább négy vertexből állnak: a legkisebb szögszámú sokszögünk a háromszög, de mivel a kezdő- és végpontnak egybe kell esni, ezért az egyik pont duplázott.

Ezeket az egyszerű elemeket a legtöbb térinformatikai szoftverfejlesztő-környezet dimenzió-számokkal jelöli, így a pontra nulldimenziós-, a vonalra egydimenziós- a poligonra kétdimenziós objektumként hivatkoznak.

3.4 Vektoros adatábrázolás II.: a „Z" és az „M" kiterjesztés

Az idáig leírt pontok, vonalláncok és poligonok egyszerű, síkban fekvő objektumok voltak, csak síkkoordinátákkal jellemeztük őket. A valóság azonban ennél bonyolultabb: a domborzatnak, a tengerszint feletti magasságnak gyakran jelentősége van, ezért a X és Y, vagy λ és φ párral jelölt pontokhoz, vagy vertexekhez hozzácsatolhatunk egy harmadik elemet, a Z, vagy h értéket, mely a magasságot írja le: ezzel egy pontot három lebegőpontos számmal jellemezhetünk. Az így kiegészített vektoros elemek nevéhez rendszerint hozzátoldanak egy „Z"-t, így angolosan a PointZ, a LinestringZ, és a PolygonZ neveket kapjuk.

A „M" kiterjesztés nem ennyire kézenfekvő, de hasznos megoldás. Képzeljünk el egy folyót! A folyó a forrásától a torkolatáig folyik. A folyók mentén a helymeghatározás folyamkilométer szerint történik, amely a torkolattól mért *virtuális* távolság. A folyamkilométer (fkm) használata nagyon praktikus megoldás, ha meg akarjuk mondani, hogy hol tartózkodunk éppen a hajónkkal, vagy csónakunkkal a folyón, mivel elég egyetlen egy számot közölni: a folyamkilométert. Nincs szükségünk a szélesség- és hosszúság adatokra, hisz' eleve ismert, hogy a folyón vagyunk, így két

szám helyett egy számmal is megadható a pontos pozíciónk. Így például Mohács 1 446,9 fkm-nél van. Ha a folyón hajózunk elég ezt a számot megadnunk és nem kell a pontos koordinátát meghatározni (46,0025528°; 18,685776°).

Az előbbi definícióban azonban az szerepel, hogy „…torkolattól mért *virtuális* távolság". De mitől virtuális? A virtualitás abból fakad, hogy a folyók változnak, miközben a mérés pontossága is korlátozott. Amikor a folyamkilométer értékeket felmérik, akkor ezeket kövek formájába állandósítják a folyóparton. A folyók azonban átalakulnak: a kanyarulatok fejlődnek, kiegyenesednek, lefűződnek, vagy az adott szakasz egyszerűen oldalazik. Ezért a torkolattól mért geometriai távolság folyton változik – de az egyszerűség kedvéért ettől eltekintenek és a korábban meghatározott kövek alapján zajlik a navigáció. Ezért a távolság csak *virtuális*.

De ha virtuális, akkor fel kell jegyeznünk! Vegyünk egy folyót, melynek ismertek a vertexei X, Y, és Z formában. Ha fel szeretnénk jegyezni a folyamkilométer adatokat is, akkor ezt a koordinátahármast ki kell egészítenünk egy másik számmal, amely megadja a virtuális távolságot. Ezt a koordináta-értéket „*M*"-koordinátának nevezzük és rendszerint lebegőpontos számmal jegyezzük le. Az M-koordinátával kiegészített vektoros elemek neve angolosan: PointM, LinestringM, PolygonM. („Z"-koordináta megléte esetén PointZM, LinestringZM, PolygonZM).

Az „M"-koordinátát széleskörben használják a folyók- és a közutak mentén, de használható például az elektromos hálózatban a vezeték-ellenállás megadására is, tehát egy univerzális eszköz távolság- vagy virtuális távolság megadására.

3.4.1 „Többes" vektoros elemek

A felsorolt vektoros elemek egyedi elemeket jelöltek, például a pont egy darab pontot jelölt. Néha azonban szükséges, hogy egy jelenséget továbbra is pontszerűen, de nem egyedi pontként, hanem pontok csoportjaként jegyezzünk le. Ez a típus a *multipont* (multipoint). Multipontnak tekinthető, ha például egy repülőgép szétszóródott roncsait szeretnénk térinformatikai környezetben rögzíteni: egy gép, egy esemény, egy folyamat, tehát a pontok logikailag összetartoznak, ezért együtt is kell őket tárolni.

Ennek analógiájaként lehetséges, hogy a vonalláncok több, egymással nem csatlakozó részből álljanak. Például, ha előre rögzítjük, hogy a közúti közlekedésben a sávokat digitalizáljuk vonallánckén t, majd megpróbálunk rögzíteni egy autópályát, akkor a sávok geometriailag különállóak, de logikailag összetartoznak: egy autópályához tartoznak. Ez a tárolási megoldás a *polivonal* (polyline).

A poligonoknak is létezik többes kiterjesztése: a poligonok állhatnak több, egymással nem érintkező *gyűrűből*, de akár lehetnek lyukasak is, mely lyukakban szigetek is előfordulhatnak.

Egyes térinformatikai rendszerek az így kiterjesztett poligonokat multipoligonnak nevezik, míg mások, kicsit megtévesztően a multipoligont is szimplán poligonnak hívjak.

4 Raszteres adattárolás

A térinformatikában nem csak vektoros eszközök segítségével jegyezhetünk le jelenségeket és tulajdonságokat, hanem raszteres adatokkal is dolgozhatunk.

A raszteres adattárolás elve, hogy kijelölünk egy vizsgálati területet és ezt a területet rés- és átfedés-mentesen beborítjuk egyforma sokszögekkel. Ez a sokszög lehet a háromszög, négyzet, téglalap, hatszög, vagy ennél komplexebb forma is – bár a térinformatikaigyakorlatban csak a négyzet és a téglalap fordul elő. A négyzet és a téglalap használata azért is előnyös, mert így a vizsgálati területet sorokra és oszlopokra bonthatjuk: egy sor és egy oszlop metszéspontjában egy négyzet, vagy téglalap (összefoglalóan: *cella*) helyezkedik el: ezt az elemet *képelemnek*, angolul „picture element”-nek, röviden *pixel*-nek nevezzük. A cella/pixel közepén ülő érték az egész cellát reprezentálja.

De miért használnánk egy új eszközt, a rasztert, ha az adat más, például vektoros módon is eltárolható?

Készítsünk egy térképet, amely az Egyesült Államokban Wyoming állam domborzati viszonyait írja le! Wyoming-gel könnyű dolgunk lesz: a legtöbb szögtartó síkkoordináta-rendszerben az állam egy nagy, kb. 550 km × 440 km-es téglalap (K ↔ Ny × É ↕ D). A domborzatmodellünk legyen 10 × 10 méteres felbontású, a pontosság megőrzése érdekében a függőleges vonatkoztatási rendszer szerinti magasságot adjuk meg méterben, 32-bites lebegőpontos számok formájában!

Ehhez első lépésként fel kell osztanunk a vizsgálti területünket rés- és átfedés mentesen, azonos méretű sokszögekkel. 10 × 10 méteres felbontás esetén 55 000 db oszlopba és 44 000 sorba rendezett, összesen 2 420 000 000 db, 10 × 10 méteres négyzetet jelent.

A felosztás után minden egyes cella középpontjába beírjuk az adott, 10 × 10 méteres cellára vonatkozó átlagmagasságot egy lebegőpontos szám formájában.

Mivel a cellaméret, a sor- és az oszlopszám adott, ahhoz, hogy a raszterünket a földrajzi térhez kössük, nem kell mást tennünk, mint egyetlenegy pontnak megadni a koordinátáit lebegőpontos számok formájában és a raszter geometriájából fakadóan máris a földrajzi környezet megfelelő részére kerül. A térinformatikába ez a *regisztrációs pon*t rendszerint a raszter bal alsó cellájának bal alsó sarokpontja, vagy a bal alsó cella középpontja.

Egy kis utána számolással kiderül, hogy erre a tárolási problémára a raszteres adattárolás hatékonyabb alternatívát nyújt, mintha vektoros eszközökkel, például multiponttal írnánk le a vizsgálati terület domborzatát. Nézzük először a raszteres megközelítés adatigényét, 32-bites lebegőpontos számmal tárolva:

$$\frac{440\,000\,m}{10\,m} \cdot \frac{550\,000\,m}{10\,m} = 2\,420\,000\,000$$

$$2\,420\,000\,000 \cdot \frac{32}{8}\ byte = \ 9\,680\,000\,000\ byte$$

$$9\,680\,000\,000\ byte\ + 2 \cdot \frac{32}{8}\ byte\ (sarokpont)\ \approx 9{,}015\ GiB$$

Ugyanez az adatmennyiség multipont formában eltárolva, szintén 32-biten (3 db 32-bites koordináta pontonként [‘X’, ‘Y’ & ‘Z’]):

$$2\,420\,000\,000 \cdot 3 \cdot \frac{32}{8}\ byte \approx 27{,}05\ GiB$$

Látható, hogy a különbség jelentős, ezért mindig a modellezendő problémának megfelelő adattárolási eljárást kell választanunk.

4.1 Sorfolytonos adattárolás

Bizonyos jelenségek (terepmodellek, légi- és műholdképek, távérzékelési adatok) elsősorban raszteres megoldásokkal tárolhatók el hatékonyan. A raszteres adatsorokat azonban informatikai eszközökkel kell eltárolnunk. Vegyük példának az előző fejezetben leírt terepmodellt: ez a 550 km × 440 km-es területet lefedő, 10 × 10 méteres felbontású 32-bites, lebegőpontos terepmodell 55 000 db sort és 44 000 db oszlopot tartalmaz. Az informatikai adattárolás során azonban nem tudunk sorokat és oszlopokat írni: minden adatunkat (legyen az raszter, vektor, vagy bármilyen szám, vagy adatábrázolás) csak sorfolytonosan tudjuk ábrázolni. A sorfolytonos adatábrázolás lényege, hogy minden adatot kiterítve, egymás után írva ábrázol. A sorfolytonos adatábrázolás előnye, hogy az adatsorban való navigáció esetén egyetlen egy értékkel leírhatjuk a pozícióinkat: ez a *kezdőponttól mért távolság*.

Kísérletképpen vegyünk egy 3 × 3-as rasztert, melyben minden cella egy bájtnyi adatot tárol, összesen 9 bájtot:

5	7	9
11	0	4
6	1	3

Ilyenkor sorokba- és oszlopokba szervezett tömbként gondolunk erre a kis raszterre, de az adatsor így íródik a memóriába:

5	7	9	11	0	4	6	1	3

Itt a bájtok sortörés nélkül, sorfolytonosan tároltak egymás után, a rasztert egyszerűen sorba fejtettük. A sorba fejtés előnye, hogy az adatban egy számegyeneshez hasonlóan egy érték segítségével navigálhatunk:

Érték:	5	7	9	11	0	4	6	1	3
Pozíció/Táv (index):	0.	1.	2.	3.	4.	5.	6.	7.	8.

Tehát, hogyha például az utolsó sor adatait szeretnénk felhasználni (6; 1; 3) akkor a 6. index pozícióra kell ugranunk, és onnan három bájtot felolvasnunk. Így egy pozíció leíró számmal (index), egy dimenziós értékként navigálhatunk egy kétdimenziós raszterben. Az indexelés mindig nullánál kezdődik.

A gondolat azonban egy metaadat megléte nélkül sántít: ismernünk kell, hogy hány bájt hosszúságú egy sor. Jelen esetben három. Ez az érték a *lépéshossz* (stride). Így, ha kapunk egy fájlt, amelynek ismert a mérete és megkapjuk hozzá a sorhosszt, akkor azonnal kiszámítható, hogy a fájl hány soros és oszlopos rasztert tartalmaz.

Ha a fájl mérete 256 kiB (262 144 bájt), a sorhossz pedig 512 bájt, akkor a fájl méretét a sorhosszal visszaosztva rögtön megkapjuk, hogy a fájl egy 512 sorszor 512 oszlopos rasztert tartalmaz.

Természetesen, ha az adat nem ábrázolható egy bájton, akkor a számábrázolás típusát is rögzíteni kell. Például egy 32-bites (4 bájtos) lebegőpontos számokból álló, 20 oszlopos, 60 soros terepmodellt így írnánk le: „Típus: 32-bit; sorhossz: 80 bájt; fájlméret: 4 800 bájt”.

Jelenleg az összes térinformatikai raszter ezen sorfolytonos, ismert sorhosszon alapuló adattárolási módot követi.

4.2 Georeferencia

A fentebb leírt raszteres adattárolás univerzálisan, a matematikától a digitális fényképezésig széles körben megjelenik. A térinformatikai raszterek azonban egy kicsit más jellegűek: az ábrázolt jelenség, például egy terepmodellen látható hegység a földrajzi térben kötött; nem akárhol van a virtuális térben, hanem egy pontos pozícióhoz kapcsolódik. Ezért a térinformatikai raszterek kiegészülnek még néhány metaadattal, melyet a fájlméret és a sorhossz mellett el kell tárolni.

Ezek:

- A regisztrációs pont: raszter bal alsó, vagy bal felső pixelének középponti, vagy kilső sarokponti koordinátája – rendszerint két lebegőpontos számmal leírva. Ez részben rögzíti a rasztert a földrajzi térben.
- A raszter vetületi rendszere: a vetületi rendszer ismeretében tudjuk az előbbi koordinátánkat valós, földrajzi pozícióba mozgatni. Lényegében az írja le, hogy az előbbi koordináták mihez képest értelmezettek. A vetületi rendszer metaadata a vetületi rendszer hosszegysége is, például láb, fok, vagy méter. (Technikai értelemben a vetületi rendszer elhagyható: ekkor a rasztert ábrázoló szoftver az alapértelmezett koordináta-rendszere és hosszegysége alapján rajzolja ki a raszterünket.)
- A raszter celláinak mérete az 'X' irányban, a vetületi rendszer hosszegységében számolva.
- A raszter celláinak mérete az 'Y' irányban, a vetületi rendszer hosszegységében számolva. Ezen utóbbi két értékkel megadható, hogy az egyes cellák mekkora földrajzi teret fednek le. Például egy 10×10 méteres terepmodell cellái száz négyzetmétert fednek le, úgy, hogy a cellák négyzetesek.
- Ha a raszter celláinak éle nem igazodik a koordináta-rendszer tengelyeihez, akkor szükséges eltárolnunk az elforgatás mértékét is. Gyakori műholdképek esetén.

Ezen adatok összességét georeferenciának – földrajzi megfeleltetésnek – nevezzük. A georeferencia adatok rendszerint vagy az adott fájlban tároltak (pl.: GeoTIFF), vagy a fájl mellé csatolt ASCII szövegfájlban (World file, XML) foglalnak helyet.

4.3 Csempézés

A raszteres adatsorok – légifelvételek, terepmodellek, műholdas felvételek – igen gyorsan, igen nagyra tudnak nőni: egy 550 km × 440 km-es terület, 10 × 10 méteres felbontású, 32-bites terepmodellje kb. 9 GiB.

Ez egy hatalmas adatmennyiség, különösen akkor, ha a felhasználó gyorsan szeretne benne navigálni: egyes részeket megnézni, majd a térképet arrébb húzva ki- és belenagyítani.

A térinformatikai raszterek tárolása általában sorfolytonos, így az adat látszólag egy dimenzióba fejtett, ahol a pozíciónk leírható a kezdőponttól mért távolsággal. Ehhez képest a felhasználó monitorja egy kétdimenziós eszköz, valódi sorokkal és oszlopokkal.

Gondoljunk az előző ~9 GiB-os raszterünkre! Itt 55 000 oszlop és 44 000 sor van. A felhasználó szeretné megnézni a raszter észak-nyugati (bal, felső) sarkát, abból is egy 100 × 100 pixeles részt.

A feladat igen könnyűnek tűnik, de a sorfolytonos adattárolás miatt nehézkes. Rajzoljuk ki az első sort! Ehhez a fájlban a 0. pozícióra kell ugrani, majd beolvasni az első 400 bájtot (32-bit, tehát 4 bájt pixelenként, százszor). Rajzoljuk ki a második sort! A fájlban a 220 000. pozícióra kell ugranunk (55 000 sor × 4 bájt), majd innen előre olvasni 400 bájtot. A harmadik sorhoz el kell ugrani a 440 000. pozícióba kiolvasni a bájtokat, és így tovább a 100. sor végéig.

Az eljárás ugyan jól algoritmizálható és programozható, a hatalmas ugrások miatt az adat betöltése nehézkes.

A nagy ugrások elkerülésére érdekében a raszter csempékre bontható: jelöljünk ki pl.: 128 × 128-as darabokat a nagy raszterünkből és ezeket a kis csempéket írjuk le sorfolytonosan. Így az eredeti raszterünket kisebb raszterekre osztjuk fel, majd ezeket a kisebb rasztereket egymás után eltároljuk a memóriában. Ezzel a módszerrel a raszteres adatok egyes részeihez sokkal kisebb ugrások mentén is hozzáférhetünk. A csempézés során használt jellemző csempeméretek a térinformatikában: 128 × 128; 256 × 256; 512 × 512; 1024 × 1024 … $2^n \times 2^n$; $n \in \mathbb{Z}^+$.

4.4 Piramis

Az előző fejezetben megnéztük, hogy hogyan lehet egy nagyméretű raszter egy kis darabjához hozzáférni, majd a felhasználó monitorján megjeleníteni. Most próbáljuk meg az ellenkezőjét és jelenítsük meg az egész rasztert a monitoron! A raszter 55 000 × 44 000-es, a monitor csak 3 840 × 2 160 pixeles. Az egész raszter nyilván nem jeleníthető meg egy az egyben a monitoron, mert a monitor felbontása túlzottan alacsony. Ahhoz, hogy a teljes rasztert mégis meg tudjuk jeleníteni, az egész rasztert be kell olvasnunk, 14 × 20-as pixelblokkokat képeznünk (55 000 ÷ 3 840 ≈ 14; 44 000 ÷ 2 160 ≈ 20), ezen pixelblokkokban lévő értékeket például átlagolni kell (vagy kiválasztani a legelsőt, a legkisebb, vagy a legnagyobb számot), majd az így kapott értéket megjeleníteni a monitoron.

Az eljárás ugyan nem bonyolult, de hatalmas adatmozgatást és egyes eljárásoknál rengeteg számítást (pl.: átlagolás) igényel: a teljes terepmodell megjelenítéséhez ~9 GiB-et kell beolvasni. És mi történne, ha a felhasználó a terepmodell egyik oldalát szeretné megnézni? Ismét 4,5 GiB-et be kell olvasni és a másik a 4,5 GiB-et pedig át kell ugrani. És ha a felhasználó ismét az egész rasztert szeretné látni? Ismét 9 GiB.

Ez a megközelítés bár helyes, gyakorlatban tarthatatlan: a felhasználónak elviselhetetlenül sokat kellene várni, hogy a beolvasás, majd ismételt beolvasás újra és újra megtörténjen. Még akkor is, ha az adat csempézett.

Az egyetlen megoldás az, ha piramisokat képzünk: a piramis az adott raszter mellé leírt, kisebb felbontású raszterek sorozata, amelyek az eredeti, nagy felbontású raszterből származnak. Ha

van például egy 2 048 × 2 048-as raszterünk, akkor abból készítünk – négy pixel átlagolásával – egy 1 024 × 1 024-es, egy felezett, 512 × 512-es, majd ismét egy felezett 256× 256-os változatot is. Ekkor, ha a felhasználó egy áttekintő képet szeretne kapni, nem az eredeti nagyméretű rasztert, hanem csak az aktuális monitor-felbontásnak megfelelő kisebb rasztert olvassuk be és jelenítjük meg.

Az eljárás gyors, de érezhetően tökéletlen, mert tárhelyet foglal: a csökkentett felbontású rasterek együttesen kb. a nyolc százalékát teszik ki az eredeti raszter méretének, így a ~9 GiB-es raszterünk helyett egy ~9,72 GiB-es adatsort kell eltárolnunk. A piramisképzés során a többlet tárhelyfoglalást időmegtakarításra cseréljük.

A piramisok, mivel a bennük lévő pixelek nem vesznek részt közvetlen elemzésben, gyakran veszteségesen tömörítettek.

4.5 Veszteségmentes rasztertömörítési eljárások

A reszteres adattárolás mind a memóriában, mind a háttértárakon igen nagy lehet. Ha fotogrammetriai eljárásokkal egy 1 km × 1 km-es területről 0,1 méteres terepmodellt készítünk, akkor az 10 000 × 10 000 pixelt jelent (1 km = 1 000 m; 1 000 m / 0,1 m = 10 000). Ha ezt 32-bites lebegőpontos számokkal tesszük, akkor az pixelenként 4 bájt, összesen 400 000 000 bájt, amely megközelítőleg 381,47 MiB, mely csak a raszter nyers mérete, melyhez a piramis, a georeferencia és a rasztert kísérő metaadatok is társulnak, de most az egyszerűség kedvéért foglalkozzunk csak és kizárólag a raszter nyers adattartalmával! Első lépésben csempézzük fel a raszterünket, szokatlanul apró, 4×4-es blokkokba (3. ábra)!

123,4	123,4	123,4	123,4
123,4	123,7	123,7	123,7
123,8	123,8	123,8	123,8
124	124	124	124

3. ábra. Egy nagyméretű raszter 4×4-es, bal felső darabja. A számértékek tengerszint feletti magasságot jelölnek méteren.

Ez a táblázat a nagyméretű raszterünk egy darabkáját ábrázolja, a bal, felső sarok 4×4 pixeles részét. A benne lévő számértékek a tengerszint feletti magasságot ábrázolják méterben. Ha térben gondolkodunk, akkor ez a raszter észak felé lejt.

Vizsgáljuk meg az adatokat! A terep jellegéből fakadóan a számértékek nem véletlenszerűek, hanem térben csoportosulnak. A rasztert sorfolytonosan olvasva láthatjuk, hogy bizonyos értékek ismétlődnek: van 5 db 123,4-es, 3 db 123,7-es, 4 db 123,8-es és 4 db 124 méteres értékünk.

Ezek az adatok ismétlődnek, az ismétlődés mintát követ, ezért az adatsor tömöríthető. Ezzel, hogy megszámoltuk a különböző számértékeket már el is végeztük a legegyszerűbb tömörítési eljárást.

Jegyezzük le az előbbi rasztert úgy, hogy felírjuk, hogy milyen értékből hányat kell behelyettesíteni: **5**, 123,4; **3**, 123,7; **4**, 123,8; **4**, 124. Számoljuk össze, hogy ez mennyi adatot igényel! Tudjuk, hogy a raszter 4×4 pixeles csempékben van és ha minden adat egyforma lenne, akkor a legnagyobb számosság is csak 16. Ezért az első, számosságot leíró értéket válasszuk egy bájtnak, melybe az $[1; 16] \in \mathbb{Z}^+$ bőven belefér. Ekkor a négy különböző érték miatt négy számosságot leíró értéket kell lejegyeznünk, ami összesen négy bájt. A magasságot leíró számok, az eredeti pixelértékek lebegőpontosak, így darabonként négy bájtot használunk, ami összesen 16 bájt, így együttesen 20 bájt (4 + 16). Az eredeti raszter tárhelyigénye 16 × 4 bájt, ami összesen 64. Tehát ha így jegyezük le a rasztert, akkor 44 bájtot spórolunk (~69%). Ez az eljárás a futáshossz-kódolás, ismertebb nevén a Run-length Encoding (RLE).

Az így tömörített raszterek kitömörítése nagyon egyszerű: beolvassuk az első öt bájtot, majd az utolsó négyet annyiszor megismételjük, amilyen számot találunk az elsőben. Így az adat veszteségmentesen, az eredetivel teljesen azonosan helyreállítható: az eljárás *veszteségmentes*.

Ezen tömörítési eljárásnak vannak nyilvánvaló hibái: csak akkor érdemes alkalmazni, hogyha az adatsor számottevő mennyiségű ismétlődést tartalmaz. Ha az ismétlődések száma nem elégséges, akkor a „tömörítés” után az adatsor mérete még növekedhet is, így minden tömörítési eljárás esetében elmondható, hogy azt az adatsor ismeretében kell megválasztani.

Az RLE kódolás továbbfejlesztett változata az LZ77 (Lempel–Ziv 1977), azzal a kiegészítéssel, hogy nem csak egy számérték, hanem egy teljes számsor ismétlődését képes eltárolni, úgy több ismétlés esetén is csak egyszer jegyzi fel az azonos kombinációkat. Az LZ77 és továbbfejlesztett változata az LZ78 (Lempel–Ziv 1978) az RLE-hez hasonlóan veszteségmentes, egyszerű tömörítési eljárás, mely ismétlődéseket tartalmazó adatsorok eltárolására alkalmas. Jelen mintaadat esetében az LZ77 eljárással nem lehet az RLE-hez képest további méretcsökkenést elérni.

Az előbbi megoldásokkal rokon eljárás az LZW (Lempel–Ziv–Welch), mely egy a fájl elején elhelyezkedő szótárban jegyzi fel a lehetséges értékeket (melyek mintázatokból állhatnak), majd az érték a szótárban elfoglalt pozíciójára hivatkozik.

4.6 Veszteséges rasztertömörítési eljárások

Az ideáig bemutatott megoldások veszteségmentesek voltak: az adat a tömörítés előtt és után is pontosan ugyanazon tulajdonságokat mutatta: a nyers, valamint a be-, majd kitömörített adat között semmilyen különbség sincs.

Bizonyos esetekben erről a „luxusról" lemondhatunk. Vegyünk egy tipikus, például LiDAR (Light Detection And Ranging – lézeres táv- és iránymérés) adatsorból előállított raszterizált domborzatmodellt (4. ábra):

123,4544589	123,4221479	123,45548974	123,435541
123,4778958	123,7554589	123,74487875	123,785542
123,8874599	123,811111	123,86879954	123,884484
124,4588466	124,455899	124,12589777	124,987823

4. ábra. LiDAR-ból nyert domborzatmodell darabkája. A cellák nem négyzetes mivolta csak ábrázolástechnikai megoldás és nem befolyásolja a bumutatott eljárást.

Itt a méterben kifejezett, tengerszint feletti magasságot leíró számok 64-bites, lebegőpontos számok, hét tizedesjegyre kerekítve. A raszterünk itt 4×4-es, összesen 16 pixel, pixelenként 64-bittel (4 bájttal), így a teljes adatmennyiség 4×4×8=128 bájt. Szükségünk van a valóságban ennyi adat eltárolására egyáltalán?

Vegyük a táblázatból az első számot: 123,4544589. Tíz számjegy, ebből hét tizedesjegy. Ez a gyakorlatban azt jelenti, hogy a számok élessége tízezred milliméteres (10^{-7} m), ami már önmagában is teljesen felesleges, de ha hozzávesszük, hogy a LiDAR és a LiDAR-hoz kapcsolt GNSS (Global Navigation Satellite System – globális, műholdas helymeghatározó rendszer) rendszer is kb. 1–5 centiméteres hibát produkál, akkor belátható, hogy a tizedesjegyek nagy részének tárolása gyakorlatilag felesleges: a rengeteg számjegy csak az utófeldolgozás lebegőpontos matematikájánál „melléktermék".

Próbáljuk meg ezt az adatsort veszteségesen tömöríteni, úgy, hogy a legnagyobb megengedhető eltérést előre definiáljuk. Ez az érték jelen esetben legyen 0,01 m!

1. Első lépésben határozzuk meg az adatsor minimumát, maximumát és terjedelmét (maximum mínusz minimum)! Jelen esetben ez 123,4221479 m; 124,987823 m és 1,565675100000007 m (vegyük észre, hogy a lebegőpontos számábrázolás korlátai miatt a terjedelem nem írható le hiba nélkül [lásd: Lebegőpontos számok])
2. Minden egyes számból vonjuk ki a minimumot, majd osszuk el a legnagyobb megengedett hiba kétszeresével, amely jelen esetben $0{,}01 \cdot 2 = 0{,}02$. Így a minimumot tartalmazó cellába nulla, a maximumot tartalmazó cellába pedig 78,28375500000035 kerül.

3. Az előző lépésben kapott lebegőpontos számokat kerekítsük a legközelebbi egész-számra. Így az eredetileg a minimum értéket tartalmazó cella értéke 0, a maximumot tartalmazó cella értéke pedig 78 lesz. Ez a gyakorlatban azt jelenti, hogy a számok bár 123 m feletti értékeket reprezentálnak, az egész raszter leírható 0,02-es lépésközzel 78 lépésben. 78 db lépés összesen 7 biten írható le (mert $\lceil \log_2(78) \rceil = 7$, ahol a '⌈' & '⌉' a felfelé kerekítés jele). Az ezen módszerrel átskálázott raszterünk így fog kinézni (5. ábra):

2	0	2	1
3	17	16	18
23	19	22	23
52	52	35	78

5. ábra. Átskálázott raszter. Forrás raszter: 4. ábra.

4. Az így kapott hétbites számokat egyszerűen lejegyezzük és megbecsüljük, hogy a tömörítés mennyire volt hatékony. Eredetileg a raszter 4×4-es, pixelenként 64-bites lebegőpontos számokkal, összesen 128 bájt, ami 1024 bit. Az itt bemutatott transzformációval a számokat hétbitesre konvertáltuk, így az eredményraszter mérete csak 4×4×7, azaz 112 bit, ami 14 bájt. Ez az eredeti méret ~11%-a!

Tegyünk kísérletet az adat visszaállítására! Ehhez szükségünk van az átskálázott pixelértékekre az előző táblázatból, valamint két metaadatra: a minimumra és a maximális hiba kétszeresére.

Állítsuk vissza a bal felső 2-es értékű pixelt: 0,02 × 2 + 123,4221479 = 123,4621479 m. Tehát a helyreállított érték 123,4621479, amely nem egyezik az eredetileg ott lévő 123,4544589 m értékkel, de ha megnézzük a kettő közötti abszolút hibát, akkor ennek értéke mindössze 0,007688999999999169 m, amely kisebb, mint az előre elhatározott maximálisan engedélyezett 0,01 m. Így ugyan tömörítési veszteség keletkezik, de cserébe az adatmennyiség kb. 90%-a elhagyható, miközben a raszter érdemi információtartalma alig csökkent. (Más raszterek és maximális hibák esetében ez az arány változik.)

Az itt leírt tömörítési eljárás neve LErC (Limited Error Compression), azaz korlátozható hibájú tömörítési eljárás (U.S.A Patent No. US 9,002,126 B2, 2015; Halmai, 2024).

Az itt bemutatott eljáráson kívül még nagyszámú veszteséges rasztertömörítési eljárás létezik, melyek közül leghíresebbek a JPEG (Joint Photographic Experts Group) és a JPEG2000 melyek főleg légi- és műholdfelvételek veszteséges tömörítésére alkalmasak.

5 A geoinformatika felsőgeodéziai alapjai

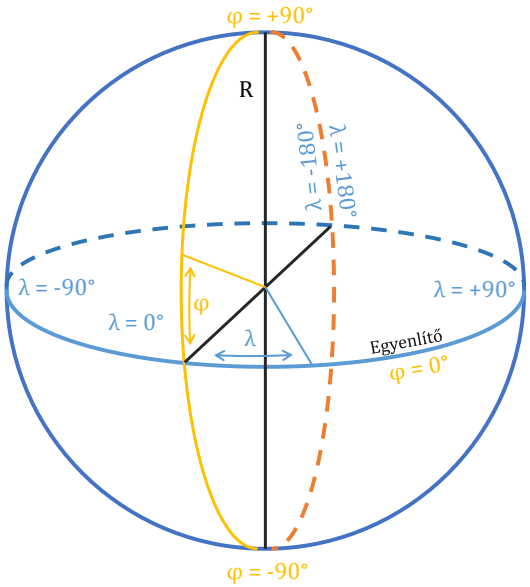
Térinformatikai vizsgálataink lehetnek lokálisak és globálisak egyaránt, de egyvalami mindkét megközelítésben rokon: a vizsgálatok a térben zajlanak és ahhoz, hogy a térben – Földön, más bolygón, vagy virtuális térben – *számszerűen* meg tudjuk határozni a pozíciókat, koordináta-rendszerekre van szükségünk.

A geoinformatikában többféle koordináta-rendszert is használunk, de ezekben egy elem közös: minden megoldás valahogy a Földhöz kötődik, ezért a Föld alakja, középpontjának helyzete, a Föld forgástengelyének helyzete közvetve-, vagy közvetlenül befolyásolja az adott koordináta-rendszer tulajdonságait.

A Föld alakja a *geoid*. A geoid egy olyan térbeli alakzat, amely a Föld aktuális formáját írja le: idő-ről-időre változó, komplex forma, melynek felmérése nehéz és az eredmények csak egy véges időintervallumra érvényesek. Ezért a komplex geoid formát a geodéziában közelítik matematikailag jól leírható geometriai formákkal (gömbökkel és forgási ellipszoidokkal) és raszteres modellekkel egyaránt.

5.1 Földrajzi koordináta-rendszer

Első, és legegyszerűbb eszköz a geoid formájának közelítésére a gömb. Ilyenkor a gömb középpontját úgy helyezik el és olyan sugarat rendelnek hozzá, hogy a legjobban közelítse a geoid egészének-, vagy egy részének (pl. egy ország területére eső részének) formáját. Ezen gömbön értelmezett koordináta-rendszereket gömbi polárkoordináta-rendszernek, gömbi koordináta-rendszernek, vagy ismertebb nevén *földrajzi koordináta-rendszernek* nevezzük (6. ábra).



6. ábra. A földrajzi koordináta-rendszer. A felénk eső folytonos sárga vonal az önkényesen kijelölt nullmeridiánt jelöli (λ = 0°). A nullmeridián síkjában fekszik a Föld forgástengelye (függőleges fekete vonal). A forgástengely fele a gömb sugara (R). A nullmeridián síkjára, így a forgástengelyre merőleges legnagyobb gömbi kör által kijelölt sík az Egyenlítő (φ = 0°). A földrajzi szélességet (φ) az Egyenlítő síkjától észak- és dél felé szögtávolságként mérjük. Az Északi-sarkot a φ = +90°, míg a Délit- a φ = -90° jelöli. A földrajzi hosszúságot a keletre (az ábrán jobbra) és nyugatra (az ábrán balra) mérjük és a nullmeridiántól mért szögtávolságként értelmezzük. A nullmeridiánnal átellenben lévő dátumválasztó vonalat narancssárga szaggatott vonal jelöli az ábra hátoldalán (λ = ±180°). Az Egyenlítő síkja a Földet északi- és déli félgömbre osztja, míg a nullmeridián/dátumválasztó vonal síkja keleti- és nyugati félgömbre bontja (Hazay, [1956] ötletei alapján).

A földrajzi koordináta-rendszer egy olyan poláris koordináta-rendszer, mely rendelkezik egy a Föld középpontjába eső origóval és két, az origón átmenő, egymásra merőleges referencia síkkal: az egyik referencia síkunk az önkényesen kijelölt nullmeridián síkja, mely síkban a Föld forgástengelye felkszik, míg a másik referencia sík az erre merőleges, és a középponton szintén átmenő, egyenlítői sík.

A földrajzi koordináta-rendszerben ezen két referencia-síktól mért szögtávolság alapján adhatjuk meg *számszerűen* egy adott földrajzi pont helyzetét.

Nézzük először az egyenlítői síkot! Ez a sík északi- és déli félgömbre osztja fel a Földet. Az egyenlítői síktól mért szögtávolság értékét földrajzi szélességnek nevezzük és a görög, kis fí ‘φ’ betűvel jelöljük.

A fí-nek két írásmódja ismeretes: ‘φ’ és ‘ϕ’. Mivel a térinformatikában mind földrajzi- (gömbi-), mind ellipszoidi koordináta-rendszereket használunk (5.2. fejezet) és mindkét esetben előfordul a „szélesség”, mint fogalom, de értelmezésük eltérő, ezért az első ‘φ’ alakot fenntartjuk a földrajzi- (gömbi-) koordináta-rendszerekben értelmezett szélességek jelölésére, a ‘ϕ’ alakot pedig az ellipszoidi koordináta-rendszerek szélességének jelölésére használjuk. Bár ez a megoldás a gyakoribb, más szakirodalmak ettől eltérhetnek.

A ‘φ’ földrajzi szélesség értéke −90°-tól +90°-ig terjedhet. Mértékegysége a fok (°). A ‘φ’ értéke pozitív, ha a vizsgált pontunk az északi féltekén helyezkedik el és negatív, hogyha a délin. A földrajzi szélességet úgy kapjuk meg, hogy a vizsgált pontunkat összekötjük a Föld középpontjával, majd lemérjük, hogy az így kapott szakasz mekkora szöget zár be az Egyenlítővel. A szélességi körök egymással párhuzamosak, így sohasem metszik egymást.

A másik referencia-síkunk az Egyenlítőre merőlegesen kijelölt sík, amely a null-, vagy más néven kezdőmeridiánra, és az ezzel átellenben lévő dátumválasztó vonalra illeszkedik. A nullmeridián a keleti- és nyugati félgömbre osztja a Földet.

A nullmeridiántól mért szögtávolságot földrajzi hosszúságnak nevezzük és a görög, kis lambdával jelöljük: ‘λ’. Értéke −180°-tól +180°-ig terjed (mindkét oldalról zárt intervallumon). Mértékegysége a fok (°).

A ‘λ’ értéke pozitív, ha a vizsgált pontunk a keleti féltekén helyezkedik el és negatív, hogyha a nyugatin.

A földrajzi hosszúságot úgy kapjuk meg, hogy a vizsgált pontunkat összekötjük a Föld középpontjával, majd lemérjük, hogy az így kapott szakasz mekkora szöget zár be a nullmeridiánra illeszkedő síkkal. A hosszúsági körök egymást két pontban, az Északi- és a Déli-sarkon metszik.

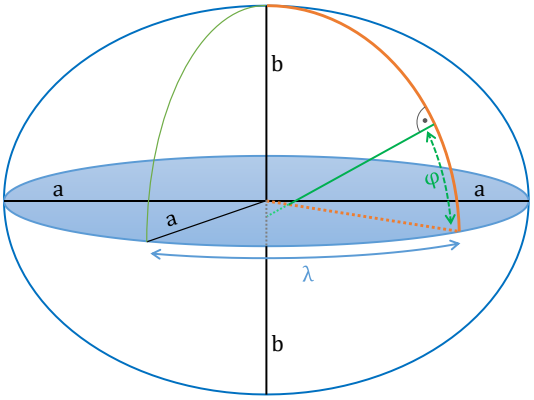
A földrajzi koordináta-rendszer segítségével hatékonyan navigálhatunk a Föld felszínén, mindössze két érték, a ‘λ’ és a ‘φ’ megadásával, melyet informatikai környezetben rendszerint 64-bites, lebegőpontos számokkal ábrázolunk.

A ‘λ’ és a ‘φ’ kiegészíthető egy harmadik ‘h’ értékkel is, mellyel a gömb felszíne feletti magasságot adhatjuk meg. Rendszerint 64-bites, lebegőpontos számmal ábrázoljuk.

5.2 Ellipszoidi koordináta-rendszer

Abban az esetben, ha a geoidhoz a gömbnél egy pontosabban illeszkedő matematikai formára lenne szükségünk, akkor forgási ellipszoidot kell választanunk.

A forgási ellipszoid felszínén ellipszoidi-, vagy más néven *geodéziai koordinátákat* vehetünk fel. A geodéziai koordinátákat a földrajzi koordinátákhoz hasonlóan ‘Λ’-val és a ‘ϕ’-vel jelöljük, ugyanúgy keleti- és nyugati, illetve északi- és déli félgömbre osztják a Földet, ugyanúgy a [−180°; +180°] ∈ ℝ, illetve a [−90°; +90°] ∈ ℝ tartományon értelmezettek, de az ellipszoid formájából fakadóan a szélesség definíciója *lényegesen* eltérő.



7. ábra. Az ellipszoidi koordináták értelmezése. Az ellipszoidi-/ geodéziai 'φ' szélesség a vizsgálati pontban az ellipszoid felszínére merőleges vektor (⏢) és az Egyenlítő síkja között bezárt szög. Az ellipszoid formájából fakadóan, ha a felületi normálvektort a Föld belseje felé irányítjuk, akkor a vektor átmegy a forgástengelyen, de nem megy át az ellipszoid középpontján (szaggatott zöld vonal). Kivételt csak az Egyenlítő és a sarkok mentén felvett pontok jelentenek, mert ekkor a normálvektor a gömbhöz hasonlóan a középponton halad át. Ebből falakadóan a szélesség nem értelmezhető középponti szögként, hanem mindig a felületi normálvektorhoz kötött. A 'Λ' geodéziai hosszúság definíciója megegyezik a földrajzi hosszúságnál megadottakkal. Az ábrán az 'a' jelöli a forgási-, kéttengelyű ellipszoid fél nagytengelyét, még a 'b' a fél kistengelyt (Hazay, [1956] ötletei alapján).

Az ellipszoidon mért, geodéziai-, '*φ*' szélesség értékét úgy határozzuk meg, hogy a felmérni kívánt pontban meghatározzuk az ellipszoid felszínére merőleges vektort, a *normálvektort*. Ezen normálvektor és az egyenlítői sík által bezárt szöget nevezzük geodéziai-, vagy ellipszoidi szélességnek. Ebből a definícióból kitűnik, hogy egy és ugyanazon pont ellipszoidi- és földrajzi szélessége eltérő lehet!

Az ellipszoidon mért 'Λ' geodéziai hosszúság definíciója a földrajzi hosszúságéval egyenértékű: az adott ponton átmenő meridián- és a nullmeridián síkja közt bezárt szög.

A földrajzi koordinátákhoz hasonlóan ez a koordináta pár kiegészíthető, egy az ellipszoid felszíne felett mért 'h' magassági értékkel.

5.3 A földrajzi- és az ellipszoidi koordináta-rendszer hibái

A két koordináta-rendszert régóta alkalmazzák, hogy a földrajzi környezet egyes elemeinek pozícióját számszerűsítsék. Két számérték, korlátozott intervallumokon, a valós számok halmazán, miközben az értékek jól megfeleltethetők az égtájakhoz viszonyított mozgással. Így, ha pontosan keletre indulunk, tudhatjuk, hogy a 'λ' földrajzi-, vagy a 'Λ' geodéziai hosszúság értéke növekedni fog és csak ez az érték változik. Hasonló mondható el a 'φ', vagy 'ϕ értékekről is, ha pontosan északra- vagy délre mozgunk.

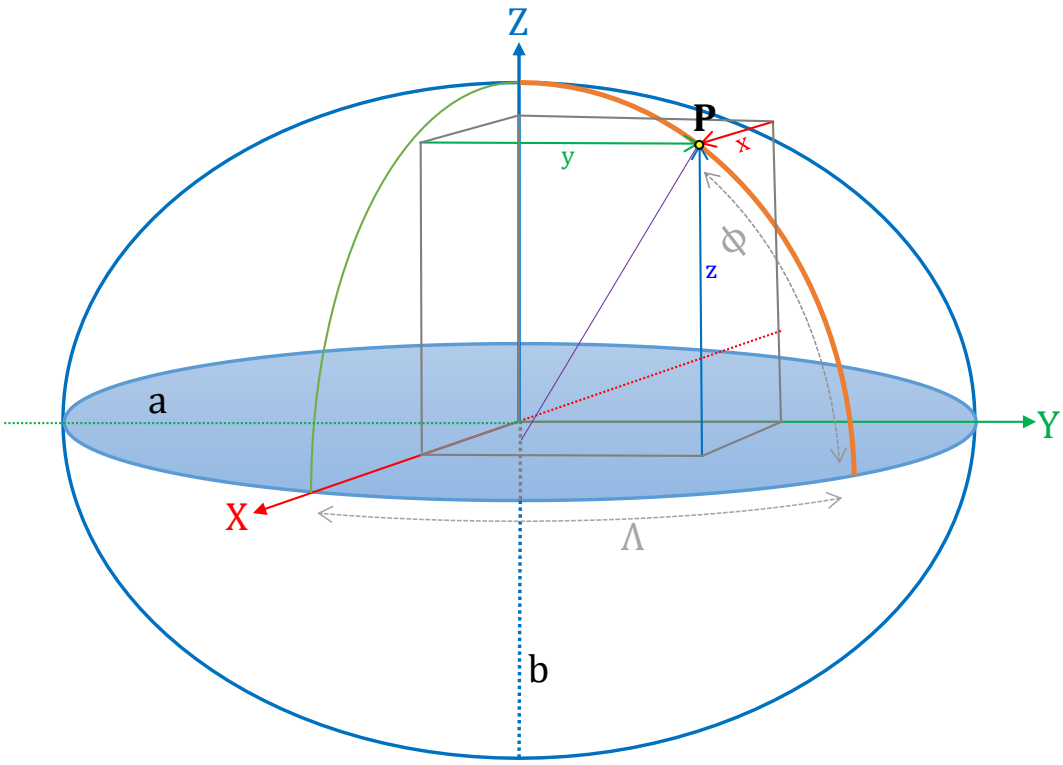
De ezen viselkedésből a rendszer egy gyengesége is adódik: normális esetben egy koordináta (pl.: 16,9682519°K 47,2036664°É) egy pontot azonosít a Föld felszínén. A két sarkpontban azonban a hosszúság fogalma értelmét veszti, mivel a meridiánok kétszer egy pontban metszik egymást. Itt végtelenszámú koordináta jelöl egy azonos pontot a földrajzi térben. Ez a probléma

látszólag elvi és elhanyagolhatónak is tűnik egyben, de rögtön navigációs számítási problémákat okoz, ha például repülővel haladunk el a sarkpontok felett: ekkor a haladási irány megállapításához nem elegendő az aktuális- és a jelenlegi pozíció ismerete, hanem nyilván kell tartanunk az előzőt megelőző pont koordinátáját is – vagy a haladási irányt. A földrajzi- és az ellipszoidi koordináta-rendszer a sarkokon *szingularitást* mutat.

A földrajzi- és az ellipszoidi koordináta-rendszerekben a számításokhoz nagyszámú szögfüggvényre és azok inverzeire van szükségünk. Ez tisztán matematikai értelemben semmilyen problémát nem jelent, de jelenleg a koordinátáinkat bináris, véges pontosságú, digitális rendszerekben tartjuk nyilván, amely már gyakorlati problémákat okoz. Egyes szélső helyzetekben például az arkuszszinusz a pontatlan eredményt adhat, ha szög értéke közel esik a ±90°-hoz.

5.4 Térbeli-, derékszögű koordináta-rendszerek

Egyes térinformatikai számítások sokkal könnyebbek, ha nem egy görbült-, referencia felszínen kell elvégeznünk őket, hanem egy térbeli-, derékszögű, Descartes–féle, háromtengelyű koordináta-rendszerben. A térbeli-, derékszögű koordináta-rendszereket egyszerűen úgy hozzuk létre, hogy a Föld belsejébe behelyezzük egy háromtengelyű-, derékszögű koordináta-rendszer origóját, úgy, hogy a tengelyeket valamilyen referencia irányokhoz, vagy -síkokhoz igazítjuk (8. ábra).



8. ábra. A térbeli-, derékszögű koordináta-rendszer és kapcsolata az ellipszoidi koordináta-rendszerrel. A nagy 'X', 'Y' és 'Z' betűk a térbeli- derékszögű koordináta-rendszer tengelyeit jelölik. A vizsgált 'P' pont az apró, sárga, fekete szegélyű kör. A vizsgált pont koordinátái leolvashatók a tengelyekkel párhuzamos segédegyenesek mentén, melyeket kis 'x', 'y' és 'z' betűk jelölnek. A fekete 'a' és 'b' betűk a forgási ellipszoid fél nagy- és fél kistengelyét mutatják. A vizsgált pont ellipszoidi koordinátái a szűrke 'Λ' és 'ϕ' mentén olvashatók le. A vizsgálati pontból az ellipszoid belseje felé bocsájtott lila segédvonal az ellipszoid felsínére merőleges, áthalad a 'Z' tengelyen, de nem halad át az ellipszoid középpontján. Az ábrán a koordináta-rendszer jobb kezes, jobbsodrású. Egyes (ritka) térbeli-, derékszögű koordináta-rendszerekben a koordináta-rendszer sodrása eltérő lehet, például az 'X' és az 'Y' tengelyek felcserélten is megjelenhetnek (Hazay, [1956] ötletei alapján).

A térbeli-, derékszögű koordináta-rendszerek tetszőlegesen elhelyezhetők ugyan, de van egy speciális családjuk: az *Earth Centered Earth Fixed – röviden ECEF – Föld középpontú, Földhöz rögzített koordináta-rendszerek*. Ezen koordináta-rendszerek origója a Föld geofizikai tömegközéppontjával egybe esik, az XY sík az Egyenlítővel síkjában fekszik, a 'Z' tengely pedig a Föld (elvi) forgástengelyéhez kötődik, az 'X' tengely egy egyezményes nullmeridián irányába mutat. Ezen koordináta-rendszerek a geoiddal együtt forognak a 'Z' tengelyük körül.

A földrajzi (gömbi-) koordináták átszámíthatók derékszögű koordináta-rendszerbe az alábbi képletekkel:

$$X = (R + h) \cdot \cos\varphi \cdot \cos\lambda$$
$$Y = (R + h) \cdot \cos\varphi \cdot \sin\lambda$$
$$Z = (R + h) \cdot \sin\varphi,$$

ahol 'R' a gömb sugara, 'h' a gömb felszíne felett mért magasság (mely, ha irreleváns, nullának tekinthető), 'φ' a *földrajzi* szélesség, 'λ' a *földrajzi* hosszúság.

Az ellipszoidi koordináták is átszámíthatók térbeli-, derékszögű koordináta-rendszerbe az alábbi képletekkel:

$$X = (N + h) \cdot \cos\phi \cdot \cos\Lambda$$
$$Y = (N + h) \cdot \cos\phi \cdot \sin\Lambda$$
$$Z = (N \cdot [1 - e^2] + h) \cdot \sin\phi,$$

ahol $e^2 = \frac{a^2 - b^2}{a^2}$, $N = \frac{a}{\sqrt[2]{1 - e^2 \cdot \sin^2\phi}}$, '$a$' az ellipszoid fél nagytengelyének-, 'b' az ellipszoid fél kistengelyének hossza, 'h' az ellipszoid felszíne felett mért magasság (mely, ha irreleváns nullának tekinthető), a ϕ és a Λ az ellipszoidi szélesség és -hosszúság.

Az ECEF koordináta-rendszert széles körben alkalmazzák olyan környezetekben, ahol nem kell a számértékeket emberi szemnek értelmezni, így a műholdas helymeghatározás, a távérzékelés és a domborzatelemzés belső számításai rendszerint ezen koordináta-rendszerben zajlanak. Az ECEF-ben használt matematikai apparátus jóval egyszerűbb a szögpárral operáló poláris koordináta-rendszerekhez képest (kevesebb a számítást lassító szögfüggvény). Így például a globális helymeghatározás során a műholdak pillanatnyi pozícióját a szoftver ECEF-ben határozza meg, majd az antenna helyét is ECEF-ben adja meg, de mivel az ECEF értelmezése nem intuitív és nehéz emberi szemlélő számára ezért az éréket a szoftver visszaalakítja ellipszoidi szélesség és -hosszúságra.

Az ECEF 'X', 'Y' és 'Z' értékeit rendszerint méterben mérjük és 3 db 32-, de inkább 64-bites lebegőpontos szám formájában tároljuk. A Z érték elhagyása esetén a koordináta teljesen használhatatlan, mivel a 'X', 'Y' pár „bezuhan" a Föld belsejébe.

ECEF koordináta-rendszer esetében, a földfelszín közelében elhelyezkedő pontokra mindig igaz, hogy az adott pontra kiszámított $d = \sqrt[2]{X^2 + Y^2 + Z^2}$ érték mindig nagyobb, vagy egyenlő, mint a forgási ellipszoid fél kistengelyének hossza, de kisebb, vagy egyenlő, mint a fél nagytengely hossza (a geoid-unduláció mértékét elhanyagolva).

Az ECEF koordináta-rendszerek nem egy konkrét megvalósítást, csak egy adatszervezési eljárást jelölnek: minden geodéziai dátumhoz (lásd később: A geodéziai dátum) rendelhető egy térbeli-derékszögű koordináta-rendszer. Egyes geodéziai dátumok középpontja a Föld geofizikai tömegközéppontjába esik: az ilyen dátumokból konstruált térbeli- derékszögű koordináta-rendszerek ECEF-rendszerek is egyben.

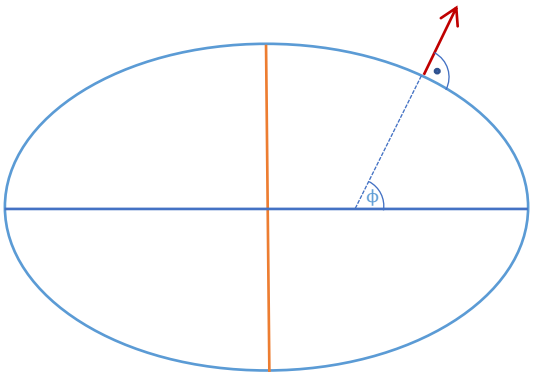
5.5 A térbeli-, derékszögű koordináta-rendszerek hibái

A térbeli-, derékszögű koordináta-rendszerek igen jól használhatók informatikai környezetben: ezen rendszerekben a számítások könnyűek és hatékonyak is egyben, technikailag nem fenyegeti érdemi pontosság-csökkenés.

Cserébe az ECEF számértékei nehezen értelmezhetők egy emberi szemlélő számára: három, akár milliós nagyságrendű metrikus érték, melyből, ha például pontosan északnak indulunk a Föld bármely pontján, akkor legalább kettő, de rendszerint mindhárom érték megváltozik egy függvénykapcsolat szerint, így nem illeszthető a térképolvasó mentális modelljében létező természetes földrajzi irányokhoz.

Az ECEF további hátránya, hogy míg a földrajzi- vagy az ellipszoidi koordináta-rendszerekből való áttérés egyszerű, az ECEF-ből ellipszoidi koordináta-rendszerbe visszatérni számításigényes, mert a formulák nem zártak, így közelítőszámítást kell alkalmazni.

5.6 n-vektor



9. ábra. Az n-vektor értelmezése (Gade, 2010). Az ellipszis az ellipszoid az egyenlítőre merőleges síkú metszetét jelöli; a narancssárga, függőleges vonal a nullmeridián síkja, az ellipszoid kistengelye; a kék, vízszintes vonal az egyenlítő síkja, az ellipszoid nagytengelye; a bíbor nyíl az n-vektor, mely minden esetben merőleges a felszínre, annak normálvektoraként értelmezhető. A 'φ' a geodéziai szélesség.

Az n-vektor használata viszonylag új a geodéziában (Gade, 2010). Az n-vektor célja, hogy egy olyan adattárolási eszközt nyújtson, amely egységesíti az ellipszoidi- és az ECEF koordináta-rendszerek előnyeit. Az n-vektor egy referencia ellipszoid felszínére állított normál egységvektor (9. ábra), mely a vizsgált pontból indul, úgy, hogy a felszínre merőleges. Átváltása földrajzi- vagy ellipszoidi koordináta-rendszerből:

$$n^E = \begin{bmatrix} \sin\lambda \\ \sin\varphi \cdot \cos\lambda \\ -\cos\varphi \cdot \cos\lambda \end{bmatrix},$$

ahol 'n^E' az 'E' koordináta-rendszerben (ECEF) értelmezett n-vektor 1×3-as mátrixa (mely n_x^E-re, n_y^E-ra és n_z^E-re osztható és végeredményben vektor – melyről az eljárás a nevét kapta), míg a 'φ'

és ‘λ’ értékek a szélesség- és hosszúság értékeket jelölik (az n-vektor esetében mind a földrajzi-, mind az ellipszoidi változat használata megengedett).

Vissaztérés földrajzi- vagy ellipszoidi koordináta-rendszerbe:

$$\lambda = \sin^{-1} n_x^E;$$

$$\varphi = \text{atan2}(n_y^E, -n_z^E),$$

ahol az ‘atan2’ az arkusztangens négy-koordinátanegyedes, informatikai megvalósítása.

Az n-vektor használatával egyes számítások leegyszerűsíthetők és beépített matematikai függvénykönyvtárak segítségével elvégezhetők, például egy tetszőleges ‘A’ és ‘B’ pont között mérhető távolság gömbfelszínen mindössze $s = \text{R} \cdot \cos^{-1}(n_{EA}^E \cdot n_{EB}^E)$, ahol n_{EA}^E és n_{EB}^E a két pont n-vektora, még az ‘R’ a Föld sugara (gömbi közelítés esetén).

5.7 A geodéziai dátum

Ahhoz, hogy *ténylegesen* meghatározzuk egy valós földi pont helyzetét, például egy templom tornyának koordinátáit valamelyik koordináta-rendszerben, ahhoz *ténylegesen* meg kell adnunk néhány paramétert. Ezen paraméterek:

1. A használt ellipszoid fél nagytengelyének hossza (‘a’), az ellipszoid lapultságának reciproka (‘$\frac{1}{f}$’). Ezen két adatból meghatározható az ellipszoidunk minden lényeges tulajdonsága, így a fél kistengelyének hossza (‘b’) is: $f = \frac{a-b}{a}, b = a \cdot (1 - f)$.
   Abban az esetben, ha referencia-felszínként gömböt használunk, akkor a = b = R, ahol ‘R’ a gömb sugara.
2. A használt szferoid helyzete a geoidhoz viszonyítva, amely két komponensből tevődik össze. Egyrészt meg kell adnunk a szferoid középpontjának helyzetét a geoid tömegközéppontjához viszonyítva (áttételesen – földi pontok sokaságán kimérve, vagy közvetlenül űrgeodéziai módszerekkel) és meg kell adnunk a referencia síkjaink irányát, így a kezdőmeridiánt és az erre merőleges egyenlítői sík, vagy a forgástengely helyzetét.

A geodéziai dátum koncepcióját végig gondolva könnyen arra juthatunk, hogy a legegyszerűbb eljárás lenne az egész Földre egy egységes, Föld középpontú, Földhöz rögzített geodéziai dátumot használni, egy univerzális, a földi geoidhoz legjobban illeszkedő forgási ellipszoiddal. Ez az elképzelés meg is valósult, ez a gondolat hívta életre például a WGS84-et (World Geodetic System '84 – Geodéziai világrendszer '84) és az ITRF2020-at (International Terrestrial Reference Frame 2020 – Nemzetközi Földi Vonatkoztatási Rendszer 2020).

A geoid azonban egy komplex, girbegurba, hepe-hupás, időben változó forma, amelyhez nem lehetséges úgy szferoidot illeszteni, hogy az mindenhol tökéletesen, vagy legalább egyforma hibával simuljon. Ezért, ha a Föld méreteihez viszonyítva egy relatív kis területet, például egy országot szeretnénk csak feltérképezni, akkor kereshetünk egy olyan méretű ellipszoidot, amely jól illeszkedik az adott vizsgálati területen elhelyezkedő geoid darabkához, de hogy a pontosságot még növeljük a térben úgy is mozgathatjuk, hogy a lehető legjobban simuljon. Ezért előfordulhat például, hogy két különböző geodéziai dátumnak azonos méretű és elnevezésű a szferoidja, de a térben a két szferoid másutt helyezkedik el, mivel középpontjuk- és/vagy referencia síkjaik nem esnek egybe.

Ezért nagyszámú geodéziai dátum van használatban – némelyik csak egy szűk földrajzi területen, némelyik az egész Földön használható. Ezen geodéziai dátumokból nagyon sok elérhető a térinformatikai rendszerekben is.

A geodéziai dátumok nevében gyakran egy évszám is szerepel, pl.: HD72, vagy WGS84. Ez az évszám a dátum megállapításának elvi évszáma, amely egyben tükrözi a megállapításhoz szükséges mérések sokaságának időbeliségét, de a geoid aktuális állapotát is, mert a kőzetlemezek mozgása és a belső tömeg-áthelyeződések miatt a geoid szüntelenül változik, a hálózat megállapítására használt pontok elmozognak. (A dátum megállapítása rendszerint múltbéli időpontra utal, így például 1990-ben hozták létre – az addig feldolgozott mérések alapján – az ETRS89-et [European Terrestrial Reference System 1989].)

A geodéziai dátumok elnevezése néha félreértésre adhat okot. A geodéziai dátumok egyik komponense az ellipszoid, és bár az ellipszoidot a fél nagytengelyének hosszával és lapultságának reciprokával szoktuk jellemezni, ez a leírási forma rendszerint kényelmetlen, mert egy milliós nagyságrendű számot és egy törtet kell lejegyezni, de sok tizedesjegy pontossággal, pl.:

$$R_{\oplus} = 6\,378\,160{,}0 \text{ m}; \frac{1}{f} = 298{,}247167427000022$$

Hogy ezt a kényelmetlenséget csökkentsék, a különböző méretű ellipszoidoknak is nevet adnak. Az előbbi például az IUGG67, mely a HD72-ben használt ellipszoid neve. Más geodéziai dátumokban az ellipszoid neve megegyezik a dátum nevével, például a WGS84 geodéziai dátum ellipszoidja a WGS84 ellipszoid. Viszont például Pitcairn 2006-os geodéziai dátumának ellipszoidja is a WGS84 ellipszoid. Tehát minden esetben meg kell győződnünk arról, hogy egy ellipszoid-, vagy egy geodéziai dátum nevét halljuk-e.

A geodéziai dátum koncepciója egy kísérlet arra, hogy egy, az őt közelítő formát térben a geoidhoz kössük, és ezzel megteremtsük a lehetőséget arra, hogy különböző dátumokon rögzített koordinátákat átszámíthassuk egymásba. Ezt az átszámítási eljárást dátumtranszformációnak nevezzük.

Mivel sem a Föld középpontja, sem a referencia síkok helyzete így a forgástengely döféspontja sem határozható meg terepi méréssel ezért a geodéziai dátumok térbeli helyzetét *egymáshoz* viszonyítva határozzák meg. Kényelmi okokból a térinformatika sok függvénykönyvtára az aktuális geodéziai dátum helyzetét a WGS84 geodéziai dátumhoz viszonyítja és így fogalmaz meg transzlációs, rotációs, vagy skálázási műveleteket.

Ennek analógiájaként az európai vonatkozású geodéziai dátumokat az összeurópai ETRS89 – European Terrestrial Reference System 1989 – nevű geodéziai dátumhoz viszonyítják.

A geodéziai dátum nevében szereplő „dátum" szó latin eredetű és megfigyelésből származó, rögzített tényt, adatot jelöl. Etimológiailag ugyanabból a szóból származik, mint a magyarban használt (időbeli) dátum, csak más jelentés, vagy annak egy más árnyalata társult hozzá. Ennek a szónak a többesszáma az angolban használt data – „adatok" szó is.

A geodéziai dátumot nevezik még geodéziai rendszernek, geodéziai vonatkoztatási rendszernek, geodéziai vonatkoztatási hálózatnak és geodéziai keretrendszernek is.

5.8 Dátumtranszformációk

A különböző geodéziai dátumoknak különböző középpontjuk lehet, különböző szferoid méretekkel, de egymáshoz képest még térben egy-, vagy több tengely mentén elfordultak is lehetnek. Ebből fakadóan, ha van egy olyan pontunk, melynek ismerjük a földrajzi-, vagy ellipszoidi koordinátáit egy adott geodéziai dátum szerint, akkor azok a koordináták nagy valószínűséggel eltérnek egy másik geodéziai dátumon értelmezett koordinátáktól – még akkor is, ha egy és ugyanazon földrajzi pont helyzetét írják le.

Nézzünk egy gyakorlati példát! Adott két, különböző geodéziai dátumunk. Az első szferoidja gömb, sugara $R_{\oplus A}$ = 6 378 137 méter. A második szferoidja szintén gömb, de a sugár $R_{\oplus B}$ = 6 378 160 méter. A két geodéziai dátumot különböző időben, különböző célokra tervezték ezért a gömbök középpontja nem esik egybe: az első geodéziai dátumunk origóját X irányban 57,54479 métert, Y irányban 23,541 métert, Z irányban pedig 17,00789 métert kell eltolnunk, hogy a második dátumunk origójával egybe essen. Szerencsére a referencia síkok egymással párhuzamosak és egy irányba is néznek.

Vegyünk egy pontot a földrajzi térben és határozzuk meg méréssel a földrajzi koordinátáit: eredménynek φ = 46,078082° és a λ = 18,207215°-ot kaptunk.

Fogjuk meg ugyanezt a pontot és mérjük meg a koordinátáit ugyanúgy, de most a második dátumhoz viszonyítva: ekkor a φ = 46,078086° és a λ = 18,207158° eredményt kapjuk.

Az eltérés csekélynek tűnik: $\Delta\varphi$ = −0,000004°, illetve $\Delta\lambda$ =0,000057°.

Mi okozhatja az eltérést? A probléma abból fakad, hogy bár mindkét geodéziai dátum szferoidja gömb, és referencia síkjaik parhuzamosak és egy irányba néznek, a két gömb középpontja nem esik egybe: ezért a vizsgált pont a Föld felszínén ugyanott van, de a vonatkoztatási rendszer térbeli helyzete más.

Természetesen ugyanazon pontunk koordinátáit nem csak méréssel határozhatjuk meg más és más geodéziai dátumon, hanem – ideális esetben – az egyik geodéziai dátumból *átszámíthatjuk* a koordináta értékeket a másikba.

Ezt az eljárást, amikor geodéziai dátumot váltunk számítás segítségével *dátumtranszformációnak* nevezzük. A dátumtranszformáció gyakorlati- és matematikai megvalósítása sokféle, de a legtöbb térinformatikai alkalmazás az alábbi eljárásrendet követi:

1. A kiinduló geodéziai dátumon értelmezett szélesség- és hosszúság értékeket térbeli-, derékszögű koordinátákká váltja át.
2. Ezt a térbeli-, derékszögű koordináta-rendszert úgy transzformálja (például eltolja, átméretezi, vagy forgatja) hogy a referencia síkok és a középpont is egybe essen a cél rendszerrel – miközben a pont helyben marad, csak megváltoznak a koordinátái.
3. Az így kapott, megváltozott koordinátákat visszaalakítja szélesség- és hosszúság értékekké a második geodéziai dátum szferoidjának paraméterei szerint.

Ha áttekintjük a transzformáció folyamatát és visszapillantunk a pontunk két különböző geodéziai dátumon értelmezett −0,000004°, illetve 0,000057°-os koordináta-különbségére, akkor felmerülhet bennünk, hogy ez az eljárás túl bonyolult, miközben az eltérés kicsi.

Nézzük meg, hogy mekkora térbeli hibát szednénk össze, ha a dátumváltást dátumtranszformáció nélkül tennénk meg és az egyik dátumon értelmezett szélesség- és hosszúság értékeket egyszerűen *változatlanul* átvinnénk a másik dátumra, tehát szimplán azt mondanánk, hogy ne törődjünk a sugár-eltéréssel és a középpont helyzetében észlelhető különbségekkel.

Induljunk ki a kapott −0,000004°, illetve 0,000057°-os koordináta-különbségből!

$$1° \approx 111\,000 \text{ m}; -0{,}000004° \approx -0{,}444 \text{ m}; 0{,}000057° \approx 6{,}327 \text{ m}; \sqrt[2]{(-0{,}444 \text{ m})^2 + 6{,}327^2}$$
$$\approx \underline{6{,}343 \text{ m}}$$

A több mint hatméteres hibából látható, hogy legtöbb dátumváltás esetén *a dátumtranszformáció rendszerint nem elhagyható*.

A fentebb bemutatott számértékek csak példák, a valóságban ennél jóval nagyobb, de csekélyebb hibák is előfordulhatnak egyes dátumok között, ha elhanyagoljuk a dátumtranszformációt és

egyszerűen csak átmásoljuk a szélesség- és hosszúság értékeket. Azt, hogy ez a hiba elfogadható vagy elfogadhatatlan, azt az adott művelet jellege dönti el.

A dátumtranszformáció elhanyagolását sok térinformatikai rendszer „nulltranszformáció”-nak nevezi.

5.9 Koordináta-rendszerek és transzformációk azonosítása

Az elvi kereteken túl a következőkben nagyszámú geodéziai dátummal, ellipszoidi koordináta-rendszerrel, transzformációval és síkvetülettel fogunk megismerkedni. Ezen rendszerek nyilvántartása összetett feladat, mert időpontokat és geodéziai paramétereket pontosan és jól nyomonkövethetően kell dokumentálni. Korábban ezt a dokumentációs feladatot minden szervezet és állam önállóan végezte, jelentősen megnehezítve a geodéziai adatok államok közötti átjárhatóságát. Ezen probléma elkerülése végett alapították a European Petroleum Survey Group (EPSG) Geodetic Parameter Dataset-et, melyben minden dátum, vetület, meridián, alapfelület stb. egy egyedi számmal azonosított, a számhoz tartozóan az adott rendszer paraméterei jól dokumentáltak. Ez a szám a térinformatikai rendszerekben, gyakorlatilag gyártótól és szervezettől függetlenül jól azonosítja az egyes vonatkoztatási rendszereket. Így például a Magyarországon használt Egységes Országos Vetület (EOV) azonosítója EPSG 23700. Ez a számot beütve szinte az összes térinformatikai rendszer az EOV-ra áll rá. Az EPSG-n kívül ritkán más szervezet, például az ESRI – Environmental System Research Institute is ad ki az EPSG azonosítóival nem ütköző azonosítószámokat. Az EPSG, az ESRI és mások által kiadott, egymással nem ütköző azonosítókat összességében WKID-nak, Well-Known ID-nak nevezzük.

5.10 Magyarországon használt dátumok és dátumtranszformációik

Magyarországon ugyan számos geodéziai dátum van, vagy volt használatban, de a gyakorlatban három az, amelyet igen széles körben használunk:

1. Egyik a GNSS-eknek köszönhetően a WGS84, amely egy globális geodéziai rendszer. Értékeivel rendszerint ellipszoidi szélesség és hosszúság formájában találkozhatunk. A WGS84 dátum ellipszoidját *is* WGS84-nek nevezik. A WGS84-kez kötődő ECEF koordináta-rendszerben zajlanak a GNSS eszközök belső számításai.
2. A másik gyakori geodéziai dátum a HD72, amellyel közvetlenül nagyon ritkán, közvetve viszont annál gyakrabban találkozunk. A HD72 (Hungarian Datum 1972) jelenti az EOV (Egységes Országos Vetület) alapfelületét, így a vetített, EOV síkkoordinátákból inverz vetítéssel visszaállíthatók a HD72 ellipszoidi koordináták. A HD72 geodéziai dátum meghatározásánál az IUGG67-es ellipszoidot használták.

3. ETRS89 – European Terrestrial Reference System 1989. Az ETRS89 értékeivel rendszerint ellipszoidi szélesség és -hosszúság formájában találkozhatunk olyan GNSS rendszerek kimeneteként, amelyek RTK (Real-Time Kinematic) korrekciót használnak. Az ETRS89 egy olyan speciális vonatkoztatási rendszer, mely kontinensvándorlás mértékét követve együtt mozog az Európai kőzetlemezzel.
   Ezzel a kontinensvándorlás miatt korábban felmért pontok pozíciója nem változik. Így, ha felmérünk egy saját mozgással nem érintett pontot, feljegyezzük a koordinátákat ETRS89-ben, majd a mérést megismételjük öt évvel később ugyanott, akkor a koordinátáknak (elvileg, közel-) változatlannak kell lenni. Ugyanez nem mondható el a WGS84-ről: az űrből szemlélve a WGS84 hálózata állni látszik, míg Európa folyamatosan mozog, így, ha WGS84-ben dolgozunk, az előbbi kísérlet során még elvileg sem kell azonos koordinátákat kapnunk az évek múltával. (Térinformatikai szemszögből szemlélve, figyelembe véve a mérőműszerek, pl. globális navigációs rendszerek pontosságát is az így keletkező eltérések minimálisak, gyakorlati problémákat évtizedes távlatokban okoznak.)

5.10.1 WGS84 ↔ HD72

A WGS84 és a HD72 különböző méretű és különböző elhelyezkedésű szferoidokat használnak. A WGS84 fél kistengelyének hossza 6 378 137 m, lapultságának reciproka 298,257223563. A HD72 ellipszoidján a fél kistengely hossza 6 378 160 m, lapultságának reciproka 298,247167427. Mivel a két ellipszoid lapultsága is eltér, nem lehet az egyik ellipszoidon értelmezett szélesség- és hosszúság értékeket transzformáció nélkül átvinni: a két ellipszoid között alaki eltérés van.

A HD72 ellipszoid térbeli helyzetét úgy határozták meg, hogy az a legjobban simuljon ahhoz a geoid darabkához, amely Magyarország területére esik, ezért a két ellipszoid (a HD72 és a WGS84) középpontja nem is esik egybe és egymáshoz viszonyítva el is fordultak, tehát a két ellipszoid között helyzeti eltérés is van.

Ezen két eltérésből fakadóan dátumtranszformációt kell alkalmazni:

- HD72 → WGS84 irányban először a HD72 ellipszoidi koordinátákat átszámoljuk a HD72-ben értelmezett ECEF koordinátákká, majd ezeket egy megadott paraméterkészlet alapján transzformáljuk. Az így kapott koordináták már a WGS84-hez kötött ECEF koordináta-rendszerben értelmezett értékeknek tekinthetők. Végül ezeket az értékeket átszámítjuk WGS84 ellipszoidi koordinátákká.
- WGS84 → HD72 irányban először a WGS84 ellipszoidi koordinátákat átszámoljuk a WGS84-ben értelmezett ECEF koordinátákká, majd ezeket transzformáljuk. Az így kapott

koordináták már a HD72-höz kötött ECEF koordináta-rendszerben értelmezett értékeknek tekinthetők. Végül ezeket az értékeket átszámítjuk HD72 ellipszoidi koordinátákká.

Ezen dátumtranszformáció neve a térinformatikai szoftverekben „HD72 to WGS 84 (4)" azonosítója 1242 (EPSG), általánosságban pedig háromparaméteres, geocentrikus transzlációnak nevezzük. Ez a transzformáció nem veszi figyelembe a két ellipszoid egymáshoz viszonyított elfordulását. A használt konstansok ΔX = 52,17 m, ΔY = −71,82 m, ΔZ = −14,9 m a HD72→WGS84 irányban. Tehát a HD72-ben értelmezett ECEF koordinátákhoz ezen értékeket hozzá kell adni, majd a kapott értékek WGS84 szélességé/hosszúsággá alakíthatók. A transzformáció a konstansok kivonásával fordítva is elvégezhető. A térinformatikai rendszerek ezt a transzformációt rendszerint automatikusan végzik. A legtöbb szoftverben ez az alapértelmezett eljárás.

Ezen transzformáción kívül létezik még egy „HD72 to WGS 84 (3)" 1448 (EPSG) azonosítójú transzformáció is, mely hét paramétert, három geocentrikus transzlációs, három térbeli elforgatási és egy skála paramétert is tartalmaz. Paraméterei: ΔX = −23,8085 m; ΔY = −17,5937 m; ΔZ = −17,801 m; RX = −0,3306″; RY = −1,85706″; RZ = 1,64828″; S = 5,4374 ·10^{-6}.

(Semmi szín alatt se használjuk az elavult és igen pontatlan, mára már csak kompatibilitási okokból megőrzött „HD72 to WGS 84 (1)" és „HD72 to WGS 84 (2)" transzformációkat!)

Mint minden transzformációs eljárás, a WGS84→HD72/HD72→WGS84 áttérés is hibával terhelt, amely akár az egy métert is megközelítheti. (A paraméterek forrása: ESRI [2024])

5.10.2 A WGS84 realizációi

A WGS84 képezi a GPS rendszer vonatkoztatási hálózatát. Kiadása óta a WGS84-et több alkalommal is pontosították. Ezen pontosítási lépéseket realizációknak nevezik. Így a WGS84 ernyőfogalommá vált. A WGS84-et kimeneti formátumként használó GNSS rendszerek mindig a WGS84 *aktuális* realizációjában mérnek, melyet a műholdrendszer pályaadataiban, az adott pillanatban közöl. Ezidáig az alábbi realizációkat használták (6. táblázat):

6. táblázat. A WGS 84 realizációi (1987–2024). Adatok forrása: United States National Geospatial-Intelligence Agency (2024): WGS 84 (G2296) Terrestrial Reference Frame Realization (US National Geospatial-Intelligence Agency, 2024).

Megnevezés	Orbitális adatok közlésének kezdete
WGS84	1987
WGS84 – G730	1994. jún. 29.
WGS84 – G730	1997. jan. 29.

WGS84 – G1150	2002. jan. 20.
WGS84 – G1674	2012. febr. 08.
WGS84 – G1762	2013. okt. 16.
WGS84 – G2139	2021. márc. 28.
WGS84 – G2296	2024. jan. 07.

Ezen realizációk technikailag olyan más és más geodéziai dátumoknak minősülnek, melyek referencia ellipszoidjainak mérete azonos (WGS84 ellipszoid), de térbeli helyzetük más és más. Például ha WGS 84 (G2139)-ből WGS 84 (G2296)-ba szeretnénk áttérni, akkor az alábbi hétparaméteres transzformációt kell elvégeznünk: ΔX: 2,6 mm; ΔY: 5,4 mm; ΔZ: −0,9 mm; S: 0,6 $\cdot 10^{-10}$; RX: −0,01 mas; RY: −0,07 mas; RZ: 0 mas. Ez a gyakorlatban azt jelenti, hogy a hálózat, kb. 6,33 milliméterrel elmozdult, jelentéktelen mértékben a pontokat távolítani kell az origótól ('S' komponens) és jelentéktelen mértékben elfordult (1 mas = $\frac{1}{3600000}$ °; mas: milliarcsecond, ezred szögmásodperc). A térinformatikai szoftverek ezen realizációkat rendszerint nem veszik figyelembe (mert gyakorlati hatásuk elhanyagolható) és az egyes realizációkat egységesen, összevontan „WGS84"-nek tekintik. Tényleges elkülönítésük csak igen precíz, speciális geodéziai számítások során szükséges. A WGS84, mint összefogó, ernyő-rendszernek (a realizáció megkülönböztetése nélkül) az EPSG azonosítója 4326. ECEF rendszerének azonosítója EPSG: 4978.

5.10.3 ETRS89 ↔ HD72

Abban az esetben, ha magyarországi RTK szolgáltató GNSS pontosítását vesszük igénybe, nagy az esélye, hogy az ellipszoidi koordinátáinkat ETRS89-ben kapjuk. Mivel az ETRS89 ellipszoidja a WGS84 ↔ HD72 fejezetben leírtakhoz hasonlóan alaki és helyzeti értelemben is eltér a HD72-től, így az ETRS89→HD72 áttérés során ugyanúgy transzformációt kell végezni.

Erre két eljárás létezik:

1. Alkalmazhatjuk egy a WGS84 ↔ HD72 bemutatott parametrikus transzformációval elvi szinten megegyező megoldást: itt is három-, vagy hét paramétert vehetünk fel, mellyel a helyzeti különbségek legyőzhetők. Jelenleg a térinformatikai szoftverekben egy hétparaméteres eljárás érhető el „HD72 to ETRS89 (2)" (EPSG: 1449) néven, mely mentén a transzformáció megvalósítható. Ne használjuk az elavult „HD72 to ETRS89 (1)" (EPSG: 1829)-es transzformációt!
   (Speciális mellékinformáció, hogy mivel az ETRS89 és a WGS84 realizációja az ETRS89

rögzítésének pillanatában nagyon közel volt egymáshoz, mind a két ellipszoid méretét-, mind azok helyzetét tekintve, ezért a „HD72 to ETRS89 (2)" (EPSG: 1449) és a „HD72 to WGS 84 (3)" (EPSG: 1448) paraméterei azonosak, a térinformatikai szoftverek ezen beállítások mentén azonosnak tekintik a két geodéziai dátumot.)

Mint minden dátumtranszformáció, így az ETRS89→HD72 is hibával jár.

Ezt a transzformációs hibát a gyakorlatban úgy kell elképzelni, hogyha van egy geodéziai alappontunk, melynek koordinátáit ETRS89-ben és HD72-ben is ismerjük, majd egy térinformatikai alkalmazásban elvégezzük az ETRS89→HD72 transzformációt, akkor a kapott eredmény nem fog egyezni az ismert, fizikailag állandósított HD72 koordinátákkal. Ez az eltérés a „HD72 to ETRS89 (2)" (EPSG: 1449) hétparaméteres transzformációval Magyarország területén helyfüggően változó, de kb. max. 0,4 m. A kontinensvándorlás miatt – elvi szinten – az idő előrehaladtával romlik.

2. Alkalmazhatunk ETRS89→HD72 korrekciós rasztert. A fentebb bemutatott többparaméteres eljárás csak addig jelent jó megoldást, amíg nincs szükség geodéziai pontosságra.

   A korrekciós raszterek használatának megértéséhez végig kell gondolnunk néhány gyakorlati problémát. Ehhez tételezzük fel, hogy olyan GNSS-t használunk, amely RTK korrekcióval rendelkezik és így a mérési eredményeket ETRS89-ben kapjuk. Magyarország területén tartózkodunk, így mérésünk eredményét HD72/EOV-ban kell közölnünk. Tehát valamilyen ETRS89→HD72 transzformációt kell végeznünk, melyekről korábban megállapítottuk, hogy hibával terheltek és a hiba mértéke helyfüggően változik. Ezen kívül más gyakorlati problémával is találkozhatunk. Az EOV hálózatát geodéziai alappontok mentén állandósították. Ezek olyan jól hozzáférhető terepi pontok, melyek koordinátája korábbi felmérésekből ismert. Ezek a pontok részben már eleve kismértékben, de hibásan lettek felmérve, részben az idő múlásával elmozogtak a kontinensvándorlás mértékét leszámítva is – vagy eleve hibával terhelten kerültek felmérésre. Az ETRS89 elvi szinten kiküszöböli a kontinensvándorlás hatását, de nem terjed ki a sajátmozgásra és a korábbi mérési hibákra.

   Ebből olyan érdekes ellentmondások keletkeznek, hogyha RTK GNSS eszközzel rá akarunk állni egy geodéziai alappontra, akkor, amikor a GNSS az alappontéval egyező koordinátákat mutat a kijelzőjén, akkor nem az alappont felett, hanem akörül, annak közvetlen közelében leszünk.

   Ebből olyan látszólagos hibák keletkeznek, hogy például egy épület sarkának pontosan egy méterre kell lenni az alappontól – és ez a valóságban így is van, mérőszalaggal a helyszínen lemérhető – de a GNSS-el mért (elvileg) ugyanazon koordinátaponttól csak 0,7 méterre van. Tehát a fizikailag észlelhető, tapintható valóság nem egyeztethető össze a

mért (és szintén helyes) koordináta-geometriai értékekkel. Ráadásul, ha ezt az összehasonlítást az ország több pontján is elvégezzük, az elcsúszás mértéke mindenütt más és más lesz, mert az alappontokat más mérési hiba terhelte, de az idő múlásával más és más mértékben mozogtak el egymáshoz, az elvi Európai kőzetlemezhez, de még egymáshoz képest is.

Ezen probléma legyőzésére vezették be a korrekciós rasztereket, melyek leírják, hogy adott földrajzi pontban mennyit- és milyen irányba kell csúsztatni a GNSS mérést, hogy a tapintható valósággal összeegyeztethető legyen a terepen. Magyarországon jelenleg erre – hivatalos célokra – a VITEL2014 nevű rasztert használják. A raszter nevében lévő évszám nem véletlen: egy elvi időpontba rögzített állapotot mutat, amely az idővel természetesen változik, hisz az alappontok egyedileg, a GNSS hálózat pedig szisztematikusan elmozog.

Így a pontossági igények függvényében kell kiválasztanunk, hogy melyik eljáráshoz nyúlunk.

Az itt leírtakon kívül természetesen még számos geodéziai dátum és transzformáció létezik a világ különböző pontjain, melyeket a helyi, geodéziai helyzet ismeretében kell kiválasztani. Ezek egy része az EPSG regiszterében is szerepel.

5.11 Geoid unduláció: a geoid-modellek

Képzeljük el, hogy egy kiszáradt sós tó medrében állunk, amely *tökéletesen* vízszintes. Az a feladatunk, hogy GNSS-el fel kell mérnünk a tó medrének kb. ⅔-át és feljegyeznünk az ellipszoidi koordinátákat, valamint az ellipszoid feletti magasságot. A tavunk elég kiterjedt, kb. 592 km^2. Ha ebben a *tökéletesen* vízszintes medrű tóban elindulunk és 2–5 méterenként felveszünk egy-egy pontot, akkor egy érdekes jelenséget figyelhetünk meg: ahogy haladunk előre, az ellipszoid feletti magasságunk lassan, de folyamatosan változik – miközben a mederfenék *tökéletesen sík és vízszintes* is egyben. Ilyenkor a valóság látszólag ellentmond a mért értékekkel: miért változik az ellipszoid feletti magasság, hogyha a felszínünk vízszintes és sík is?

A válasz a Föld alakjában és a földi gravitációs mező aktuális inhomogenitásaiban keresendő, melyet a *geoid* ír le.

A *geoid* egy olyan elméleti konstrukció, ami a Föld aktuális alakját jeleni. Ez egy összetett forma, melyet a geodéziai dátumban meghatározott helyzetű- és formájú ellipszoiddal *közelíthetünk meg*. Ez a megközelítés azonban szükségszerűen pontatlan: a geoid néhol alá-, néhol fölé vág a geodéziai dátumban definiált ellipszoidunknak. Ezt a jelenséget nevezzük *geoid undulációnak*.

Idáig a geoid geometriai jellemzőit vizsgáltuk: alak, méret, középpont. Azonban vizsgáljuk meg a geoidot geofizikai szemszögből is.

Először tételezzük fel, hogy az ω szögsebességgel forgó Földön minden anyag képlékeny, melynek sűrűsége univerzálisan $\rho \cong 5{,}51\,\frac{kg}{m^3}$ és az eloszlása teljesen homogén is egyben. Ekkor a bolygó tökéletes forgási ellipszoid formát venne fel. Ekkor meg lehetne adni egy olyan geodéziai dátumot, melynek ellipszoidja tökéletesen illeszkedne a bolygó felszínéhez, és minden pontban igaz lenne, hogy a helyi függőleges pontosan merőleges lenne az ellipszoid felszínére. Ekkor, ha az ellipszoid felszínétől egységnyire elemelkednénk, akkor mindenütt ugyanakkora munkát kellene végeznünk. Ebből az is következik, hogyha az ellipszoid felszínén mozgunk, akkor egyáltalán nem kell a gravitáció ellenében munkát végeznünk: a felszín mindenütt vízszintes, és mivel vízszintesen mozgunk, nem végzünk munkát, nem kapaszkodunk felfelé és nem ereszkedünk lefelé.

Így egységnyi magasságra mindig egységnyi munkával emelkedhetünk el a felszíntől, ezért ezt a képzeletbeli felszínt *ekvipotenciális* felületnek, vagy *szintfelületnek* nevezzük. A szintfelületeken a gravitációs gyorsulás értéke állandó.

A Föld – anyagi összetételét tekintve – azonban nem homogén, ezért az ekvipotenciális felület alakja nem forgási ellipszoid, hanem *geoid* – amely egy helyről-helyre változó összetett forma. A teljes nyugalomban lévő homogén óceán – és peremtengerei – erre a szintre állnának be. Ahol az ekvipotenciális felületünk a referencia ellipszoidunk alá vág, ott *negatív*-, ott, ahol pedig fölé emelkedik, ott *pozitív geoid undulációról* beszélünk.

Ezen összefüggés miatt alkalmazható a tengerszint a geodéziában: ha elég sokáig figyelünk egy mareográfot, akkor megkaphatjuk az adott pontban értelmezett 0 szintet. Innen, ha mindig a helyi függőlegesre merőleges sík – a vízszintes sík – mentén, kis lépésekben mérünk, akkor a tengerszint, így az ekvipotenciális felület felett mért magasság messze, a szárazföldek belsejébe is bevihető, majd onnan magassági alappontok formájában nagy területre kiterjeszthető. Ezzel a geoid ekvipotenciális-/szintfelületéhez köthetjük a magasságméréseinket.

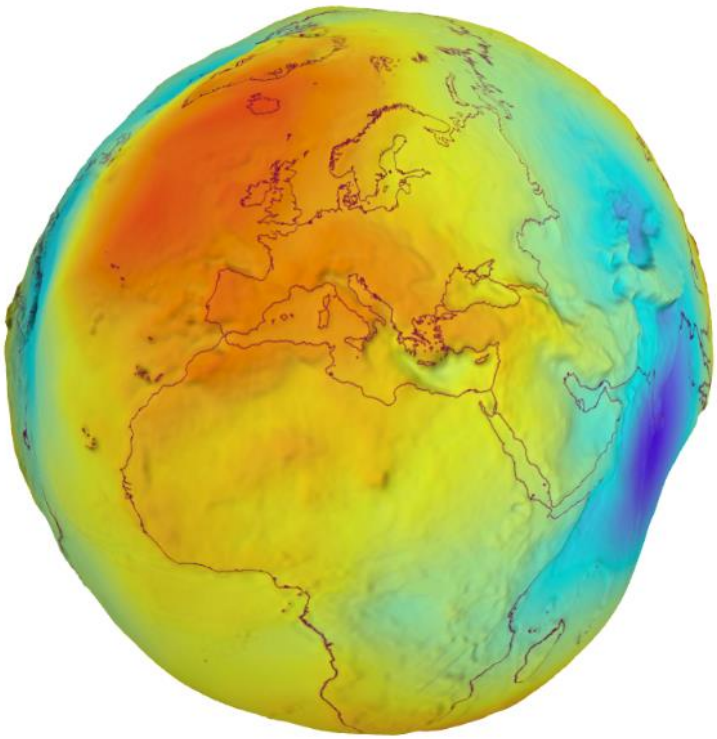
Ha a képzeletbeli, vízszintes tavunkban a magasságot nem az ellipszoidtól, hanem ezen geoidként kijelölt ekvipotenciális felülettől mérnénk, akkor a magasságunk szám szerint nem változna: a tó immáron a mért értékek szerint is vízszintes lenne.

A szintfelület felett mért magasság használata azért nagyon praktikus, mert így a magasságértékek összeegyeztethetők pl. a víz terepi mozgásával. Ezen szintfelülethez kötött elképzelés teszi például lehetővé, hogy egy folyó esését több száz kilométeres szakaszra kiszámíthassuk és ez alapján megbecsülhessük a folyó mozgási energiáját. Abban az esetben, ha a magasságokat egy

inhomogén anyagi összetételű- és sűrűségű bolygón ellipszoid feletti magasságokként értelmeznénk, akkor akár olyan irreális számokat kaphatnánk, amelyek azt állítanák, hogy egy folyó dombnak felfelé mászik, miközben a valóságban végig lejtő mentén folyik lefelé.

Az ekvipotenciális felület használatának van egy érdekes következménye: az ekvipotenciális felület nem minden esetben párhuzamos az ellipszoid felszínével, ezért a helyi függőleges (amerre az inga lóg) ugyan mindig merőleges a helyi vízszintesre, így az ekvipotenciális felületre, de nem mindig merőleges a referencia ellipszoidra. Ezt a jelenséget *függővonal-elhajlásnak* nevezzük.

A geoid formájának meghatározása és a geoid unduláció mértékének meghatározása azonban nem csak hagyományos, optikai, tengerszinthez kötött módszerekkel oldható meg: a műholdas távérzékelés segítségével az elvipotenciális felület alakja közvetlenül is meghatározható. Ilyen modell például a széles körben használt EGM96 (Earth Gravitational Model – Földi Gravitációs-modell '96) amely a WGS84 geodéziai dátumában rögzített ellipszoid felszínéhez képest adja meg a geoid unduláció mértéket. Az EGM96 egy 721 sorból és 1440 oszlopból álló raszter, amely 15´×15´-es felbontásban írja le az unduláció mértékét – méterben kifejezve (10. ábra).



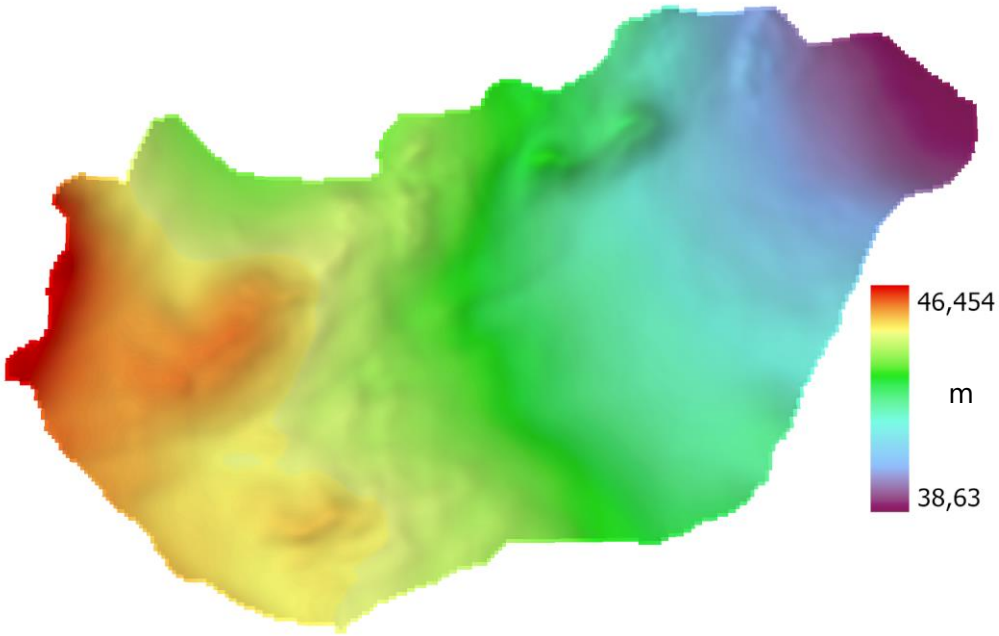
10. ábra. Az EGM96 geoid undulációs modell. Az EGM96 a WGS84 ellipszoidhoz viszonyítva írja le az ekvipotenciális, „nulla" szint helyzetét. Ahol az ábra kék, ott a szintfelület a WGS84 ellipszoid alá bukik (negatív geoid unduláció), ahol zöld, ott körül-belül az ellipszoid felszínével egybe esik, ahol narancs, vagy vörös, ott az ellipszoid felszíne fölé esik (pozitív geoid unduláció). A legjelentősebb negatív geoid unduláció az Indiai-óceánban, az Indiai-szubkontinens előterében helyezkedik el: itt a nulla szintfelület –106,99 méterre helyezkedik el WGS84 ellipszoid felszínéhez képest (az ábra jobb oldalán). A Kárpát-medencében az unduláció értéke kb. +44,5 méter. A legnagyobb pozitív geoid unduláció Új-Guinea szigetén található +85,391 méterrel (az ábrán nem látható). Mivel ezen eltérések a WGS84 ellipszoid méretéhez képest elenyészők, ezért a láthatóság kedvéért az értékeket 7 500×-os torzítással jelenítettük meg. A torzítás érdekes, pozitív következménye, hogy láthatóvá teszi a függővonal-elhajlást. Nézzük meg a Földközi-tenger medencéjét Krétától délre: a domborzat-árnyékolásnak köszönhetően látható, hogy helyi nulla szintfelület normálvektora nem esik egybe az ellipszoid normálvektorával (Office of Geomatics, National Geospatial-Intelligence Agency, 1996).

A geoid feletti magasság leírására számos tengerszint és modell létezik, melyek különböző tengerekhez, különböző időpillanatokban, és különböző mérésekhez kötődnek, ezért bár a mögöttes, ekvipotenciális tulajdonság azonos, a kijelölt 0 szintek különböző pontokban, különböző magasságban lehetnek a referencia ellipszoid felet. Így a Magyarországon történetileg használt Balti- és Adriai alapszint sem esik egybe: a Balti alapszint 67,47 centiméterrel magasabban van az Adriai felett (melyet a Nadapi főalapponton állandósítottak, majd ezután onnan mérték).

Ha térinformatikai rendszerekben transzformációt végzünk meg kell győződni róla, hogy a magassági adatok transzformációja is helyes-e, mert a legtöbb térinformatikai rendszer a magassági adatokat érintetlenül hagyja és csak a szélességet és a hosszúságot transzformálja.

5.12 Magyarországon használt geoid modellek és a magassági adatok transzformációja

Magyarországon az EOMA – Egységes Magassági Alaphálózathoz mért magasságokat használjuk a magassági adatok elsődleges referencia-szintjeként. Ez ugyan egy fizikailag állandósított, magassági alappont-hálózaton nyugszik, de gyakorlatilag egy, csak Magyarország területén használható geoid modellnek is tekinthető, melynek felbontása magasabb a globális geoid modelleknél (11. ábra).



11. ábra. 268×186 pixel felbontású EOMA geoid, mely a WGS84 ellipszoidja feletti unduláció mértékét mutatja – méterben. A magasságadatok tízezerszeresen torzítottak, melyekre többirányú domborzatárnyékolás került. Nyers adatok forrása: (Takács & Siki, 2017).

Egy felmérési munkafolyamatban veszünk rész. Például egy vízfolyást és annak környékét kell felmérnünk, melyből vektoros eredményeket is válnak és a mérést ki kell egészítenünk egy olyan ortofotóval és egy digitális felszínmodellel melynek a terepi felbontása kb. 2 cm. A területen vannak előzetesen felmért műtárgyak, melyek magassága EOMA-ban, horizontális pozíciója EOV-ban adott. A légitérképezést drónnal végezzük, mely GNSS-el határozza meg térbeli pozícióját így horizontális pozícióként WGS84-es geodéziai dátumon nyugvó értékeket ad, magassági adatait pedig EGM96 geoidhoz igazítva (a GNSS-ek más geoid modellhez is igazodhatnak, esetleg nyersen, WGS84 ellipszoid feletti magasságban is jelenthetnek). Ehhez képest a leadandó digitális felszínmodell raszterének EOMA szerintinek kell lenni, hogy egyezzen a korábban felmért műtárgyak EOMA magasságával.

Légitérképezésünket kézi geodéziai GNSS-el is támogatjuk, mely az egyszerűség kedvéért szolgáltassa pozícióadatait ETRS89-ben.

Látható, hogy a forrás- és a cél vonatkoztatási rendszerek nem egyeznek, azok geodéziai dátuma sem azonos, ezért transzformációt kell végeznünk (Dátumtranszformációk & Magyarországon használt dátumok és dátumtranszformációik). De ezen transzformációk csak a horizontális koordinátákra terjednek ki. Mi lesz a magassági adatokkal?

A magassági adatok helyes átszámításához egy különbség-rasztert kell képeznünk az EGM96 és az EOMA között. Meg kell vizsgálni, hogy a kapott különbség-raszteren az átszámítandó vektoros pont hova esik. Itt bilineáris-, vagy cubic convolution (lsd.: 8.3 Újramintavételezés) interpolációval fel kell vennünk a különbség-raszter adott pontban észlelt értékét. Ezt az értéket hozzá kell adni a bejövő mért pont magasságához. Szerencsére ezt a műveletet az esetek többségében a térinformatikai szoftverek elvégzik helyettünk, ha tisztában vagyunk a konverzió szükségességével.

6 Vetületek a térinformatikában

A Föld aktuális alakja a geoid: ez egy komplex változó forma, melyet gömbbel vagy forgási ellipszoiddal közelíthetünk. Mind a gömb, mind a forgási ellipszoid háromdimenziós, valódi térfogattal rendelkező test. A térinformatikában használt monitorok és kijelzők viszont síkok, olyan kétdimenziós megjelenítő eszközök, mint a hagyományos kartográfia papírlapja.

Háromdimenziós objektumokat nem jeleníthetünk meg egy kétdimenziós kijelzőn: a dimenzió számát csökkenteni kell, az objektumot síkba kell lapítani. A síkba fejtés adatvesztéssel és torzulásokkal jár.

A síkbafejtés eljárását vetítésnek, vagy projekciónak nevezzük. A vetítés során fellépő torzulás a vetítési eljárástól függ. A térképeken, a térkép céljának megfelelően megpróbálhatjuk csökkenteni a torzulás mértékét – vagy csak esztétikusabbá tenni a térképet –, így különböző felhasználási célokhoz, különböző vetületeket rendelhetünk. A vetítési eljárások száma igen nagy és mindegyik más és más torzulási viszonnyal jellemezhető. A vetületek csoportosíthatók a képfelület típusa szerint, a Föld formáját közelítő ellipszoid és a képfelület egymáshoz viszonyított helyzete szerint és a koordináta-transzformáció elve szerint, valamint ezek eredőjeként fellépő torzulás milyensége szerint.

6.1.1 Térképi torzulások

A síkvetületek a háromdimenziós valóságot a síkba fejtik. Ennek során a valóság leképezett képe torzul. A vetítési eljárások azonban úgy is megválaszthatók, hogy a valóság legfeljebb egy, kiválasztott tulajdonsága megmaradjon. Ezek alapján az alábbi vetületeket különítjük el:

1. **Területtartó vetületek**. Ha a valós felszínen felmérünk két vizsgálati területet, majd a vetített képen is kiszámítjuk a területüket, akkor kapott érték aránya azonos lesz. Cserébe az alakok, a formák, a szögek és a hosszak is (akár jelentősen) torzulnak.
   Ott válasszunk területtartó vetületeket, ahol a földrajzi terület mérete számít, például a népsűrűség térképek, vagy a besugárzási térképek esetében, vagy ott, ahol országokat kell területük alapján rangsorolni.
   Területtartó vetületeken ne kíséreljünk meg navigálni, mert ha a valóságban egy adott szög alatt látunk egy objektumot magunk előtt, ez a szög nem fog megegyezni a térképen leolvasható szöggel. A legtöbb területtartó vetületen nem lehet univerzális északi irányt kijelölni, hanem az helyről-helyre változik, annak függvényében, hogy hol vizsgáljuk a térképet.

2. **Szögtartó vetületek**. A szögtartó vetületeken a terepen látható irányszögek megegyeznek a térképről leolvasható irányszögekkel. Így például, ha terepen egy tornyot látunk magunk előtt azimut szerinti 310°-ra, majd megvizsgáljuk a térképen az álláspontunkat a toronnyal összekötő egyenes irányszögét, akkor ott is 310°-ot kapunk. Ezért a szögtartó vetületek nagyszerűen alkalmasak navigációs célokra. Cserébe a területeket akár drámai (több száz százalékos) mértékben is torzíthatják.
   Vigyázat! A szögtartó vetületeken kijelölt két pontot összekötő egyenes hosszú távokon nem jelöli a két pont közötti legrövidebb utat! Szögtartó vetületeken a két pontot összekötő szakasz irány igen, de szakasz hossza nem helyes. Így például, ha repülőutat tervezünk, az irányszög leolvasása lehetséges, de a megtett út hossza nem. (A kijelölhető szakasz nem a legrövidebb út, de a hossza még ennek sem helyes!)
   Ne alkalmazzunk szögtartó vetületeket területméretekkel összefüggő jelenségek ábrázolására, pl. a fentebb említett népsűrűség.
3. **Hossztartó vetületek**. A hossztartó vetületek két pont közötti távolságot őrzik meg helyesen, de cserébe torzítják a területet és a szögeket is.
   Vigyázat, a hossztartó vetületek csak kitüntetett irányokban (pl. a meridiánok mentén) vagy kitüntetett pontokon áthaladva (pl. középpont) őrzik a helyes hosszakat.
   Nincs olyan hossztartó vetület, amelyik a térkép egész felületén, tetszőleges pontok közötti távolságokat, kivétel nélkül mindenütt helyesen őrzik.
   A hossztartó vetületek egy fontos használati területe repülés, ahol egy adott ponttól mérve helyes távolságokat olvashatunk le. A hossztartó vetületek egy különleges veszélye, hogy a meridián mentén hossztartó vetületek fokhálózata akár négyzetes is lehet, ezért a térképolvasó helytelenül szögtartónak feltételezi őket.
   A hossztartó vetületek egyik alkategóriája az azimutális ekvidisztáns vetület, amelyen az egy, maximum kettő kitüntetett pont irányából nézve az irányszögek is helyesek. (De csak onnan nézve, tetszőleges pontok között ez már nem igaz!)
4. **Általános torzítású vetületek**. Ezen vetületek egyik fentebbi, kitüntetett tulajdonságot sem őrzik meg hibátlanul, helyette arra törekednek, hogy a torzítás mértéke minden kategóriában minimális legyen. Nagyszerű vetületek falitérképekhez, globális tematikus ábrázolásokhoz. Ne kíséreljünk meg rajtuk semmit sem mérni.

6.2 A helyes vetület kiválasztása

A síkvetületek jelnetős matematikai apparátuson nyugszanak. Térinformatikai környezetben ez a matematikai apparátus azonban előre programozott, így követlen megvalósításával keveset kell foglalkoznunk. A rendelkezésre álló vetületek száma igen nagy. A következő fejezetekben végigvesszük, hogy milyen területeken, milyen vetületeket kell használni és ezen milyen torzítások

jelentkeznek. Az itt bemutatott vetületek valójában vetületi egyenletek, melyek leképezik az ellipszoidi-, vagy gömbi koordináta pontokat a síkban. Ahhoz, hogy tényleges hasznát vegyük egy vetületnek ismerni kell a bemeneti koordináták geodéziai dátumát is. Így az alább bemutatott Albers–féle területtartó kúpvetület nyugodhat WGS84-en, ETRS89-en, ED50-en, de bármilyen az adott területre használható más geodéziai dátumon is.

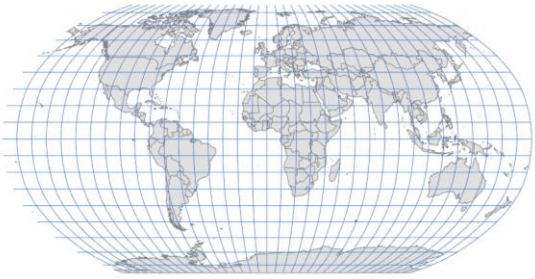
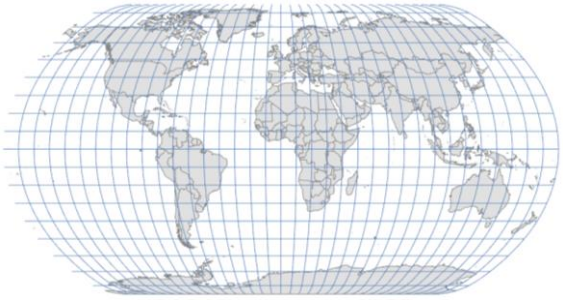
Ezért a térinformatikai rendszerek a vetületi rendszereket geodéziai dátumukkal együtt azonosítják, például az EPSG 23700 HD72 EOV magában foglalja a vetítési eljárást (egyenletet, középpontot és eltolást) és a geodéziai dátum definícióját is.

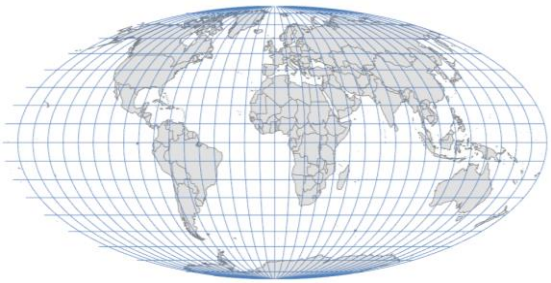
Az egyszerűség kedvéért a következő fejezetekben WGS84-en nyugvó vetületeket mutatunk be.

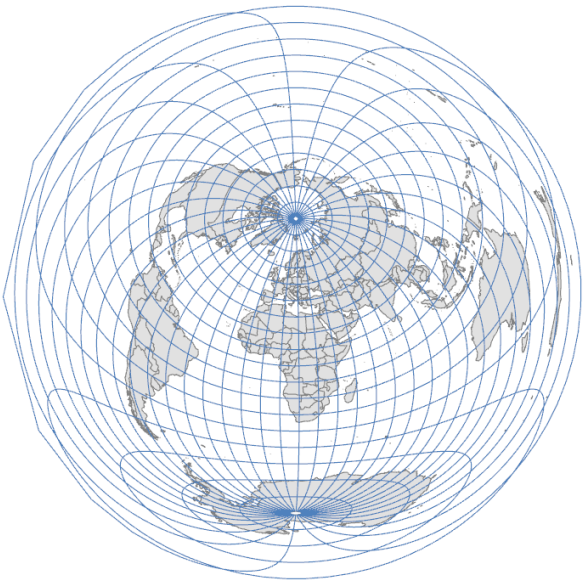
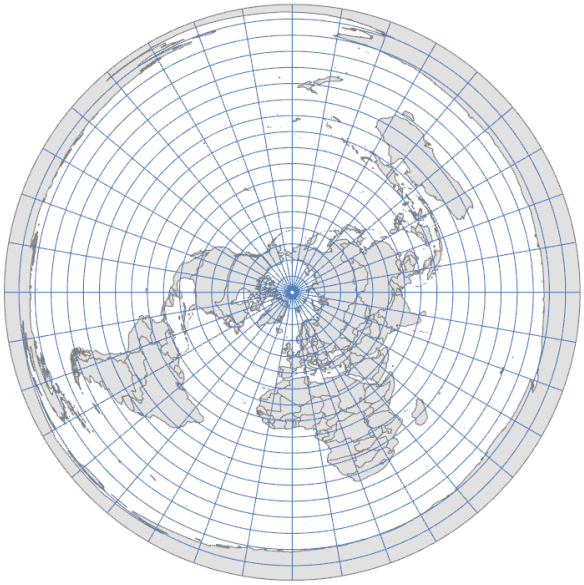
6.2.1 Világvetületek

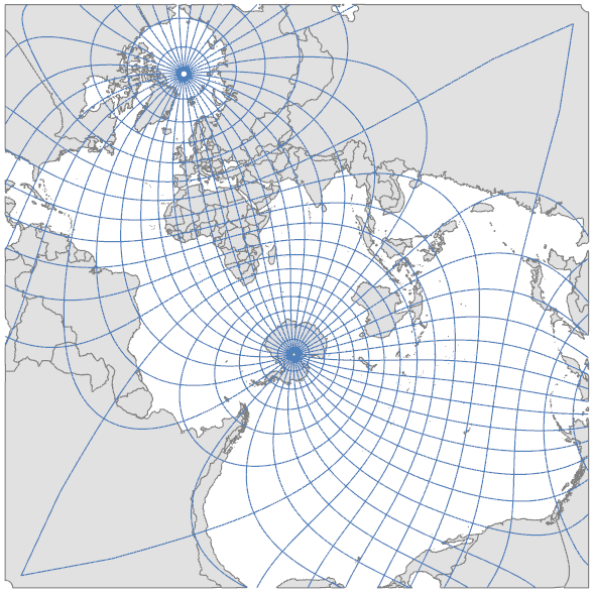
Abban az esetben, ha a Föld teljes területét szeretnénk ábrázolni az alábbi lehetőségel közül választhatunk (7. táblázat):

7. táblázat. Világvetületek. Alapadatok forrása: ESRI (2023).

Általános torzítású világvetületek. Ebbe a családba azon vetületek tartoznak, melyeknek nincs olyan kitűntetett tulajdonsága, ahol a torzítás nulla lenne.	Robinson–féle vetület:  A Robinson–féle vetület a sarkpontokat vonalként képezi le. Általános torzítású világvetület. A meridiánok görbék, a szélességi körök egyenesek. Alkalmas nem területkötött, globális jelenségek ábrázolására. Natural Earth vetület: 

	A Natural Earth a Robinson–féle vetülethez képest kevésbé „nyomja össze" kelet–nyugati irányban a sarkörökön túli terleteket, ezért az északi félteke északi partvonalai jobban kivehetők. Winkel Tripel (Winkel III.):  A Winkel Tripel a Robinson–féle vetülethez képest jóval kevésbé nyomja össze a sarkvidéki területeket észak–déli irányban, ezért a poláris jelenségek jobban láthatók.
Területtartó világvetületek. Ebbe a családba azon vetületek tartoznak, amelyek a területeket őrzik meg helyesen. Olyan esetben használjuk őket, amikor a terület mérete releváns, pl. országok rangsorolására területük szerint, vagy népsűrűség globális ábrázolására.	Mollweide vetület:  A Mollweide egy széles körben használt területtartó világvetület. Globális ábrázilási célokra sokszor optimálisabb választás a térinformatikai környezetben oly gyakran használt Web Mercator helyett.

Távolságtartó (ekvidisztáns) világvetületek. Ebbe a családba olyan vetületek tartoznak, melyek egy, vagy maximum kettő pontból helyesen ábrázolják a távolságokat. Fontos, hogy az ilyen típusú térképeken a távolság csak a kitüntetett pont(ok)on keresztülhaladva helyes: tetszőleges pontok között mérve a többi vetülethez hasonlóan helytelen eredményt ad. Kiválóan használhatók például repülésben, ahol egy pontból nézve a távolság helyesen leolvasható.	Ferdetengelyű, azimutális ekvidisztánsz vetület:  Ez a ferdetengelyű azimutális ekvidisztáns vetület a Pécs–Pogányi Reptér (PEV; 18,2425918°; 45,9888992°) kifutópályájának középpontjára nézve ekvidisztáns. Az innen mért távolságok és irányszögek is helyesek. Ez viszont semelyik más pontban nem igaz. Északi pólusra centrált azimutális ekvidisztáns vetület:  Ez a vetület minden tulajdonságában megegyezik az előzővel, de specialitása, hogy a kitüntetett centruma az Északi sarkponttal esik egybe. Ezen vetület stilizált képét használja az Egyesült Nemzetek Szervezete címerében és

	zászlójában is. Hatalmas peremi torzítása miatt az Antartisz egybefüggő konteinense felszakad és gyűrűként terül el a széleken. A jelenség az azimutális viselkedés következménye: az Antarktisz egyes ponjai tényleg ezen irányokban látaszanak az ellentétes sarkpontról. Bár a vetület ritkán használt, torzulásai miatt a laposföld-hívők az igazság rajzi megtestesülésének tekintik.
Speciális világvetületek. A speciális világvetületek csoportja igen tág, jelentős részük csak esztétikai értékkel bír, ezért itt csak egy, a természetföldrajzban és az oceanográfiában jól hasznosítható vetületet vizsgálunk meg.	Spilhaus–vetület:  A Spilhaus–vetület a szárazföldeket igen torzan ábrázolja, de az összefüggő óceáni területek torzítása alacsony és megszakítás-mentes, ezért például a tengeráramlások megszakítás nélkül ábrázolhatók, így a lég- és vízkörzési ciklusok jobban szemléltethetők.
Szögtartó világvetületek. Olyan szögtartó világvetületek, melyek *ténylegesen* (a sarkpontokat is beleértve), az egész Föld felszínét ábrázolni képesek *nincsenek*.	

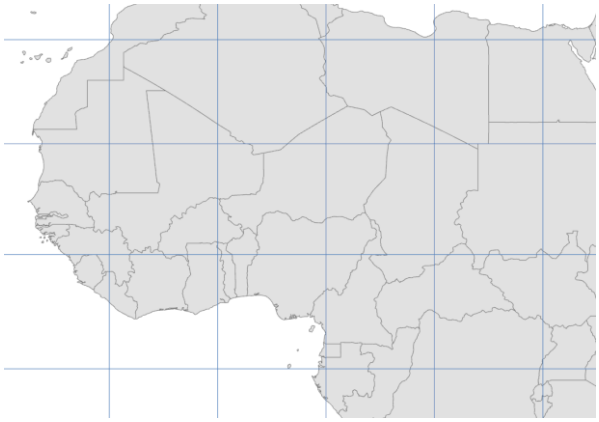
6.2.2 Kvázi-kontinentális, Geo-kartográfiai vetületek

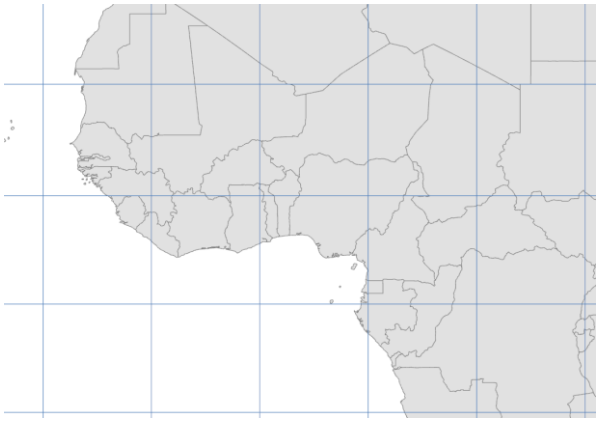
A kvázi-kontinentális vetületekhez olyan megoldások tartoznak, melyek egy egész kontinenst, vagy egy nagyobb kontinensrészt írnak le. Torzításuktól függetlenül két csoportra oszthatók (8. táblázat):

1. Alacsony földrajzi szélességen alkalmazható vetületekre,

2. Közepes- és magas földrajzi szélességeken alkalmazható vetületekre.

8. táblázat. Kvázi-kontinentális, Geo-kartográfiai vetületek.

Területtartó kvázi-kontinentális vetületek	Albers–féle területtartó kúpvetület:  Az Albers–féle területtartó kúpvetület nagyszerű választás a közepes- és magas földrajzi szélességek ábrázolására. Nem alkalmas Kelet–Nyugati irányban nagy kiterjedésű terület (pl. Európa és a teljes Oroszország) ábrázolására, mert a vetület megszakításos a kezdőmeridiánnal átellenben. Ilyen területek esetében ferdetengelyű Lambert–féle területtartó vetületet kell alkalmazni. Ha magasabb földrajzi szélességen a kezdőmeridántól a kezdőmeriadiánaig a teljes területet ábrázolni akarjuk, akkor poláris, azimutális Lambert–féle területtartó vetülettel érdemes dolgoznunk. Területtartó hengervetület: 

	Az alacsonyabb földrajzi szélességekre nagyszerű választás a területtartó hengervetület. Mivel az alacsony földrajzi szélességeken hengervetületként torzítása alacsony, hálózati beosztása négyzetesnek tűnik, de magasabb szélességek felé haladva látható, hogy a szélességek osztásköze változó. Az alacsony földrajzi szélességeken szöghibája alacsony, de nem szögtartó.
Szögtartó kvázi-kontinentális vetületek	Mercator–féle hangervetület:  A Mercator vetület az alacsonyab földrajzi szélességek klasszikus szögtartó vetülete. Jelentősen megkönnyíti a navigációt, mert a térképen a helyes szögek mellett az irányszögek is könnyen leolvashatók. Klasszikus, kedvelt vetület, mert a tengeren hajózva, ha két pontot egy egyenessel összekötünk, akkor az egyenes irányszögét tartva ténylegesen a célpontoz jutunk. (Igaz, ez nem a legrövidebb távolság.) Ezen Mercator–féle hengervetület egyszerűsített, nem szögtartó változata a Web Mercator (WGS84 Web Mercator; az okokat lásd alább: 6.3.2 WGS84 Web Mercator). Lambert–féle szögtartó kúpvetület: 

	A Lambert–féle szögtartó kúpvetület a magasabb földrajzi szélességek szögtartó vetületeként használhatjuk, ha a térkép kiterjedése Nyugat–Keleti irányban korlátozott (kisebb, mint fél Föld-kerület). A poláris térségek szögtartó ábrázolására használjunk poláris sztereografikus síkvetületet.
Távolságtartó (ekvidisztáns) kvázi-kontinentális vetületek	Magasabb szélességi körökön megegyeznek a világvetületeknél bemutatott „Ferdetengelyű, azimutális ekvidisztánsz vetület"-tel, mely egy ponról kiindulva őrzi meg a helyes távolságokat.

Abban az esetben, ha a vizsgálati területünk kiterjedése pontosan ismert, valamint az elvárás, hogy milyen torzítást kell eliminálnunk, akkor használhatunk vetületi tanácsadót is, amely megoldás több térinformatikai szoftverbe integrált, de on-line is elérhető, pl.: https://projectionwizard.org/. Az ilyen felületeken az adott területre maximálisan illeszkedő, valamilyen kiválasztott elvárás szerint (pl.: szögtartóság) vetületet szerkeszthetünk.

6.2.3 Nemzeti- és kisebb területek ábrázolása

A nemzeti- és annál kisebb kiterjedésű területek ábrázolása során a fentebb felsorolt mintáknál összetetteb a helyzet és ennek megfelelően a vetületválasztás is több utána járást igényel:

1. Ha egy konkrét nemzet területét szeretnénk térképezni, az országok rendelkeznek saját, hivatalos vetületi rendszerrel, általában szögtartóval, de a területtartó változatok is előfordulnak. Ország szerinti keresést a https://epsg.org/ honlapon végezhetünk.
2. Szükség esetén szelvényezett vetületekkel is dolgozhatunk, ilyen például az UTM – Universal Transverse Mercator. Ebből Magyarország a WGS 1984 UTM Zone 33N (EPSG: 32633) és a WGS 1984 UTM Zone 34N (EPSG: 32634) nevű pásztákra esik.

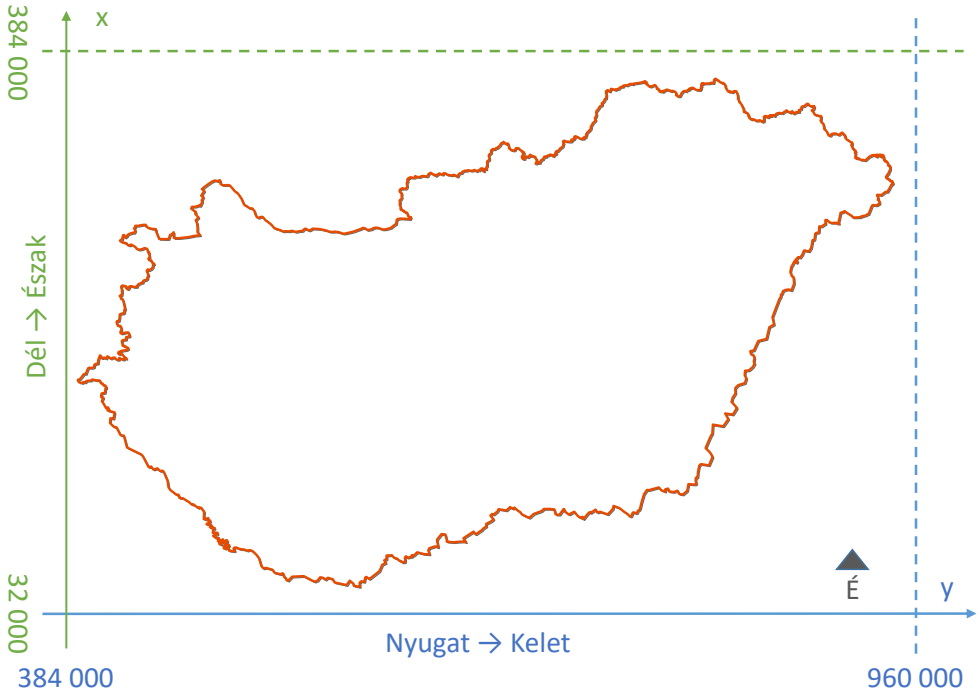
6.3 Magyarországon gyakran használt vetületi rendszerek

6.3.1 HD1972 Egységes Országos Vetület (EOV)

Magyarországon 1975 óta a polgári térképészet – törvényi előírás mellett – széles körben használja az EOV-t, az Egységes Országos Vetületet. Az EOV érvényességi területe Magyarország.

Az EOV *szögtartó, süllyesztett hengervetület*. Az EOV geodéziai dátuma a HD72, alapfelülete a dátumban használt ellipszoid, az IUGG67 (IUGG GRS 1967; a = 6 378 160 m, $\frac{1}{f}$ =298,247167427).

Az EOV az objektumok helyzetét a földrajzi térben két koordinátával írja le: '*x*' és '*y*'-al, amelyből az '*x*' a Délről ↑ Északra mutató tengelyt, míg az '*y*' a Nyugatról → Keletre mutató tengelyt. Az EOV koordinátáit méterben mérjük. Az EOV mindkét koordináta értéke minden esetben pozitív szám. Az '*y*' (Nyugat → Kelet) értéke mindig nagyobb, mint 384 000 m és mindig kisebb, mint 960 000 m és Nyugatról Kelet felé növekszik. Az '*x*' (Dél ↑ Észak) értéke mindig nagyobb, mint 32 000 m és mindig kisebb, mint 384 000 m (12. ábra). Ebből következik, hogy az EOV koordinátáit sosem lehet összetéveszteni, mert a fentebbi szabály szerint mindig kideríthető, hogy melyik értéket nézzük – még akkor is, ha az '*x*' és '*y*' megjelölések hiányoznak.



12. ábra. Az EOV tengelyei és a koordináta értékek értelmezési tartománya (y:] 384 000 m; 960 000 m [∈ $\mathbb{R}^+$; x:] 32 000 m; 384 000 m [∈ $\mathbb{R}^+$ *MÉM Országos Földügyi és Térképészeti Hivatal, [1975]).*

Az EOV kettős vetítésű rendszer:

1. Első lépésben, az IUGG67-ről szögtartó vetítéssel a pontokat egy szögtartó Gauss–gömbre vetítjük az alábbi egyenletekkel:

$$\varphi = 2 \cdot \tan^{-1}\left[k \cdot \tan^{n}\left(\frac{\pi}{4} + \frac{\phi}{2}\right) \cdot \left(\frac{1 - \varepsilon \sin\phi}{1 + \varepsilon \sin\phi}\right)^{\frac{n \cdot \varepsilon}{2}}\right] - \frac{\pi}{2},$$

$$\lambda = n \cdot \Delta\Lambda,$$

$$\Delta\Lambda = \Lambda - \Lambda_0,$$

ahol a 'φ' és a 'λ' a Gauss–gömbön értelmezett földrajzi szélesség és -hosszúság, a 'ϕ' és a 'Λ' az IUGG67-es ellipszoidon értelmezett szélesség és hosszúság; 'Λ_0' a vetületi kezdőpont IUGG67 ellipszoidi hosszúsága, melynek értéke az EOV Gauss–gömbjének esetében:

19°2′54,8584″ (~19,0485717777778°); ‘ε’ az IUGG67 ellipszoid első numerikus excentricitása (ε = 0,0818205679407); az ‘n’ értéke 1,000719704936; a ‘k’ értéke 1,003110007693.

2. A második lépésben az így kapott, Gauss–gömbön értelmezett ‘φ’ és a ‘λ’ értékeket hengerre vetítjük:

$$x = x_0 + \frac{m_0 \cdot R}{2} \cdot \ln\left[\frac{1 + \sin\varphi \cdot \cos\varphi_k - \cos\varphi \cdot \sin\varphi_k \cdot \cos(\lambda - \lambda_k)}{1 - \sin\varphi \cdot \cos\varphi_k + \cos\varphi \cdot \sin\varphi_k \cdot \cos(\lambda - \lambda_k)}\right],$$

$$y = y_0 + m_0 \cdot R \cdot \tan^{-1}\left[\frac{\sin(\lambda - \lambda_k)}{\tan\varphi \cdot \sin\varphi_k + \cos\varphi_k \cdot \cos(\lambda - \lambda_k)}\right],$$

ahol a ‘φ’ és a ‘λ’ az előző egyenletpárból kapott Gauss–gömbön értelmezett földrajzi szélesség és -hosszúság;

Ha rápillantunk a koordináta-rendszer tengelyeinek megnevezésére a 12. ábrán, akkor láthatjuk, hogy az ‘x’ és az ‘y’ elhelyezkedése szokatlan – mintha helyet cseréltek volna. Ennek a helycserének navigáció- és matematikatörténeti okai vannak: a szélesség maghatározása, különösen az északi félgömbön technikailag könnyű, elég csak a sarkcsillag horizont feletti magasságát lemérni. Ezért a Délről ↑ Északra mutató tengely könnyen előállítható, egzakt adottság, így matematikai értelemben független változónak tekinthető. A Nyugat → Kelet pozíció meghatározása nehezebb, mert egy egyezményes, de a felszínen nem azonosítható, koordináta-geometriai eszközökkel *választott* vonaltól kell mérni, összetett-, például időmérési eszközökkel, így ez a változó matematikailag függő változó. A matematikában a rendszerint a független változót az ‘x’ tengelyen, a függőt pedig az ‘y’ tengelyen vezetjük, gondoljunk például a függvények ‘ƒ(x)’ jelű definíciójára, ahol az y értékét adja meg az aktuális x érték függvényében. Így az EOV Délről ↑ Északra mutató tengelye a független, ‘x’, a Nyugat → Keleti tengely pedig a függő ‘y’.

Ebből a történeti elkülönítésből rengeteg probléma adódik, mivel a térinformatikai szoftverek a nagyszámú támogatott vetületi rendszer miatt nem lehetnek tekintettel a tengelycserére ezért a térinformatikai adatbevitel során az EOV X értékét kell megadni (térinformatikai-) Y-nak és az EOV Y értékét X-nek!

A helyzetet tovább bonyolítja, hogy a legtöbb térinformatikai rendszer vetületfeldolgozó motorjában az EOV egyenleteit nem implementálták, hanem helyettük egy rokon eljárással, a Hotine–féle ferdetengelyű Mercator-vetülettel közelítik őket. Így ugyan az eredmény nem tökéletes, de a keletkező hiba szubmilliméteres, így a gyakorlatban teljesen elhanyagolható.

Az EOV azonosítószáma 23700 (EPSG).

Ha WGS84 ellipszoidi koordináta-rendszerből szeretnénk eljutni EOV-ba, akkor az eltérő geodéziai dátum miatt az áttérés minden esetben transzformációt igényel (lásd: Magyarországon használt dátumok és dátumtranszformációik).

6.3.2 WGS84 Web Mercator

A Web Mercator egy globális vonatkoztatási rendszer, melynek alapját képező geodéziai dátum a WGS84. Bár nevében ott a „Mercator" megnevezés, amely szögtartó vetületre enged következtetni, a Web Mercator nem szögtartó (sem nem területtartó), mivel vetületi egyenletei gömb alakú Földre vonatkoznak, miközben ellipszoidális alapfelületről származnak a szélesség- és hosszúság értékei:

$$y = \mathrm{R} \cdot \ln\left[\tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right)\right],$$
$$x = \mathrm{R} \cdot \Lambda$$

ahol a 'ϕ' és a 'Λ' a WGS84 ellipszoidi szélesség és -hosszúság radiánban kifejezve; 'R' a WGS84 ellipszoid fél nagytengelyének hossza méterben kifejezve (6 378 137 m). A Web Mercator vetületi egyenletei megegyeznek a közönséges gömbi alapfelületet használó Mercator vetület egyenleteivel.

Az 'y' komponens tangense miatt a Web Mercator nem alkalmas a sarkpontok ábrázolására, mert azokat vonalként elnyújtottan, a végtelenben jelenítené meg, ezért a legtöbb térinformatikai rendszer $\phi \approx \pm 85°$ környékétől északra és délre elveti a bejövő koordinátákat, így nem jeleníti meg őket a térképen.

A Web Mercator nagy előnye, hogy vetületi egyenletei igen egyszerűek, informatikai környezetben jól implementálhatók, de hátránya, hogy ellipszoidi koordinátákat használ (helytelenül) gömbi formulákkal ezért nem szögtartó, így a térképen két pontot összekötő egyenes nem tekinthető loxodrómának sem.

Bár a Web Mercator globális vonatkoztatási rendszer, magyarországi használata elkerülhetetlen, mert az online térképek (Google; ESRI; Microsoft; HERE; OpenStreetMap; stb.) ezen vetületi rendszerben tárolják koordinátáikat.

És miért nem szögtartó a Web Mercator? Önmagában a Mercator vetület szögtartó, a problémát az okozza, hogy a Mercator vetületnek van ellipszoidi- és gömbi koordinátákra vonatkozó változata is, de a Web Mercator esetében ellipszoidi koordinátákat táplálunk be egy gömbi koordináták vetítésére tervezett egyenletbe: ez egy szándékosan elkövetett hiba, mellyel az informatikai

számítási igény csökkenthető, de a szögtartóság odavész. A Web Mercator szögtartó, azonos elhelyezésű, ellipszoidi koordinátákat, ellipszoidra tervezett egyenletekkel vetítő változata a WGS84 World Mercator (EPSG: 3359).

6.4 A vetületek használatának veszélyei a térinformatikában

A vetületek Descartes–féle, derékszögű koordináta-rendszerek, melyekben az Euklideszi geometria szabályai érvényesek, így könnyen számíthatunk távolságokat és területeket – melyek rendszerint nem valósak és csak hibákat és félreértéseket szülnek. Nézzük meg a gyakoribb hibákat:

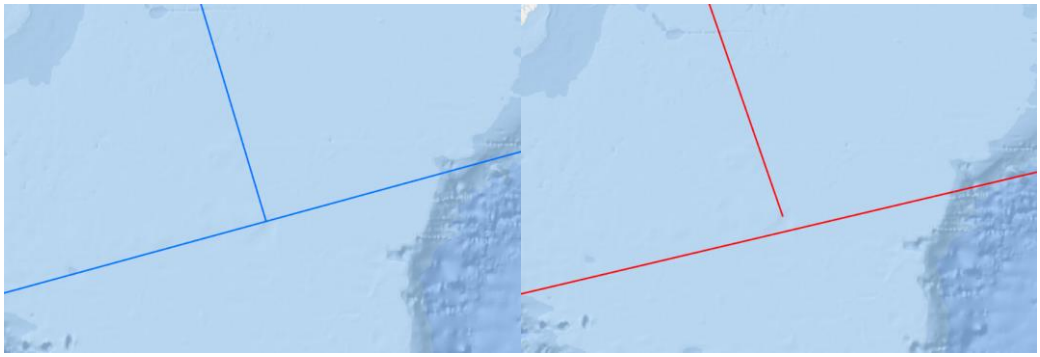
- Első körben vegyünk egy nagyobb kiterjedésű multipoligont, például Kalifornia államot, mely multipoligonban az egyes csomópontok WGS84 Web Mercator vonatkoztatási rendszerven vannak. Próbáljuk meg kiszámolni a rendelkezésre álló adatok alapján Kalifornia népsűrűségét! Ehhez az adott elemen ott szerepel az állam 2022-es lakosságszáma: 39 770 476 fő; viszont nincs ott az állam területe négyzetkilométerben, amely a számításhoz kellene. Semmi probléma, a geoadatbázisok rendszerint nyilvántartják az adott poligonnak az aktuális vetületi rendszerben értelmezett területét és ezt egy származtatott attribútum adatban fel is tűntetik: ez az érték itt 649 179 579 516,077148 m^2, tehát kb. 649 179,58 km^2.
  Ha elosztanánk a két számot máris kapnánk egy eredményt, amely azonban hibás lenne, mert a számítás alapját jelentő vetületi rendszer nem területtartó; egy eleve torzított poligonnak kaptuk csak meg a területét. Mekkora az állam valódi területe? A valódi terület kb. 409 313,82 km^2. Tehát a valódi terület az eredetileg kapott, hamis számértéknek mindössze a ~63%-a. Ha a rossz értékből számoljuk az állam népsűrűségét, egy pillanat alatt majdnem felezhetjük a népsűrűséget (amely a valóságban 97,2 fő volt négyzetkilométerenként 2022-ben, amely összevethető Magyarország népsűrűségével).
- Vegyünk egy egyszerű vonalat, ugyanitt Kaliforniában a Santa Monica Pier és a San Ysidro–Tijuana-i határátkelő között. A vonalunknak két vertexe van, a kezdő- és a végpontja. Mindkét pont vonatkoztatási rendszere WGS84 Web Mercator. A kérdés igen egyszerű: milyen távol van a két pont egymástól légvonalban, síkban? A probléma az előzőhöz igen hasonló. Csábítást érzünk, hogy bevessük a koordináta-geometriát, és egy egyszerű $\Delta x^2+\Delta y^2$ képlet segítségével megkapjuk a távolságot. A geoadatbázisok is ugyanezt teszik: a származtatott attribútum adatban olvasható, hogy a távolság 254 669,56 m. Csakhogy a két pont a vetületi torzítás miatt eleve torzult pozícióval került térképre, így megmértük a torz távolságot. De mennyi a valós? Ehhez a WGS84 Web Mercator koordi-

nátákat vissza kell váltanunk WGS84 ellipszoidi szélességre és -hosszúságra és az ellipszoid görbületét figyelembe vevő eljárással (pl. Vincenty–, vagy Karney–féle algoritmussal) kell meghatároznunk a távolságot. A helyes távolság 212 532,07 m, így kb. 42 kilométerrel kevesebb, mint az első számérték esetében.

- Kössünk össze két nagyon távoli pontot, például Pécset Los Angeles-szel! A térképünk vonatkozgatási rendszere legyen a WGS World Mercator (nem Web, hanem „valódi" ellipszoidi koordinátákkal számoló szögtartó Mercator)! A feladat ugyanaz: milyen messze van a két település – *légvonalban* mérve? Vegyük először a naiv megközelítést. Digitalizáljunk egy vonalat egyenesen a térképen a két település között, úgy, hogy a vonal csak kettő (kezdő- és vég-) pontból álljon! Olvassuk le az euklideszi távolságot (amiről az előző példa alapján tudjuk, hogy rossz). Az eredmény 15 295,5 km. Tudjuk, hogy rossz, mert vetületi torzítást szenvedett, ezért nem törődünk vele és kiszámoljuk a vonal geodéziai hosszát. Ennek az eredménye 11 660,3 km. Az eredmény nem meglepő, nem ért minket váratlanul. De valóban ilyen messze van a két település *légvonalban*?
  Nem, egyáltalán nem, mert a Mercator vetületeken két pontot összekötő egyenes vonal *nem* a két pont közötti legrövidebb távolságot jelöli ki, hanem a loxodrómát, amely két pontközötti olyan útvonal, hogy közben az északi iránytól mért relatív irányszögünk változatlan marad.
  A loxodróma jelentése „hosszú futás". Ebből már sejthetjük, hogy nem a számítással, hanem az *eljárással* van a gond. Kössük össze a két pontot iteratív vonallánccal (rengeteg töréspontal), úgy, hogy tényleg légvonalban fusson és számítsuk ki ennek a vonalnak a hosszát! Az eredmény 10 107,6 km. A különbség rettenetes, pedig a feladat nagyon egyszerűnek tűnik.
- Ezzel rokon probléma a hálózatok szétesése, a topológiai degradálódása. Képzeljük el, hogy van egy 400 kV elektromos távvezetékünk, amely 600 km hosszan halad. Az ilyen gerinchálózatokról természetesen lesznek leágazások, akár azonos, de főleg alacsonyabb feszültség-szinteken. Az így en 400 kV-os vezetékek viszonylag keveset kanyarognak, hosszan futnak egyenesen, ezért kevés vertex-szel is ábrázolhatók, így előfordulhat, hogy egy leágazó szakasz úgy csatlakozik a gerinchez, hogy ott a gerincen nincsen töréspont. Ez nem probléma, mert geometriailag a leágazó kezdő töréspontja pontosan az élre (két töréspont közé) esik a gerincen.
  Ekkor azonban, például egy térképészeti szabvány miatt át kell térnünk egy másik vetületi rendszerbe. Ekkor a vetületi motor rendszerint csak a töréspontokat projektálja. Innen már érezhető a probléma: két távoli pont között az egyenes továbbra is egyenes marad – az aktuális vetületi rendszerben. Ez nem jeleni azt, hogy a vetületi torzítás miatt, a leágazó szakasz kezdőpontja továbbra is a gerincvezetéken marad, hanem egyszerűen

elszakad tőle. Ezzel tönkre is ment a hálózat topológiája.

Ez csak úgy kerülhető el, ha a topológiailag fontos pontokon, az összes elemen van (azonos helyen lévő) vertex. A jelenség nem csak vezetékeknél, hanem poligonoknál, azok illeszkedésénél is felléphet (13. ábra):



13. ábra. Optikai kábelek „őrvertex" nélkül, vetületi transzformáció előtt (bal) és vetületi transzformáció után (jobb oldal). A vezeték nyomvonala fiktív.

Itt egy tengeralatti optikai kábelt látunk Új-Fundland közelében. A bal oldali az eredeti, Web Mercator vetületben készült térkép. A találkozási pontban nincs vertex. A jobb oldali ugyanezen vezetéket ábrázolja Miller–féle általános torzítású hengervetületen. A megszakadás a topológiai őr-töréspont hiányában tisztán látható. (A leágazó vonal kb. 17 km-re távolodott el, az 1:2 500 000 méretarányú térképen.)

7 Vonatkoztatási rendszerek a térinformatikában

Az előző két fejezetben leírtak („A geoinformatika felsőgeodéziai alapjai” és „Vetületek a térinformatikában”) egy elvi keretet jelentenek, melyben a térbeli számítások elvégezhetők és az adatok tárolhatók. Ezen elvi elemeket vonatkoztatási rendszerekké szervezzük össze, hogy konkrét georeferenciát nyújthassunk téradatainknak. Ezen vonatkoztatási rendszerek kettő plusz egy nagycsaládra oszlanak: földrajzi- és vetített vonatkoztatási rendszerekre, valamint függőleges vonatkoztatási rendszerekre. Ezen vonatkoztatási rendszerek a térinformatikában a fent leírt elméleti háttér és a tényleges konstansok és -egyenletek gyakorlati összesége.

7.1 Földrajzi vonatkoztatási rendszerek

A földrajzi vonatkoztatási rendszerek a földrajzi- és ellipszoidi koordináta rendszerek gyakorlati megvalósításai, melyek az alábbi elemekből épülnek fel:

- **Szferoid**. A szferoid lehet gömb, vagy ellipszoid, melynek eltároljuk a nevét, a fél nagytengelyének hosszát és a lapultságát, vagy annak reciprokát. Ilyen szferoid például a GRS 1980, amely egy forgási ellipszoid, melynek fél nagytengelye 6 378 137 m, lapultságának reciproka 298,257222101. A szferoid önmagában csak a vonatkoztatási felület méretét és alakját írja le. A szferoid önmagában nem tér ki annak térbeli helyzetére sem annak más tulajdonságaira.
- **Szögegység**. Bár megszokásból rendszerint fokot használunk, lehetséges más szögegységek használata is, például a radiáné, vagy a gradiáné is. Abban az esetben, ha vonatkoztatási rendszer számértékeit nem emberi használatra tervezték, akkor a radián optimális választás a számítások során használt szögfüggvények miatt.
- **Kezdőmeridián**. A kezdőmeridián vagy nullmeridián határozza meg, hogy hova tűzzük le a Föld felszínén a 0 kezdőpontot a hosszúságértékek esetében. Például a WGS ’84 esetében ez az International Earth Rotation and Reference Systems Service (IERS) IERS Reference Meridian (IRM) melyet az egyszerűség kedvéért a legtöbb térinformatikai rendszer Greenwich-i Nullmeridiánnak nevezi. Történeti szemszögből számos meridián létezett (Ferro, Párizs, Lisszabon, Oszló, Jakarta stb.).
- **Geodéziai dátum**. A geodéziai dátum (A geodéziai dátum) feltűntetése itt némiképp’ kakukktojás, mert az elvi geodéziai dátum magába foglalja a fentebb leírtakat is, de a térinformatikai megvalósításban ez csak azonosító, amely alapján a dátumtranszformációk konstansai kikereshetők.

- **Leíró- és kiegészítő adatok**. A leíró adatok rendszerről rendszerre változnak, de rendszerint itt jelölik meg a vonatkoztatási rendszer azonosítószámát (pl.: EPSG), az érvényességi területet (pl.: Ausztrália) és a várható terepi pontosságot. A leíró- és kiegészítő adatok feltüntetése nem kötelező.

Nézzünk erre egy gyakorlati példát! A WGS 1984 (EPSG: 4326) vonatkoztatási rendszer szferoidja egy ellipszoid, melynek neve szintén WGS 1984, fél nagytengelyének hossza 6 378 137 m, lapultságának reciproka 298,257223563. Geodéziai dátuma a szintén azonos nevű WGS 1984. Egysége a fok. Nullmeridiánja az IERS Reference Meridian (IRM). Leíró adatai szerint az egész Bolygón használható.

A vonatkoztatási rendszerek térinformatikai lejegyzésére számos formátum létezik. Ezek közül az egyik leggyakoribb az ESRI-stílusú WKT (Well-Known Text):

```
GEOGCS["GCS_WGS_1984", DATUM["D_WGS_1984",
SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Greenwich",0.0], UNIT["Degree",0.0174532925199433]]
```

Ebben a WKT-ban minden fontos paraméter megtalálható: a nullmeridián, a szferoid, a geodéziai dátum és a szögegység egyaránt.

Ugyanezen vonatkoztatási rendszernek az OGC (Open Geospatial Constortium) stílusú WKT-ja:

```
GEOGCS["WGS84",DATUM["WGS_1984",SPHEROID["WGS84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]]
```

A HD72 (Hungarian Datum 1972) nevű földrajzi vonatkoztatási rendszer egysége a fok, nullmeridiánja a Greenwich-i, geodéziai dátuma a Hungarian 1972, szferoidja a GRS 1967. A HD72 ESRI WKT-ja:

```
GEOGCS["GCS_Hungarian_1972",DATUM["D_Hungarian_1972",SPHEROID["GRS_1967",6378160.0,298.247167427]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]]
```

7.2 Vetített vonatkoztatási rendszerek

A vetített, vagy projektált vonatkoztatási rendszerek a földrajzi vonatkoztatási rendszerekre épülnek és azok valamilyen vetítési eljárással létrejött képét jelentik.

A vetített vonatkoztatási rendszerek az alábbi elemekből állnak:

- **Földrajzi vonatkoztatási rendszer**.
- **Hosszegység**. A hosszegység rendszerint méter, de különösen angolszász területeken más egységek is előfordulnak (nemzetközi láb, öl, lánc, indiai láb stb.).

- **Vetítési eljárás**. A vetületi eljárás keretében definiáljuk a vetületi egyenletet és az ahhoz tartozó konstansokat (pl.: kezdőmeridián-, eltolás-, elforgatás értékei).
- **Leíró adatok**. A földrajzi vonatkoztatási rendszerekhez hasonlóan itt is befoglaló téglalapokat találhatunk a használati terület körül.

Nézzünk néhány gyakorlati példát a vetített vonatkoztatási rendszerekre is! A WGS 84 Web Mercator földrajzi vonatkoztatási rendszere a WGS 1984 (EPSG: 4326), hosszegysége a méter, vetítési eljárása a „Mercator Auxiliary Sphere".

Egy vetítési eljárás *számos vetített vonatkoztatási rendszerben előfordulhat*, más földrajzi vonatkoztatási rendszerrel, más hosszegységgel, de akár már vetítési konstansokkal is.

A térinformatikai rendszerekben az EOV kettős vetítése nem implementált, a Hotine–féle ferdetengelyű Mercator-vetülettel közelítik. Az EOV (EPSG: 27300) földrajzi vonatkoztatási rendszere a HD72 (Hungarian Datum 1972), egysége a méter. Az EOV ESRI WKT-ja:

```
PROJCS["Hungarian_1972_Egyseges_Orszagos_Vetuleti",GEOGCS["GCS_Hungarian_1972",DATUM["D_Hungarian_1972",SPHEROID["GRS_1967",6378160.0,298.247167427]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Hotine_Oblique_Mercator_Azimuth_Center"],PARAMETER["False_Easting",650000.0],PARAMETER["False_Northing",200000.0],PARAMETER["Scale_Factor",0.99993],PARAMETER["Azimuth",90.0],PARAMETER["Longitude_Of_Center",19.0485717777778],PARAMETER["Latitude_Of_Center",47.1443937222222],UNIT["Meter",1.0]]
```

A WKT-ban megfigyelhetjük a földrajzi vonatkoztatási rendszert („GEOGCS") mely a vetítés alapfelületét képezi, a vetületi eljárást („Hotine_Oblique_Mercator_Azimuth_Center") és az ehhez tartozó vetületi paramétereket is.

7.3 Függőleges vonatkoztatási rendszerek

A függőleges vonatkoztatási rendszerek a térinformatikai adatok függőleges értékeit azonosítják. Csak 3D adatok esetében használtak. Két fő családra oszlanak:

1. **Szferoid alapú rendszerek**. A szferoid alapú függőleges vonatkoztatási rendszerek egy adott szferoid felett értelmezik a pont magasságát. Adattartalmukat tekintve megegyeznek a földrajzi vonatkoztatási rendszerekkel kiegészítve egy pozitív irány és egy magasság-egység adattal.
2. **Gravimetrikus rendszerek**. A gravimetrikus rendszerek olyan függőleges vonatkoztatási rendszereket írnak le, amelyek tengerszinthez, vagy geoidhoz kötöttek. Ilyen rendszer például az EOMA is. A pozitív irány és a magasság-egység mellett a tengerszint megállapításának forrását vagy a geoidot nevezik meg.

8 Műveletek raszterekkel

8.1 Raszterek megjelenítése

8.1.1 Egysávos raszterek megjelenítése

A térinformatika eszközkészletének fontos része a raszterek elemzése. A raszterek matematikai értelemben mátrixok, melyekben az egyes pixelek (négyzetes, vagy téglalap alakú képelemek) sorokba- és oszlopokba szerveződnek, rés- és átfedés mentesen lefedik a vizsgálni kíván területet.

Az egysávos raszterek csak egy „réteget”, egy tematikus elemet tartalmaznak, például csak olyan pixelek vannak rajtuk, melyek például a domborzatot, vagy a talajtípust írják le. A továbbiakban az egysávos raszterek megjelenítését a domborzatmodellek példáján mutatjuk be, de az itt megadott eljárások általánosíthatók más, egysávos raszterekre is.

8.1.2 Domborzatmodellek folytonos színskálán történő megjelenítése

A domborzatmodellek olyan egyszerű, egysávos raszterek, melyek egyetlen sávjuk pixelein a pixel által lefedett térrész felszíndomborzatának abszolút magasságát írják le, numerikus, bináris formában, egész- vagy lebegőpontos számokként. Ezzel a domborzatmodellekre olyan táblázatként is gondolhatunk, ahol a magassági értékek egyszerűen sorokba és oszlopokba szervezettek.

Vegyünk egy kísérleti rasztert valahol Somogy-megyében! A raszter sorainak száma 127, az oszlopok száma 226, a cellák négyzetesek, 10×10 méteresek, így a teljes raszter térbeli kiterjedése 2 260 m × 1 270 m. Ebből fakadóan 1:15 000-es méretarányban ábrázolva, a kiterjedése ~15 cm × ~8,47 cm. A raszter adattípusa 32-bites, lebegőpontos szám. Horizontális vonatkoztatási rendszere EOV; vertikális vonatkozgatási rendszere EOMA. Tömörítettlen mérete ~112,12 KiB. A raszter minimumértéke 148,131 m, maximuma 237,422 m.

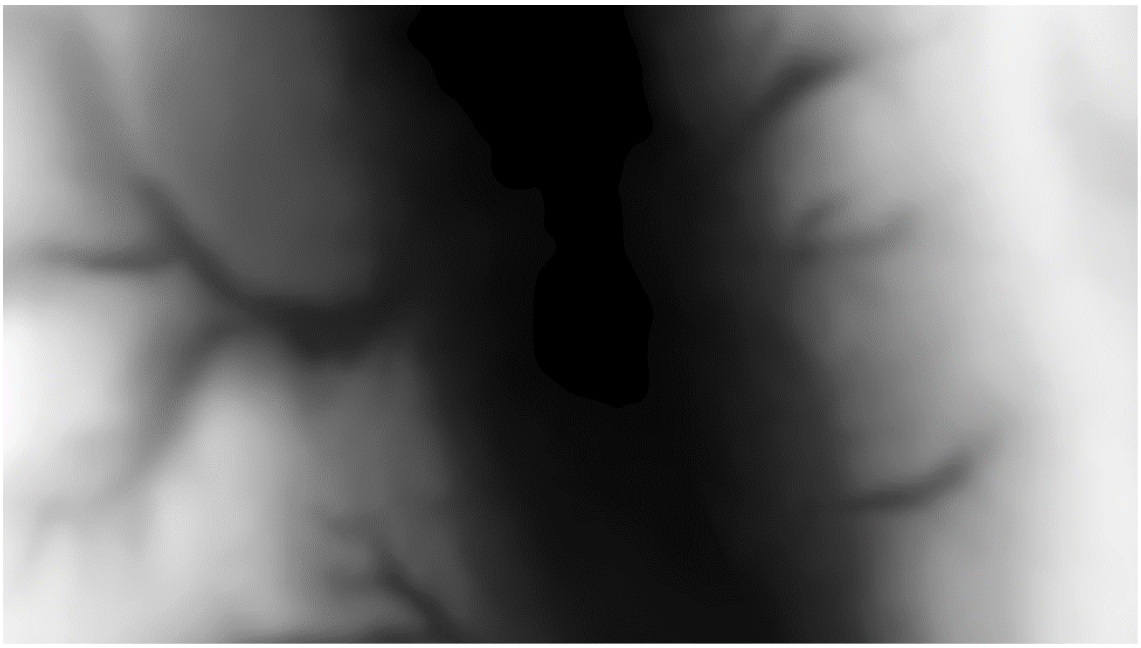
Mostanra elég sokat tudunk a raszterről, de még mindig nem láttuk. Hogyan lehetne megjeleníteni? A monitor három színsáv megjelenítésére képes, a vörösre, a zöldre és a kékre, úgy, hogy minden egyes sávban meg kell az adott monitor-pixelnek mondanunk, hogy mennyit bocsájtson ki az adott színből. Nullát kell küldenünk, ha az adott színt az adott pontban egyáltalán nem szeretnénk látni és 255-öt, ha az adott pontban maximális szín intenzitást szeretnénk az adott színből. Technikailag pixelenként három byte-ot, összesen 24 bitet kell küldenünk, ahol az első byte leírja a vörös, a második byte a zöld, a harmadik byte pedig a kék mennyiséget.

Ha visszapillantunk a raszter összefoglaló technikai adataira, akkor láthatjuk, hogy ez a raszter nem alkalmas közvetlen megjelenítésre: mindössze egyetlenegy sávot tartalmaz és az az egy sáv is harminckétbites, lebegőpontos szám, ahelyett, hogy három nyolcbites (vörös, zöld, kék) sávot tartalmazna, amit a monitornak el tudnánk küldeni megjelenítésre.

Ezért első lépésünk, hogy el kell határoznunk, hogy milyen módszerrel szeretnénk a raszterünket megjeleníteni.

Próbálkozzunk elsőre a legegyszerűbbel és ábrázoljuk úgy a terepmodellünket, hogy a legalacsonyabban lévő pontok legyenek teljesen feketék (R: 0; G: 0; B: 0), a legmagasabban fekvők fehérek (R: 255; G: 255; B: 255), a köztük lévő értékek pedig lineárisan felosztva jelenjenek meg a szűrke valamilyen árnyalataként (így például a közepes magasságú pontok legyenek közép-szűrkék [R: 127; G: 127; B: 127]), az alábbi színskála szerint:

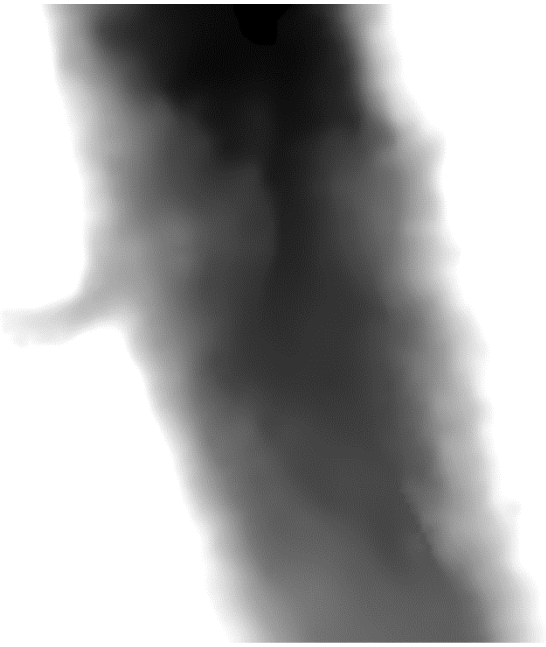
Így ahhoz, hogy ezt a skálát felhasználjuk az eredeti pixel értékeket lineárisan újra kell skáláznunk az alábbi képlet szerint: $R = G = B = \frac{255}{Max-Min} \cdot (Px - Min)$, ahol R, G és B a három színsávban várt érték 'Px' az adott pixel értéke, a 'Min' és 'Max' az adott raszter minimum és maximum értéke. Ha ezt a műveletet minden egyes pixelre elvégezzük, majd az R, G, B értékeket pixelenként elküldjük a monitornak, akkor meg is jeleníthetjük a raszterünket (14. ábra):



14. ábra. A vizsgálati terület képe, minimum–maximum hozzárendelés szerint, lineáris módszert követve. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából. Vizualizáció: ArcGIS Pro 3.3.1.

Így a raszterek képernyőn történő megjelenítését egy hozzárendelési függvények is tekinthetjük, amely úgy működik, hogy a raszteren kapott értékekhez egy függvénykapcsolat alapján színeket rendel a monitoron, egy színskála szerint. Jelen esetben a hozzárendelés lineáris: a kapott R, G, B érték egyenesen aránylik a pixel értékével.

A minimum- és a maximum értéket a térinformatikai szoftverek automatikusan a megjelenítendő raszterből nyerik, azonban, ha mi specifikusan csak egy magasság- (érték-) tartományra vagyunk kíváncsiak akkor mi magunk is beállíthatjuk (15. ábra):

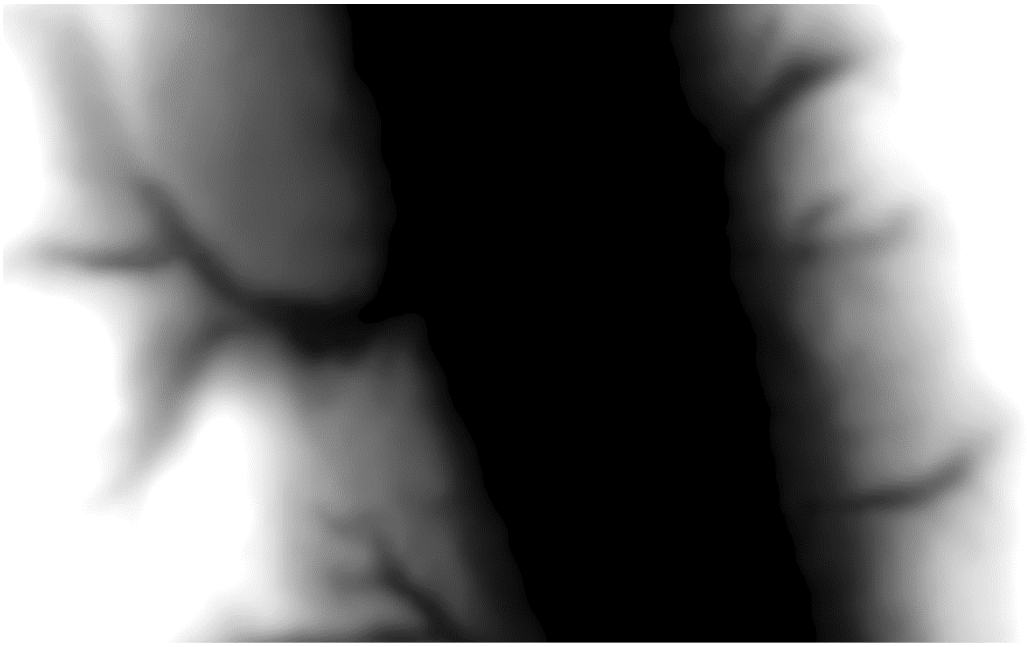


15. ábra. A vizsgálati terület képe minimum–maximum hozzárendelés szerint egy módosított, [148,13; 163,26] méteres intervallumon. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából. Vizualizáció: ArcGIS Pro 3.3.1.

Ebben a példában a minimumot az eredeti, 148,13 méteren hagytuk, a maximumot azonban 163,26 méterre szállítottuk le. Ekkor minden egyes pixel, melynek értéke nagyobb vagy egyenlő 163,26-tal automatikusan fehér lesz, a többire viszont a fentebb leírt lineáris hozzárendelés szabálya vonatkozik, így azt is mondhatjuk, hogy úgy jelenítettük meg a rasztert, hogy minden legyen fekete, ami kisebb vagy egyenlő 143,13-mal és minden legyen fehér, ami nagyobb vagy egyenlő 163,26. A köztes értékek pedig lineárisan vegyék fel a szűrke 255 árnyalatát.

De nézzük meg egy kicsit közelebbről is az eredményt! Ez a megoldás nem csak megjelenítési lehetőség, hanem egyben elemző eszköz is. A terepmodell alacsonyabban fekvő részein feltűntek olyan morfológiai formák, amelyek az eredeti (15. ábra) képen nem látszottak, mert mivel a minimum- és maximum értékek távol voltak egymástól, egyszerűen olyan hasonló értékeket kaptak, hogy beleolvadtak a környezetükbe.

A minimum–maximum megjelenítés a legtöbb térinformatikai szoftver alapértelmezett beállítása. Ez a megközelítés rendszerint jó eredményt ad, de ha a minimum- és/vagy a maximum érték extrém messze esik az adatsor átlagától, akkor a lényegi részek „elkenődnek" és elvesznek a homogén szürkeségben. Ezen probléma automatikus kezelésére született meg a szórás alapú megjelenítés. A szórás alapú megjelenítés első lépése, hogy végigszaladunk a raszter összes pixelén és meghatározzuk a pixelek értékének átlagát. Az átlag alapján kiszámítjuk a szórást, majd a minimumot az átlagtól balra egy szórásnyira, a maximumot az átlagtól jobbra, szintén egy szórásnyira határozzuk meg. Jelen esetben a raszterünk átlaga 186,2 m, a szórása 27,6, így a minimum 158,6 m, a maximum 213,8 m. Az így kapott raszter jól ábrázolja az átlag köré rendeződött értékekben rejlő geomorfológiai mintázatokat (16. ábra), miközben a valódi minimum és maximum közelében lévő értékek elfeketednek, vagy elfehérednek:



16. ábra. Szórás alapú megjelenítés. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából. Vizualizáció: ArcGIS Pro 3.3.1.

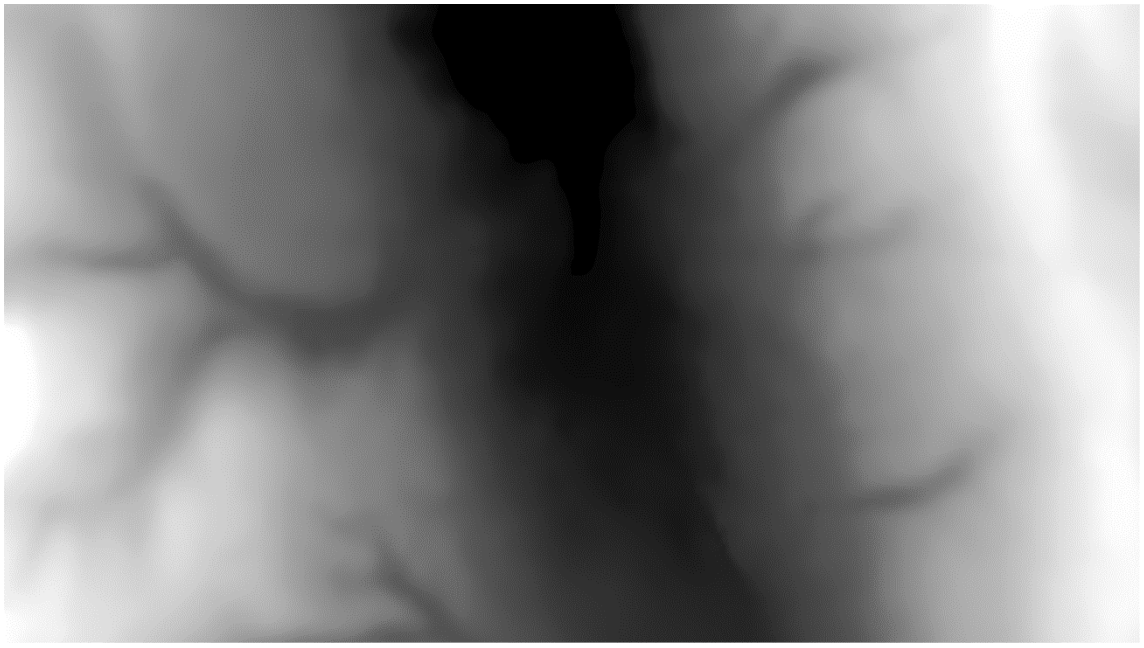
A szórás alapú megjelenítéssel rokon eszköznek tekinthető a legtöbb térinformatikai szoftverben elérhető hisztogram-vágó eszköz is: ekkor a minimumot és a maximumot úgy határozzák meg, hogy az adatsor alsó és felső néhány százalékát (rendszerint 2, vagy 5) levágják, majd az így kapott maradék adatsor minimum- és maximum értékét használják.

Az idáig áttekintett eszközök kivétel nélkül lineárisak voltak: két számérték között lineárisan rendeltek színértéket az egyes pixelekhez. Ez az elképzelés nagyon hatékony és a térkép olvasó számára intuitív is egyben: bármikor kitehetünk egy jelmagyarázatot, amelyben egy színskála szerepel és mellé írhatjuk, hogy a fekete ennyi métert, a fehér pedig annyi métert jelöl, és az ol-

vasó innen már rögtön tudni fogja, hogy a közepes szűrkék közepes értékeket takarnak a térképen. A megoldásnak azonban van hátránya is: mint ahogy az előző ábrákon láthattuk, az eredmény vagy általánosan kielégítő (14. ábra), vagy specifikusan fókuszál egy-egy jelenségre (pl. az alacsony térrészekre [15. ábra]), de a többi jelenséget elrejti.

A raszterek megjelenítésének azonban vagy egy *nem lineáris* megközelítése is. Ez az eszköz a hisztogram-kiegyenlítés, vagy más néven hisztogram-ekvalizáció. A művelet egy kissé komplex, de könnyedén lépésekre bontható:

1. Az első lépésben – az egyszerűség kedvéért – tekintsünk úgy a raszterünkre, hogy nem 32-bites lebegőpontos, hanem 32-bites egészszámokat tartalmaz. (Ez a konverzió gyorsan megtehető a valóságban is.)
2. Határozzuk meg az egyedi értékek számát és azt, hogy az egyes egyedi értékekből összesen hány darab van a raszter egészén elszórva! Iteratívan végezzük el a következő műveletet: $k(i) = \sum_{j=0}^{i} \frac{n_j}{n}$, ahol '$i$' az egyedi elemek számossága, n_j az adott érték számossága a raszter egészén, 'n' az összes pixel száma a raszteren (sorok- és oszlopok számának szorzata). Ezzel a művelettel minden egyes egyedi értékre kapunk egy kumulatív gyakoriságot (minden relatív gyakoriság értéket összeadunk az őt megelőző értékek relatív gyakoriságaival).
3. Végig megyünk a raszteren, és miden egyes pixelt, az adott pixel értékére jellemző kumulatív gyakorisággal helyettesítjük.
4. Az eredmény a $]0;1] \in \mathbb{R}$ intervallumba fog esni az összes az összes pixelen, amelyet már csak 255-tel kell megszoroznunk. Az eredményt egészszámmá konvertálva közvetlenül felhasználhatjuk a szürkeárnyalatos megjelenítésre (17. ábra):



17. ábra. Megjelenítés a hisztogram-ekvalizáció eszközével. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából. Vizualizáció: ArcGIS Pro 3.3.1.

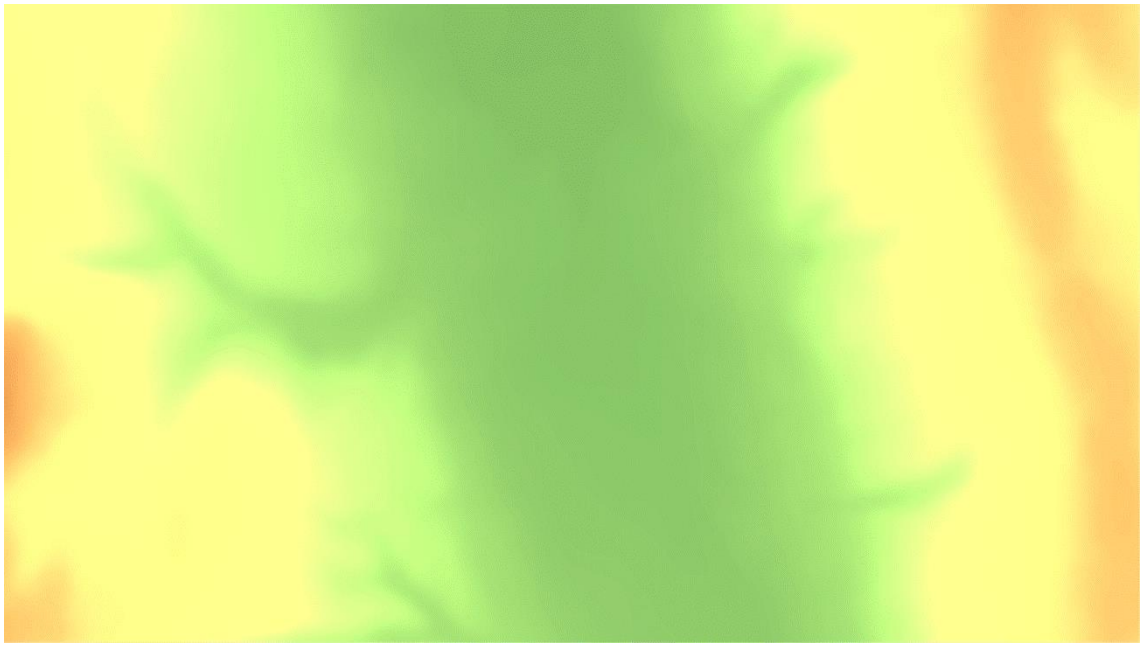
A hisztogram-ekvalizációt intuitívan úgy is felfoghatjuk, hogy a szűrkeskála nagyrésze abba az érték tartományba kerül, ahol sok adat van. Vegyünk egy példa rasztert! A minimum 0, a maximum 1 000, de az adatok 90%-a [50; 70]-es intervallumba esik, akkor hisztogram-ekvalizált megjelenítéssel a szűrkeskála 90%-a is erre a tartományra fog esni.

A hisztogram-ekvalizáció segítségével így hatékonyan meg lehet jeleníteni olyan adatsorokat, melyek részletei hagyományos, lineáris színezés esetében egyszerűen láthatatlanok lennének. Figyeljük meg, hogy a képen a völgy- és a magasabban fekvő térrészek formakincse is jól kivehető.

Az eljárás hátránya, hogy a térképolvasó számára a mellékelt színskála nem jelent többé mérhető összefüggést: a jelen ábrán is csak azt lehet megállapítani, hogy a világos részek magasak, a sötétek alacsonyak, de szemmel nem tehetünk többé olyan megállapításokat, hogy „ez a rész kétszer világosabb, ezért kétszer magasabb is”.

Soha ne alkalmazzunk hisztogram-ekvalizált megjelenítést olyan rasztereken, amelyeken színalapú mérési igény merül fel! Így például az előző, hisztogram-ekvalizált terepmodell alkalmas geomorfológiai elemzésre, de használhatatlan például túrázáskor, ahol a legyőzött szintkülönbséget előre, számszerűen, a szűrkeárnyalat alapján kell meghatározni.

Természetesen az ábrázolás során nem szükséges a szürkeárnyalatos megjelenítéshez ragaszkodnunk: a három színsávot tetszőleges kezdő-, vég- és köztes pontokkal megszerkeszthetjük. Ekkor a 0–255-ös tartományra normált egész számokat nem közvetlenül ábrázoljuk, hanem egy előre eltárolt színskálán megnézzük, hogy az adott értékhez milyen RGB-szín tartozik (18. ábra):



18. ábra. Színskála alapján színezett domborzatmodell, módosított minimum–maximum hozzárendelés szerint. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából. Vizualizáció: ArcGIS Pro 3.3.1.

Itt a színskála az alábbi:

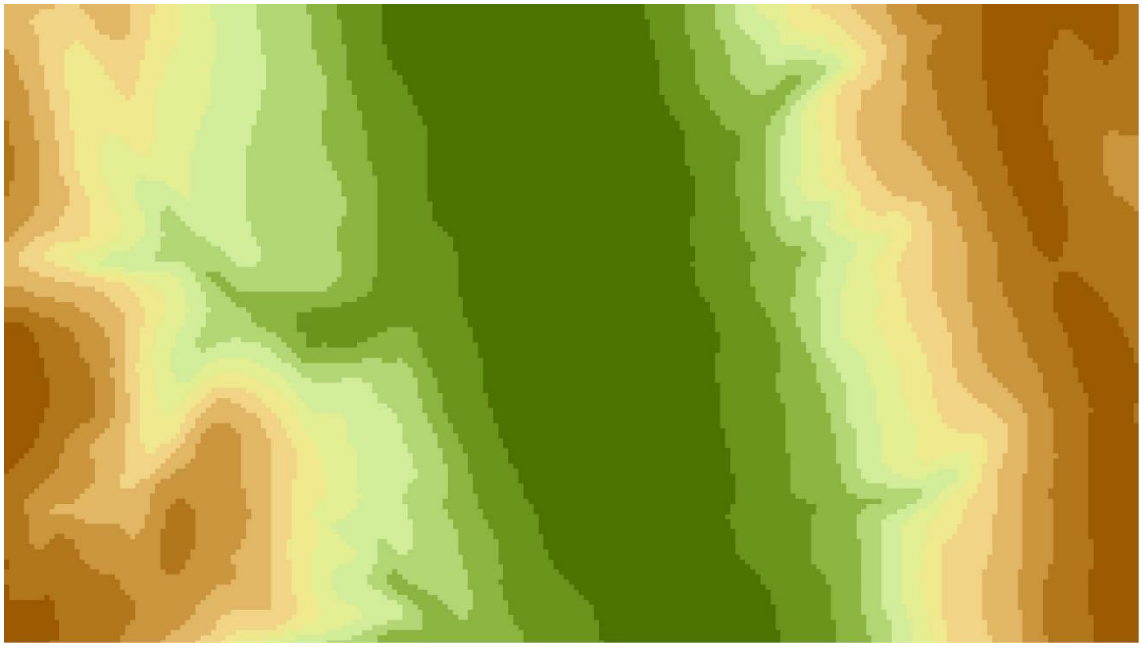
A jobb oldalon szereplő erős barnák azért nem jelennek meg a képen, mert a maximum értéket manuálisan a raszter maximumán túlra helyeztük.

Ezen hozzárendelési színrámpák, vagy függvénykapcsolat szerint, vagy adatszótár szerűen írják le a megjelenítendő színt, például 1 ≔ [0; 13; 17]; 2 ≔ [0; 20; 24] … 255 ≔ [0; 200; 255] (R; G; B).

8.1.3 Domborzatmodellek osztályozott színskálán történő megjelenítése

A rasztereket nem csak folytonos színskálán ábrázolhatjuk, hanem osztályba sorolás alapján is: itt a pixeleket nem egyedi értékeik szerint színezzük valamilyen kvázi-folytonos függvénykapcsolat, hozzárendelési függvény szerint, hanem az értékeket csoportokba soroljuk és az egy csoportba tartozó pixeleket egy és ugyanazon színnel jelöljük meg.

Ennek első és legegyszerűbb változata, amikor a raszter minimum- és maximum értéke között előre meghatározott számú osztályt hozunk létre, úgy, hogy az osztályok szélessége, az osztásköz azonos (19. ábra).



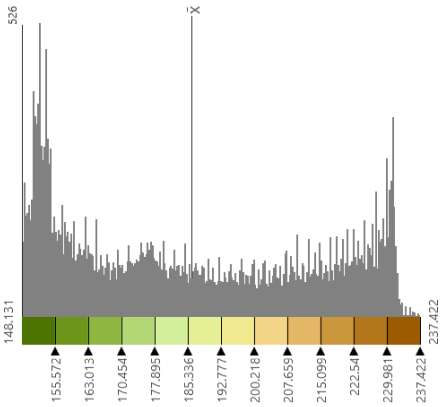
19. ábra. Színfokozatos ábrázolás 12 osztállyal, egyenlő osztásközökkel. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából.

Ezen az ábrán 12 db egyenlő osztásközű kategória szerepel: az első intervallum [148,132 m; 155,572 m[∈ ℝ, melyhez az R: 77; G: 115; B: 0, ■ színt rendeltük. Itt az intervallum szélessége: 155,572 – 148,132 = 7,44 m. A következő osztály 155,573-tól tart 163,013-ig (balról zárt, jobbról nyitott intervallumon). A szélesség ismét 7,44 m. Ehhez az intervallumhoz a R: 106; G: 148; B: 27, ■ színt rendeljük – és így lépdelünk tovább a raszter maximum értékéig (ahol egy jobbról és balról is zárt intervallum van). Az eljárás igen egyszerű és hatékony is egyben: informatikai-, matematikai alapja egyszerű és az eredmény mérhető is egyben.

Egy ilyen térképhez bármikor rendelhetünk egy olyan jelmagyarázatot, amely azt állítja, hogy egy adott szín egy adott magasságtartományhoz tartozik. Eztért ez az ábrázolási típus széles körben elterjedt például a középiskolai atlaszokban.

A térinformatikai szoftverekben az osztásköz/osztályszám tetszőlegesen állítható és rendszerint úgy állítják be, hogy a térképolvasó gondolataiban élő domborzati-, morfológiai kategóriákhoz illeszkedjen, például 0–200 m → zöld → alföld.

Az egyenlő osztásközök módszere könnyen megérthető, ha megnézzük a raszter hisztogramját és annak abszcisszája mentén ábrázolt intervallumokat és a hozzá kapcsolt színeket (20. ábra):



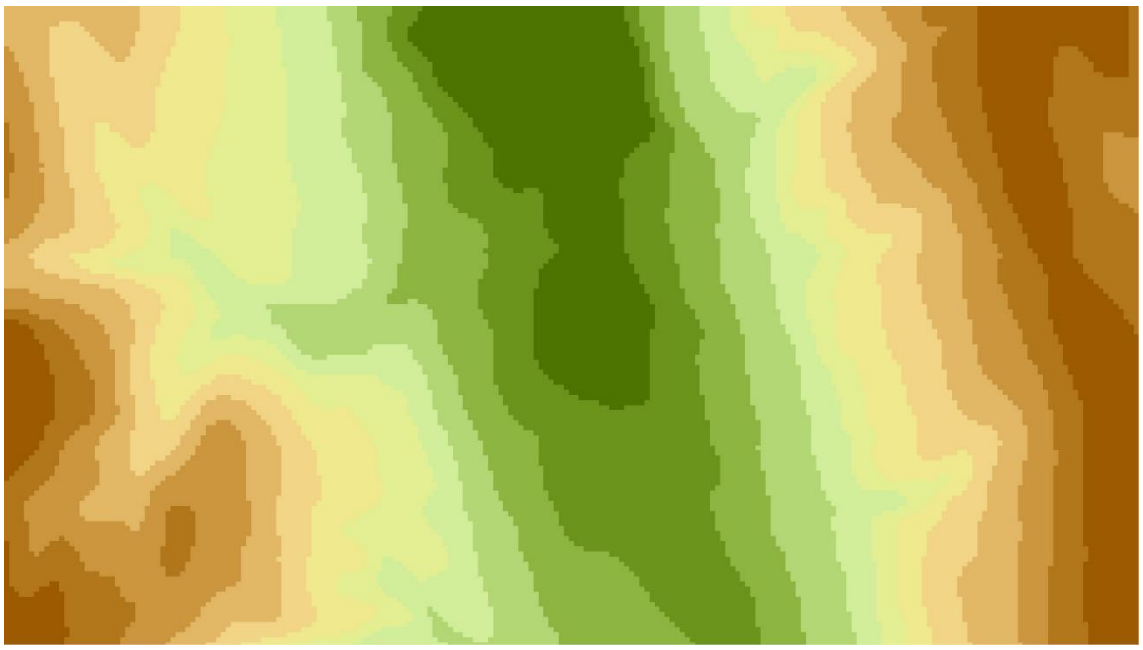
20. ábra. A 19. ábrán bemutatott DEM hisztogramja és az ehhez kapcsolt színek sorozata – egyenlő osztásközökkel.

Itt látszik, hogy a pixelértékek gyakoriságától függetlenül az osztályozás teljesen egyenközű és a minimum- és maximum között adott lépésszámban osztja fel a rasztert.

Az ilyen egyenközű osztályba sorolási megoldásnak, és az összes osztályba sorolási megoldásnak azonban van egy közös hátránya: ez az adatvesztés. Ha összehasonlítjuk az fentebbi ábrát (19. ábra) az előző, 8.1.2-es fejezet ábráival (pl.: 16. ábra) akkor az adatvesztés nyilvánvaló: számtalan morfológiai részlet az osztályozás miatt egyszerűen odaveszett.

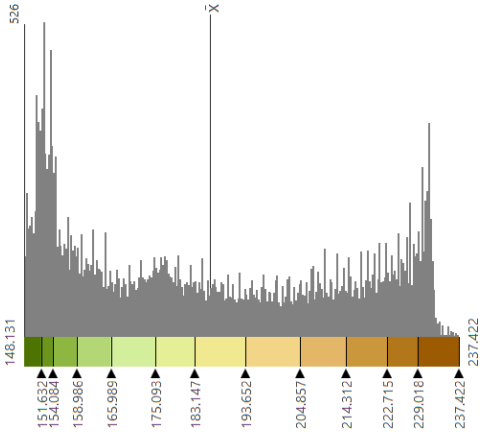
Néhány esetben ezt a hátrányt előnyünkre is fordíthatjuk: az osztályba sorolás értelmezhető úgy is, mint a kartográfiai generalizáció egy hatékony eszköze.

Második gyakori osztályba sorolási megoldás a „kvantilisek", vagy osztóérték módszere az, amikor az előre meghatározott osztályszámosság alapján úgy osztjuk fel a rasztert kategóriákra, hogy az egyes kategóriákba azonos mennyiségű pixel kerüljön – a pixel értéke alapján (21. ábra).



21. ábra. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából.

Ebből következik, hogy az intervallumok szélessége változó (22. ábra). A raszter hisztogramján látható, hogy azokon a helyeken, ahol a pixelértékek csoportosulnak (magas oszlopok a hisztogramon) ott az osztás finomabbá válik keskenyebb intervallumokkal, és ott ahol egy adott pixelértékből viszonylag kevés van az adott raszteren ott széles átfogó intervallumok jelennek meg.



22. ábra. A 19. ábrán bemutatott DEM hisztogramja és az ehhez kapcsolt színek sorozata – a kvantilis módszer szerint felosztva. Látható, hogy az osztásközök a magasabb hisztogram-oszlopoknál sűrűsödnek.

Az így színezett raszterekhez rendelhető jelmagyarázat, de a térképolvasót zavarhatja, hogy az intervallumok nem egyenközűek. Ezen jellegzetesség gyakorlati hatásait megfigyelhetjük a völgytalp környéken: itt nagyszámú alacsony értékű pixel helyezkedik el. Az osztóértékek módszere itt több kategóriát vezet be, ezért több morfológiai forma látszik – cserébe elmossa az összes többi magasság-kategória részleteit.

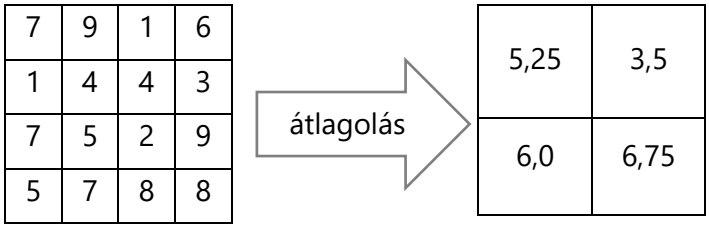
Ezen két gyakori típuson kívül alkalmazzák még a mértani haladvány szerinti, a Jenks–féle és a szórás-alapú osztályozási megoldásokat is.

8.2 Aggregáció

A raszterek aggregációja az egyik legegyszerűbb rasztermatematikai művelet, melyet felbontásváltásnál alkalmaznak. Van például egy terepmodellünk, melyben a pixelek 5 × 5 méteres terepi felbontást reprezentálnak. Ezen terepmodellt együtt kell használnunk egy Sentinel–2 műhold adatsorából szintetizált normalizált vegetációs indexszel, melynek terepi felbontása mindössze 10×10 méter. Ahhoz, hogy a két raszteren hatékonyan lehessen műveleteket végezni, a két rasztert azonos felbontásra kell hozni. Ezen példában igen nagy szerencsénk van: a két cellaméret hányadosa egész szám.

Ilyen esetekben az aggregáció úgy működik, hogy n×n cellát összevon és abból egy nagyobb méretű cellát formál, valamilyen matematikai művelet mentén, mely lehet átlagszámítás, minimum, maximum, szórás, medián, a leggyakoribb, vagy a legritkább érték is.

Itt egy 4×4-es rasztert aggregálunk, átlagszámítással 2×2-es raszterré. Mivel 4 \ 2 = 2 (ahol a '\' az egészosztás jele) az aggregáció faktora 2 (23. ábra):



23. ábra. A 4×4-es raszter aggregációja átlagolással, 2×2-es raszterré (az aggregáció fakora kettő). A számítás folyamata: (7+9+1+4)/4=5,25; (1+6+4+3)/4=3,5; (7+5+5+7)/4=6,0; (2+9+8+8)/4=6,75.

Az aggregáció természetesen nagyobb méretű raszterekkel, más összegző művelet segítségével is megtehető.

Az aggregáció azonban csak egyszerű esetekben használható: akkor, amikor az aggregációs faktor egész szám. Mit tehetünk akkor, ha az előbbi 5×5 méteres terepmodellünket 3,75×4,2 méteres raszterré kell átalakítanunk? A megoldás az újramintavételezés.

8.3 Újramintavételezés

Az újramintavételezés a raszterek felbontás-váltásának univerzális eszköze, mellyel magasabb és alacsonyabb felbontások irányában is mozoghatunk, miközben a cellák méretét, szélességük és hosszúságuk arányát is változtathatjuk.

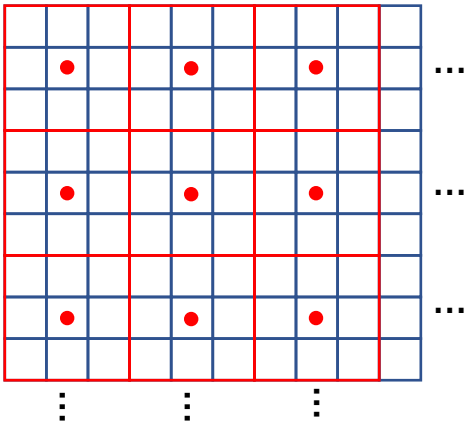
A térinformatikai szoftverek többségében az alábbi három, fő eljárás fordul elő:

1. Legközelebbi szomszédság elve szerinti,
2. Bilineáris interpoláció,
3. Görbeillesztési megoldások

Néhány ritka, speciális szoftverben ennél bővebb is lehet a lista. Itt csak azon eljárásokat ismertetjük, amelyek felskálázásnál (áttérés kis felbontástól a nagyobbra) és a leskálázásnál (nagyobb felbontásról alacsonyabba való váltás) is egyaránt használhatók.

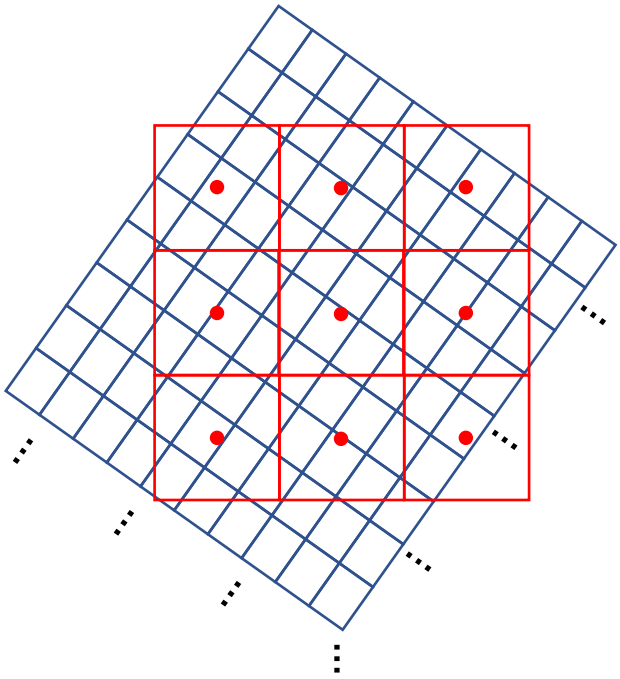
8.3.1 Legközelebbi szomszédság elve

A legközelebbi szomszédság szerinti újramintavételezés megértéséhez végezzünk el egy gondolatkísérletet! Adott egy 11 200 oszlopból és 10 500 sorból álló légifelvételünk, három színsávon (RGB). A pixelek rektangulárisok, a terepen 7 × 7 centiméteres térrészt ábrázolnak. Ezt a raszter egy 3840×2160 pixeles monitoron szeretnénk megjeleníteni. Az egész rasztert egyben. Ez egész' biztos, hogy nem lehetséges, legalábbis úgy nem, hogy egy kép pixel pontosan egy monitor pixelnek feleljen meg. Az arányokat nem tökéletesen tartva a probléma kb. ehhez hasonló (24. ábra):



24. ábra. A forrás- (kék) és céloldali raszter celláinak méretviszonya.

Itt a kisméretű, kék cellák jelölik az eredeti légifelvétel raszter celláit, a nagy, átfogó pirosak a monitor pixeleket. Hogyan alakítsuk át az eredeti rasztert, hogy a kép megjeleníthető legyen?
A legközelebbi szomszédság elve szerinti újramintavételezés azt az utat követi, hogy a cél pixel (nagy, piros, monitor-pixelek) középpontjához legközelebb eső forrás-pixel (kicsi, kék) közép-ponton feltűntetett értéket veszi fel.
Tehát, a fentebb bemutatott aggregációval ellentétben azon cellák értékeit, amelyek nincsenek a cél-pixel középpontjához a legközelebb, azokat teljes mértékben elveti és semmilyen mértékben nem vesznek részt a számításban.
A fentebbi mintát ugyan kényelmes elképzelni, de probléma rendszerint összetettebb (25. ábra):



25. ábra. A forrás- és célraszter celláinak méretviszonya elforgatott és egyik tengely mentén deformálódott helyzetben.

Itt a forrás-oldali raszter cellái nem négyzetesek, tengelyirányaik nem esnek egybe a monitor-pixelek tengelyirányaival sem. Ennek ellenére mindig lehet egy olyan forrás-oldali cellát találni, melynek középpontja a legközelebb esik egy cél-, monitor-pixel középpontjához.
Ez az újramintavételezési eljárás a térinformatikai raszterfeldolgozás alapértelmezett eljárása. Előnye, hogy szoftveresen könnyen megvalósítható, a művelethez viszonylag kevés adatot kell beolvasni (elég a legközelebbit). Alkalmas diszkrét állományok újramintavételezésére, például tájhasználati kategóriák esetében, illetve minden olyan szituációban, ahol az értékek átlaga, mediánja nem hoz érdemi eredményt.
A diszkrét állományokat egy példa mentén a legkönnyebb megérteni. Van egy felszínborítási raszterünk, amely három kategóriát tartalmaz:

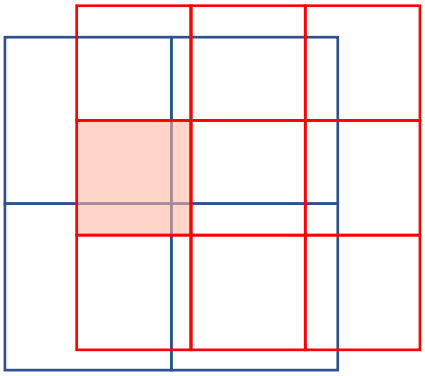
- beépített terület, melyet kódoljunk a 7-es számértékkel, 8-biten tárolt egész számmal;
- mezőgazdasági terület, melyet a 9-es számértékkel, 8-bites egészszámként jelölünk;
- erdő, melynek értéke 1.

Ebből a számsorból látható, hogy a jelenség nem átlagolható: ha van két forráspixelünk egymás mellett, közülük az egyik értéke 7 (tehát beépített terület), a másiké 9 (tehát mezőgazdasági terület). Ezek átlaga természetesen nyolc, de ilyen kategória nincs is, ezért újramintavételezésnél a legközelebbi szomszédság elvét kell használni.
A legközelebbi szomszédság szerinti újramintavételezés hátránya, hogy nem kezeli jól a folytonos adatokat. Az előbbi diszkrét-, tájhasználati raszterünk helyett képzeljünk el egy terepmodellt! Itt a számok elvileg folytonosan követik egymást, a minimum és a maximum érték között bármilyen értéket felvehetnek, melyet az adott számábrázolás képes eltárolni – nincsenek kategóriák útján előre kizárt értékek. Ha egy ilyen raszteren alkalmazzuk a legközelebbi szomszédság szerinti újtramintavételezést, akkor az eredmény lépcsős, pixeles lesz, mert az eredmény-pixel felvette azt az értéket, melyhez középpontja a legközelebb esett – a trend figyelembevétele nélkül.
Így olyan helyzetekben, ahol a forrásraszternek van valamilyen lokális trendje (pl. egy hegyoldal, ahol a számok trendszerűen emelkednek), ott az eredmény lépcsős lesz.

Ezen tulajdonságai miatt a legközelebbi szomszédság szerinti újramintavételezés a térinformatikai szoftverek alapértelmezett újramintavételezési eljárása: lehet, hogy az eredmény nem optimális folytonos adatok esetében, de legalább nem rontja el a diszkréteket sem.
A legközelebbi szomszédság szerinti újramintavételezés pozitív- és negatív irányban is végrehajtható: akkor is működik, ha nagyobb felbontásról kisebbre térünk át, és akkor is, ha kisebbről nagyobbra (csak ekkor sok lesz az egyforma pixel, de lehet, hogy pont erre van szükség).

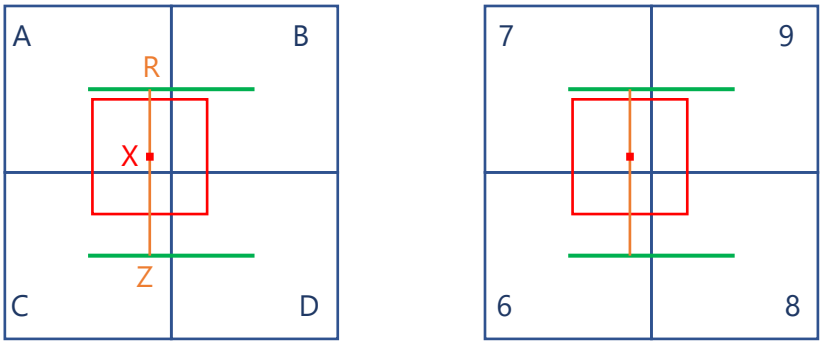
8.3.2 Bilineáris interpoláció

Ezek után joggal merül fel a kérdés, hogy hogyan kell helyesen újramintavételezni a folytonos jelenséget ábrázoló rasztereket, például a terepmodelleket. A jobb megértés érdekében itt először térjünk át rosszabb felbontásról jobbra, tehát a felhasználó erősen belenagyított a raszterbe és egy terepi- (forrás-) pixelre több monitor pixel jut (26. ábra):



26. ábra. A forrás- és célraszter viszonya. A félig átlátszó, rózsaszín térrész jelöli azt a pixelt, melynek értékét keressük.

Itt a kékek a forrás oldali (terepmodell) pixelek, melyek most nagyobbak, rosszabb felbontásúak, mint a cél-, piros-, monitor-pixelek. Határozzuk meg a rózsaszínnel kitöltött pixel értéket bilineáris interpoláció segítségével. Az egyszerűbb kezelés miatt az egyetlenegy kiszámítandó célpixelünk kivételével a többit elhagyjuk (27. ábra):



27. ábra. A keresett 'X' érték kiszámítása bilineáris interpoláció segítségével.

Itt 'A'-val, 'B'-vel, 'C'-vel és 'D'-vel jelöljük az eredeti-, forrásoldali pixeleket. Az 'A' pixel értéke 7; a 'B' 9; a 'C' 6; a 'D' 8. Minden szám 32-bites, lebegőpontos szám. Az újramintavételezés keretében a piros- (monitor-) pixel piros négyzettel jelölt középpontjának értékét keressük. (Minden raszter cellát a középpontjában lévő érték reprezentál.)

Első lépésként összekötjük az 'A' pixel középpontját a 'B' pixel középpontjával. Az '$\overline{AB}$' szakasz hossza pontosan egy pixel szélességű, ezért egységnyinek tekinthető.

Második lépésként határozzuk meg, hogy az '$\overline{AB}$' szakasz mentén hol helyezkedik el a célpixel középpontja, az '$\overline{AB}$' szakasszal párhuzamos tengely mentén. Ezt úgy érhetjük el a legkönnyebben, ha célpixel középpontjából merőlegest bocsájtunk az '$\overline{AB}$' szakaszra (narancssárga- függőleges szakasz). Jelenleg az '$\overline{AB}$' szakaszt a narancssárga merőleges ⅓ és ⅔ arányban osztja, úgy hogy a rövidebb tag az 'A' pixel középpontjához van közelebb. Azt a pontot, ahol a narancssárga merőleges eléri az '$\overline{AB}$' szakaszt, jelöljük 'R'-el!

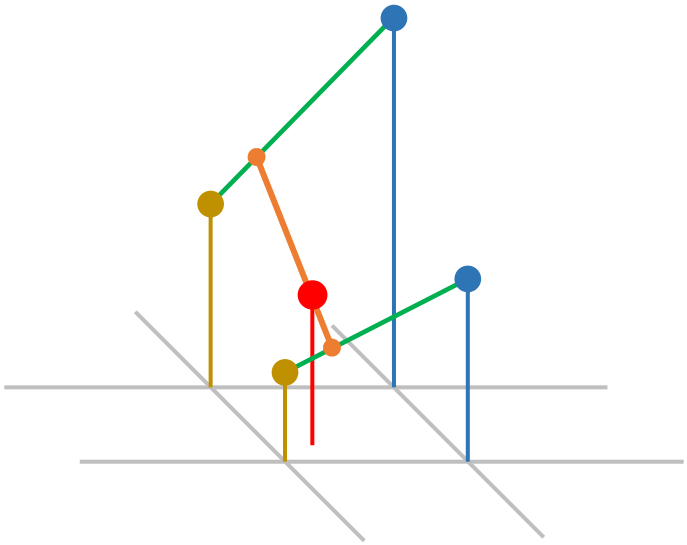
Harmadik lépésben kiszámítjuk az 'R' pont értékét: $R = ([1 - ⅓] \cdot A) + (⅓ \cdot B)$. A képletet megvizsgálva láthatjuk, hogy lényegében a súlyozott átlaggal van dolgunk. Behelyettesítve azt kapjuk, hogy $R = \frac{2 \cdot 7}{3} + \frac{9}{3} = \frac{23}{3} \approx 7{,}6666665$. Tehát az R értéke 7,6666665. (A tizedesek sora a 32-bites lebegőpontos számábrázolás korlátai miatt végződik '5'-ös számjegyre.)

Negyedik lépésben az előbbi műveletet a '$\overline{CD}$' szakaszon is elvégezzük, hogy megkapjuk a 'Z' pont értékét: $Z = ([1 - ⅓] \cdot C) + (⅓ \cdot D)$. Behelyettesítve: $Z = \frac{2 \cdot 6}{3} + \frac{8}{3} = \frac{20}{3} \approx 6{,}6666665$. (A tizedesek sora ismét a 32-bites lebegőpontos számábrázolás korlátai miatt végződik '5'-ös számjegyre.)

Negyedik-, utolsó lépésként meghatározzuk, hogy az 'X' pont milyen arányban osztja a narancssárga szakaszt, melynek mindkét végpontjában már ismert érték ül. Mivel a narancssárga szakasz hosszát itt is egységnyinek tekintjük, elmondható, hogy az 'X' pont a szakaszt 0,55 és 0,45 arányban osztja, úgy, hogy a hosszabb oldal a 'Z' ponthoz van közelebb. Ezek után a felírható az 'X' pont képlete: $X = ([1 - 0{,}55] \cdot Z) + (0{,}55 \cdot R)$. Behelyettesítve: $= ([1 - 0{,}55] \cdot 6{,}6666665) + (0{,}55 \cdot 7{,}6666665) \approx 7{,}216666$. Ezzel meg is határoztuk a cél- (monitor-) pixelünk értékét.

Természetesen valódi bilineáris interpolációnál a műveletet az összes pixelen el kell végezni, és a forrás raszter akár „elcsavarodott" is lehet a célpixelek tengelyeihez képest.

Segíthet megérteni a bilineáris interpolációt, ha térbeli jelenségként gondolunk rá (28. ábra):

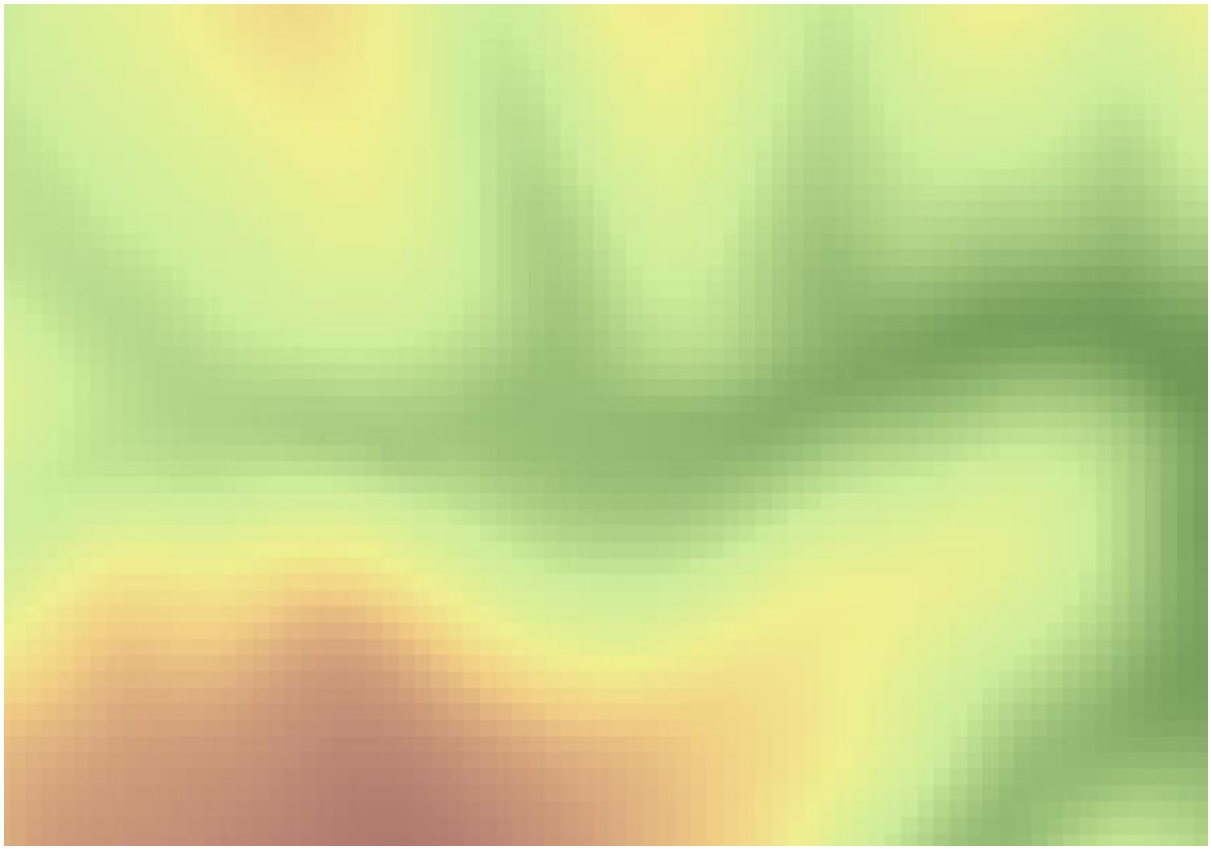


28. ábra. A bilineáris interpoláció térbeli értelmezése.

Az előző ábrákhoz hasonlóan itt is a zöld szakaszok jelölik a forrás pixelek középpontját összekötő szakaszokat, a narancssárga a két interpolációs segédszakaszt összekötő szakasz, a piros pont pedig a keresett érték az új, újramintavételezett felbontásban. Az alsó szűrke rácsháló a forráspixelek középpontjain halad át (a körülhatárolt terület nem egy pixelt jelöl).

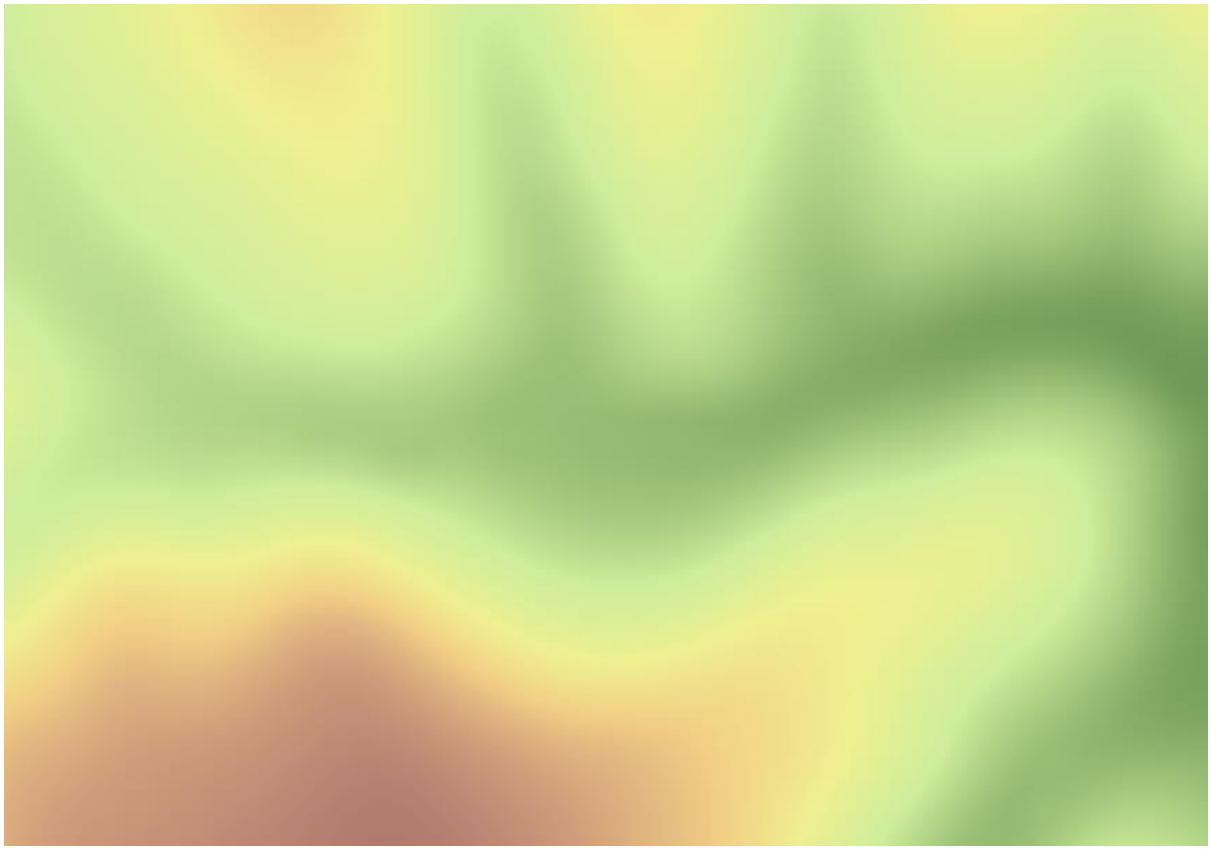
A bilineáris interpoláció nagyszerű választás folytonos adatsorok megjelenítésére, de soha ne használjuk diszkrét adatok újramintavételezésére, mert nemlétező számértékeket told az adathoz.

Nézzük meg a két, idáig megismert újramintavételezés különbségét a gyakorlatban is! A 29. ábra egy legközelebbi szomszédság elve szerint újramintavételezett domborzatmodellt ábrázol, melyet erősen „túlnagyítottunk”, így a terepi pixelek jóval nagyobbak, mint az aktuális monitor pixelek, ezért az eredeti terepmodell pixelek – értékismétlődéssel – több monitorpixelre is rákerülnek:



29. ábra. Domborzatmodell újramintavételezése jelen esetben helytelen, legközelebbi szomszédság elve szerint. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából.

Láthatjuk, hogy az eredmény meglehetősen pixeles, mert folytonos jelenséghez helytelen újramintavételezést választottunk. Nézzük meg ugyanezen ábrát bilineáris interpolációval is (30. ábra):



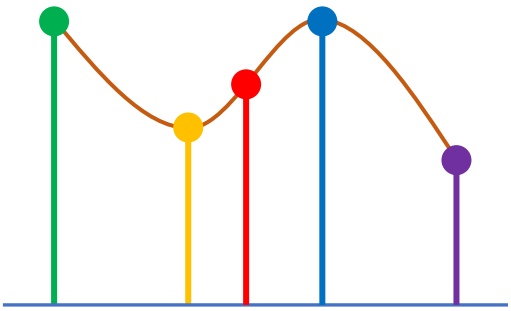
30. ábra. Domborzatmodell, bilineáris interpolációval. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából.

Látható, hogy az eredmény sokkal jobb, mert az elvileg folytonos felszín, ha még interpoláció árán is, de megőrzi folytonos viselkedését és nem lépcsősödik, pixelesedik.

8.3.3 Görbe illesztési megoldások

A görbeillesztési megoldások egy ernyőfogalom, amely két matematikailag különböző, de vizuálisan nagyon hasonló eljárást foglal magában: a „B-Spline” és „Cubic Convolution”-t.

A kétdimenziós, raszteres görbe illesztési megértésekez először nézzük meg, vonal mentén, egy dimenzióban (31. ábra)!



31. ábra. Görbe illesztés egydimenzióban.

Itt (balról jobbra) a zöld, a sárga, a kék és a lila pontok ismertek, melyek egymástól azonos távolságra, egymás mellett helyezkednek el a számegyenes felett. A keresett, ismeretlen pontunk a piros, amely a sárga és a kék között helyezkedik el. A piros pont értéke ismeretlen, tehát interpolációval kell meghatároznunk.

Itt először meg kell keresnünk a piros ponttól jobbra- és balra eső két-két ismert pontot. Ezen pontokra egy görbét illesztünk, majd a görbe értékét a keresett pontban leolvassuk.

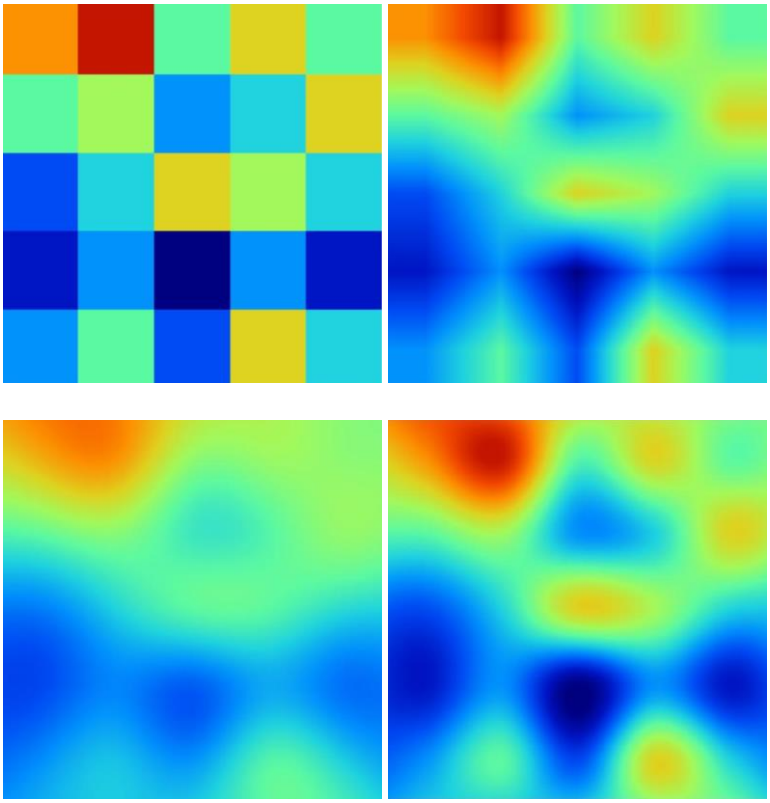
Ezt két dimenzióban, egy raszteren is megtehetjük, a bilineáris interpolációnál bemutatottakhoz hasonlóan: először összekötjük az egyik irányban az ismert pontokat négy görbével (kettő a keresett pont egyik-, kettő a másik oldalon), majd az erre merőleges irányban az interpolált vonalakon négy segédpontot veszünk fel. Ezen pontokat ismét összekötjük egy másik görbével, majd erről a görbéről leolvassuk a keresett értéket.

Így a görbe-illesztéses interpoláció során a szomszédos tizenhat pont értékét vesszük figyelembe, úgy, hogy a pontokra egy matematikai szabályokkal leírható felületet feszítünk.

Ez az eljárás csak folytonos jelenségek esetében használható, mert az osztályozott adatokat „meghamisítja".

A görbeillesztés viszonylag ritkán fordul elő a térinformatikában, főleg a szintetikus apertúrájú radarképek újramintavételezésénél használják, zavarszűrés céljából.

A négy, idáig megismert eljárás kimenetét az alábbi, 32. ábrán hasonlíthatjuk össze:



32. ábra. Bal, felső saroktól kezdve órajárás szerint: legközelebbi szomszédság szerinti-, bilinieáris-, Cubic Convolution-, és B-Spline újramintavételezés. A mintaképek SAGA-ban készültek (Conrad, et al., 2015).

Itt, a felső sorban a baloldali pixeles ábra a legközelebbi szomszédság elve szerint, a jobb, a bilineáris interpolációval készült.

A második sor a görbe illesztési megoldásokat mutatja be: a bal oldali a B-Spline, a jobb oldali a Cubic Convolution eredménye. A B-Spline-t bázisgörbének, a Cubic Convolution-t négyzetes konvolúciónak, Bicubic Convolution-nak és Bicubic Spline interpolációnak is nevezik az egyes szoftverek.

Fontos, hogy a felsorolt eljárások közül a görbe illesztési megoldások sem nagyobbról kisebbre, de még kissebről nagyobb felbontásra való áttérés esetén sem őrzik meg az adat minimum és maximum értékét, hanem egy *kicsit módosíthatják* az. Ez a jelenség látható a fentebbi ábra alsó sorában: ez egyik eljárás szűkítette, a másik tágította az adat terjedelmét, miközben a színezés az eredeti intervallumon maradt.

A legközelebbi szomszédság elve, valamint a bilineáris interpoláció szerinti újramintavételezés is csak akkor őrzi meg az eredeti adatsor minimum- és maximum értékét, ha alacsonyabbról magasabb felbontásra térünk át (nagy cellaméretről kisebb terepi cellaméret felé), mert ellenkező esetben a pixelek egy része „kimarad” a számításból. Például, ha egy 10 méteres terepi felbontású rasztert újramintavételezünk legközelebbi szomszédság elve szerint 600 méteres felbontásúra, akkor az eredmény pixelek száma $^{1}/_{3\,600}$-ára csökken ennek megfelelően a pixelek

$3\,599/3\,600$ része (~99,972%) egyszerűen be sem kerül a számításba, így az ott lévő minimum-/maximum értékek elvesznek. Ezen veszteség negyedével, de a jelenség a bilineáris interpolációnál is fellép.

8.3.4 Megjelenítés képernyőn

Abban az esetben, ha a felhasználó egy adott terepi felbontású rasztert szeretne a képernyőn megjeleníteni, akkor az újramintavételezéshez ki kell számítani, hogy az aktuális méretaránynál egy monitor pixel, mekkora terepi felbontásnak felel meg: $L = \frac{0{,}0254\ m \cdot M}{R}$. Itt az 'L' a keresett pixelméret méterben, 'M' a célméretarány nevezője, 'R' az adott monitor felbontása pixel/inch (ppi) mértékegységben kifejezve, a '0,0254 m' a méter és az inch közötti váltószám ($1'' \overset{\text{def}}{=}$ 0,0254 m).

Például, ha adott egy 10×10 méteres terepi felbontású raszterünk, melyet 1:2 000 000-s méretarányban szeretnénk megjeleníteni, egy 96 pixel/inch felbontású monitoron, akkor R = 96; M = 2 000 000; $= \frac{0{,}0254\ m \cdot 2\,000\,000}{96} \Rightarrow$ L ≈ 529,16666666666667 m.

8.4 Moiré–jelenség

A moiré–jelenség (magyarosan moaré, angolul moire) a mintavételezés és így az újramintavételezés általános jelensége. Megértéséhez nézzük meg az alábbi, 33. ábrát!

33. ábra. Moiré–jelenség újramintavételezett légifelvételen. A bal alsó sarok EOV koordinátája 541 555 m, 172 376 m (Lechner Tudásközpont, 2012).

Figyeljük meg a kép bal alsó (dél-nyugati) sarkát! Itt egy érdekes parcellát látunk, melyben jellegzetes halszálkás sraffozás látható, ▧-szerű mintázatban. Néhány parcellával jobbra, keletre ennek az ellentétes irányú megjelenését látjuk, ▨-szerűen. Ha végigfuttatjuk a szemünket a képen, még több helyütt előfordulnak különböző erősségű és irányú csíkos mintázatok.

Mi okozhatja ezt? Először nézzük meg, hogy hogy került a kép a képernyőnkre. Az eredeti kép egy 0,4×0,4 méteres terepi felbontású RGBI kép, melyet 1∶15 000-es méretarányban jelenítettünk meg egy 96 ppi-s (pixel-per-inch, képpont hüvelykekként) monitoron. A méretarányból sejthetjük, hogy itt újramintavételezésre van szükség.

Az előző fejezetben megismert $L = \frac{0{,}0254\ m \cdot M}{R}$ képlet alapján kiszámíthatjuk, hogy ebben az esetben a monitor pixelei $L = \frac{0{,}0254\ m \cdot 15\,000}{96} = 3{,}96875 \times 3{,}96875$ méteresek, így bizonyos, hogy a kép újramintavételezéssel került a monitorra – az alapértelmezett beállítások szerint – a legközelebbi szomszédság elvét követve.

De mi adja a csíkos mintát? Kézenfekvő a gondolat, hogy valamilyen soros művelés, például szőlő, vagy gyümölcsfák sorait látjuk. De ha megnézzük a domborzatot és a sorok irányultságát, erősen valószínűtlen, ilyen nehezen kezelhető sorokba ültetnének – bármit is.

Nagyítsunk rá a bal alsó sarokra 1∶6 000-es méretarányig és állítsuk az újramintavételezési eljárást bilineárisra (34. ábra)!

34. ábra. (Lechner Tudásközpont, 2012).

Itt valami egészen furcsát látunk: a korábbi északnyugat–délkeleti csapású, széles sorok keskeny, észak–déli irányúvá váltottak.

Ezt a hatást moiré–jelenségnek nevezzük (egy francia selyemszövet mintázata miatt). Jelenség oka, hogy az újramintavételezés során olyan szerencsétlenül ugrunk át, vagy vonunk össze pixeleket, hogy azok egy hamis, az eredetitől akár jelentősen eltérő mintát adnak vissza. A jelenséget vizuális megjelenése miatt pixel-interferenciának is nevezik. Nem csak az informatikában, de a környezetünkben és a természetben is megjelenik. Az összes diszkrét mintavételezési eljárást használó szenzor, így az emberi szem is kitett a moiré-nek. Megjelenésének feltétele, hogy valamilyen szabályos, vagy közel szabályos, ismétlődő, az egyes elemek között nagyobb kontrasztú mintát vizsgáljunk.

A moiré–jelenséget és az ebből fakadó hibás értelmezést teljeskörűen kiküszöbölni nem tudjuk, de megfelelő újramintavételezési eljárással (bilineáris, bikubikus) és a felbontás növelésével felbukkanásának valószínűségét csökkenthetjük.

A jelenség rokonságot mutat az hangtechnikában jól ismert Nyquist–frekvenciával, mely megadja, hogy mintavételezési eljárás frekvenciájának fele mintázható megbízhatóan. Ez képi környezetre lefordítva azt jelenti, hogyha az eredeti felvétel 0,4 méteres pixeleket tartalmaz, akkor a legkisebb ismétlődő, periodikus terepi jelenség, amely megbízhatóan vizsgálható 0,8 méteres. Ennél kisebb sorközű, ismétlődő jelenségek nem mintázhatók megbízhatóan ezen a felbontáson és számítanunk kell a moiré–jelenség felbukkanására.

A 33. ábrán azért jelent meg a moiré, mert a vizsgált szőlő sorköze kb. 3,5 méteres miközben a monitorra újramintavételezett pixelméret kb. 4 méter, ezért a csíkok, hamisan 8 (2×4), vagy 16 (4×4) méteres lépésekben jelennek meg az interferenciától függően.

8.5 Konvolúció

A konvolúció az egyik leginkább használt rasztermanipulációs eljárás. Széles körben használják eredeti-, vagy módosított változatait a térinformatikában, a hagyományos képfeldolgozásban és a mesterséges intelligencia minden szegletében.

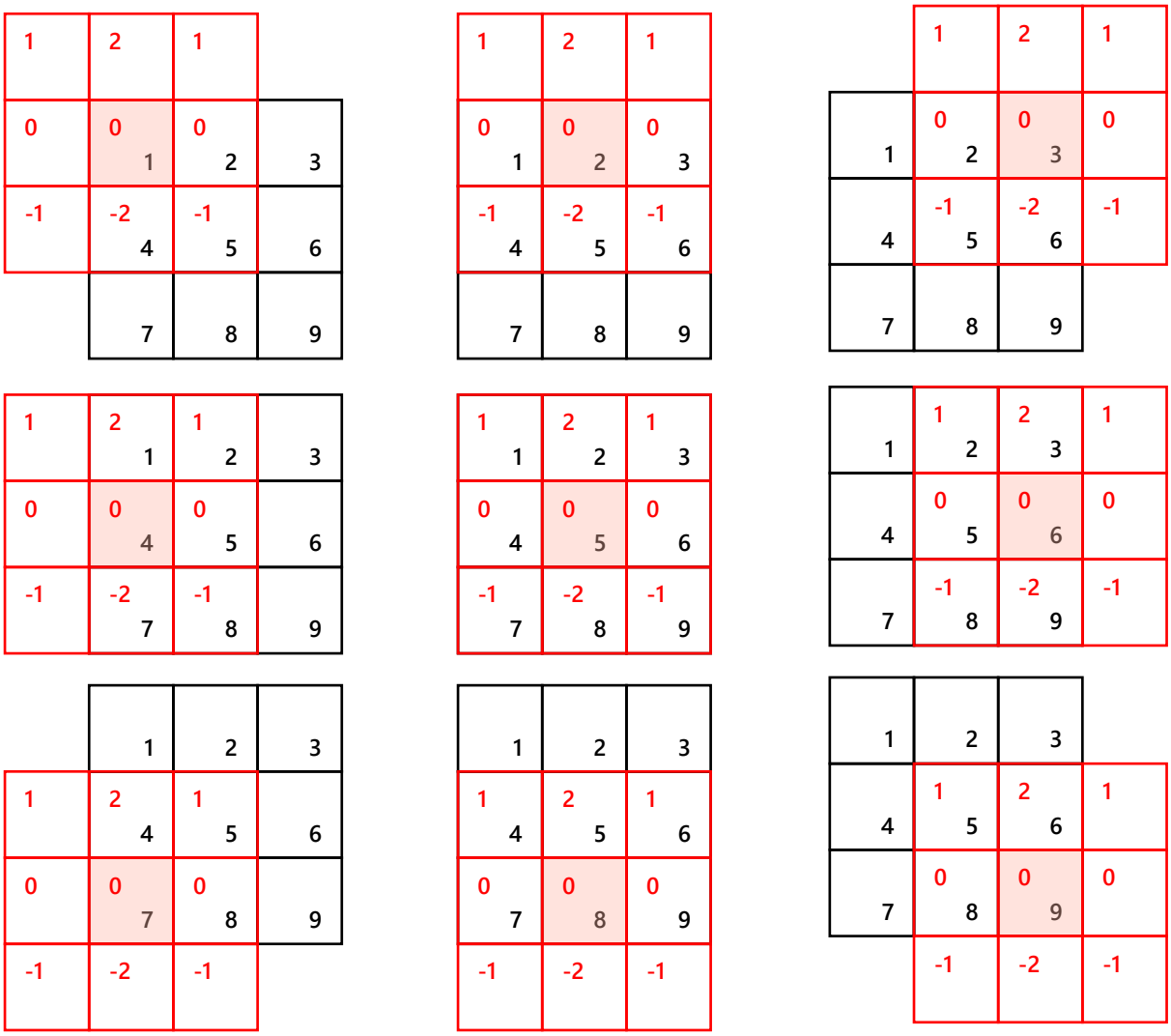
A konvolúció alapötlete igen egyszerű. Vegyünk két tetszőleges méretű rasztert! Az egyiket nevezzük el bemeneti raszternek, a másikat konvolúciós magnak. Az egyszerűség kedvéért, itt most 2 db, 3×3-as raszter szerepel. A fekete keretű, bemeneti raszter egész számokat tartalmaz, egytől kilencig. A konvolúciós mag piros, a felső sorban pozitív-, az alsó sorban negatív értékek szerepelnek, köztük pedig nullák (35. ábra):

1	2	3
4	5	6
7	8	9

1	2	1
0	0	0
-1	-2	-1

35. ábra. Raszter (bal oldali fekete) és a raszteren alkalmazni kívánt konvolúciós mag (jobb oldali piros):

A konvolúció mindössze annyit jelent, hogy a konvolúciós magot ráhelyezzük a bemeneti raszterre, úgy, hogy a mag középső pixele a bemenet bal felső pixelére essen. Ekkor a mag egy része „lelóg”, de négy pixel ráesik a bemenetre. Az itt lévő pixeleket összeszorozzuk és a szorzások eredményét összeadjuk. A kapott számot beírjuk egy újonnan létrehozott 3×3-as raszter bal felső sarkába. Ezután a magot eggyel jobbra mozdítjuk, és az átfedő részen megismételjük a szorzásokat, majd az eredmények összeadását. Az így kapott számot az újonnan létrehozott raszter következő (felső sor, középső) pixelébe írjuk. Majd ismét jobbra lépünk; majd a következő sorra, majd ismét jobbra. A lépéseket addig ismételjük, amíg el nem érünk a jobb alsó pixelre is. Ezzel kész van a 3×3-as eredményraszterünk. A teljes folyamatot az alábbi, 36. ábra mutatja be:



36. ábra. A 3×3-as konvolúció lépései.

Számoljuk végig! 36.ábra, első sor:

$0 \cdot 1 + 0 \cdot 2 + -2 \cdot 4 + -1 \cdot 5 = -13;$

$0 \cdot 1 + 0 \cdot 2 + 0 \cdot 3 + -1 \cdot 4 + -2 \cdot 5 + -1 \cdot 6 = -20;$

$0 \cdot 2 + 0 \cdot 3 + -1 \cdot 5 + -2 \cdot 6 = -17$

Második sor:

$2 \cdot 1 + 1 \cdot 2 + 0 \cdot 4 + 0 \cdot 5 + -2 \cdot 7 + -1 \cdot 8 = -18;$

$1 \cdot 1 + 2 \cdot 2 + 1 \cdot 3 + 0 \cdot 4 + 0 \cdot 5 + 0 \cdot 6 + -1 \cdot 7 + -2 \cdot 8 + -1 \cdot 9 = -24;$

$1 \cdot 2 + 2 \cdot 3 + 0 \cdot 5 + 0 \cdot 6 + -1 \cdot 8 + -2 \cdot 9 = -18$

Harmadik sor:

$2 \cdot 4 + 1 \cdot 5 + 0 \cdot 7 + 0 \cdot 8 = 13;$

$1 \cdot 4 + 2 \cdot 5 + 1 \cdot 6 + 0 \cdot 7 + 0 \cdot 8 + 0 \cdot 9 = 20;$

$1 \cdot 5 + 2 \cdot 6 + 0 \cdot 8 + 0 \cdot 9 = 17$

Ezzel az eredményraszterünk (37. ábra):

-13	-20	-17
-18	-24	-18
13	20	17

37. ábra. A konvolúció eredménye.

De mi a célja a konvolúciónak? A konvolúcióval információt nyerhetünk ki a raszteres adathalmazból. Konvolúció a térinformatikában jól ismert lejtőszög, kitettség és domborzatárnyékolás is, de klasszikus alkalmazásai az élkiemelés, élesítés és lágyítás.

Az alábbi, 38. ábra egy 8-bites szűrkeárnyalatos jelenetet mutat be:

38. ábra. A szerző saját felvétele, Allied Vision Alvium 1800 u-2050m, IMX183-as monokróm kamera, Kowa LM6FC24M | 1.1" 6 mm-es objektív.

Konvolváljuk egy 3×3-as élkiemelő- (Laplace–) operátorral! Az élkiemelés konvolúciós magja:

0	-1	0
-1	4	-1
0	-1	0

Az eredmény az alábbi képen látható (39. ábra):

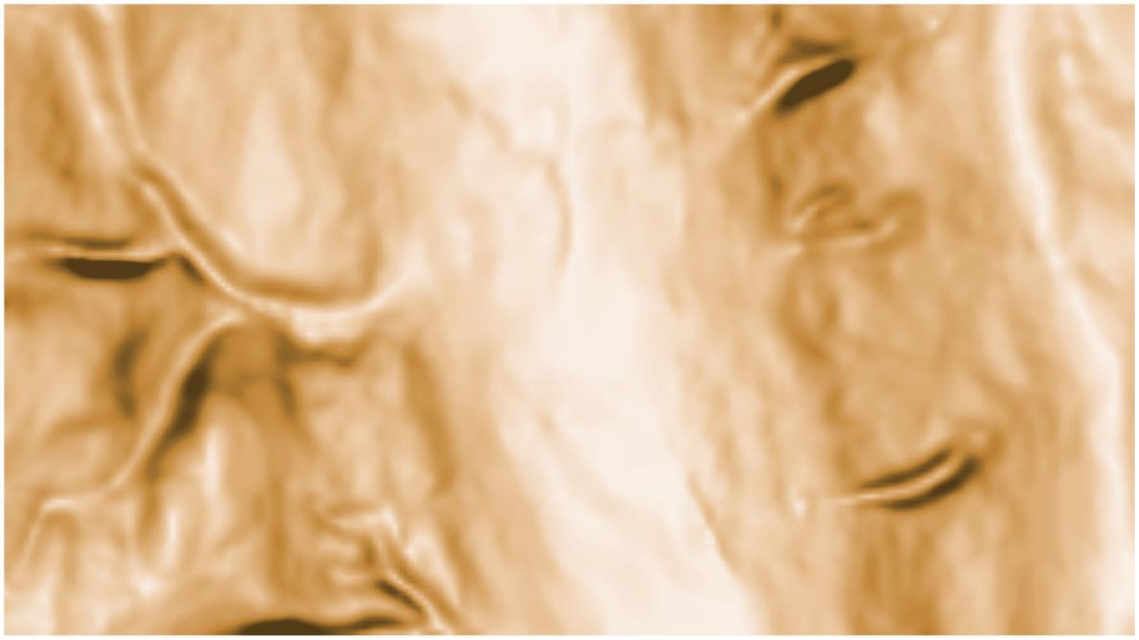
39. ábra. Az élkiemelő konvolúció eredménye.

Látható, hogy a konvolúció kiemelte az éleket (magas pixelértékek) a közel homogén kitöltésű részeket pedig kimaszkolta, a pixeleket nullára állította, ezzel egy a felhasználó számára fontos elemet – az információt – kiemelte az adatból.

A konvolúció számtalan magot, magméretet és utófeldolgozást használhat, ezért széles körben alkalmazzák.

8.6 Lejtőszög ábrázolása

A raszteres domborzatmodellek elemzéséhez köthető, egyik leggyakoribb számítás a lejtőszög ábrázolása. A lejtőszög gyakran felbukkan döntéstámogató rendszerekben (például: hova telepíthetünk szőlőt?) és vizualizációs megoldásokban egyaránt (40. ábra).

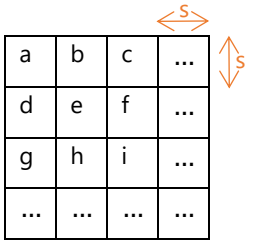


40. ábra. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából.

Itt a fehérek és sárgák sík-, vagy közel sík területeket, még a sötét barna részek meredek, lejtős területeket jelölnek. A megjelenítés módszere folytonos, hisztogram-vágott.

A lejtőszög ábrázolása nem egy egyszerű megjelenítési eljárás: a lejtés megadásához számításokat kell végezni.

Első lépésben vegyük a raszter bal felső sarkát és az ott lévő pixeleket nevezzük el az ábécé betűi szerint (41. ábra):



41. ábra.

Először próbáljuk meg kiszámítani az ‘e’ cella lejtését! Ehhez két segédmennyiséget kell bevezetnünk:

$$\frac{dz}{dx} = \frac{4\cdot(c + 2f + i) - 4\cdot(a + 2d + g)}{8\cdot s};$$

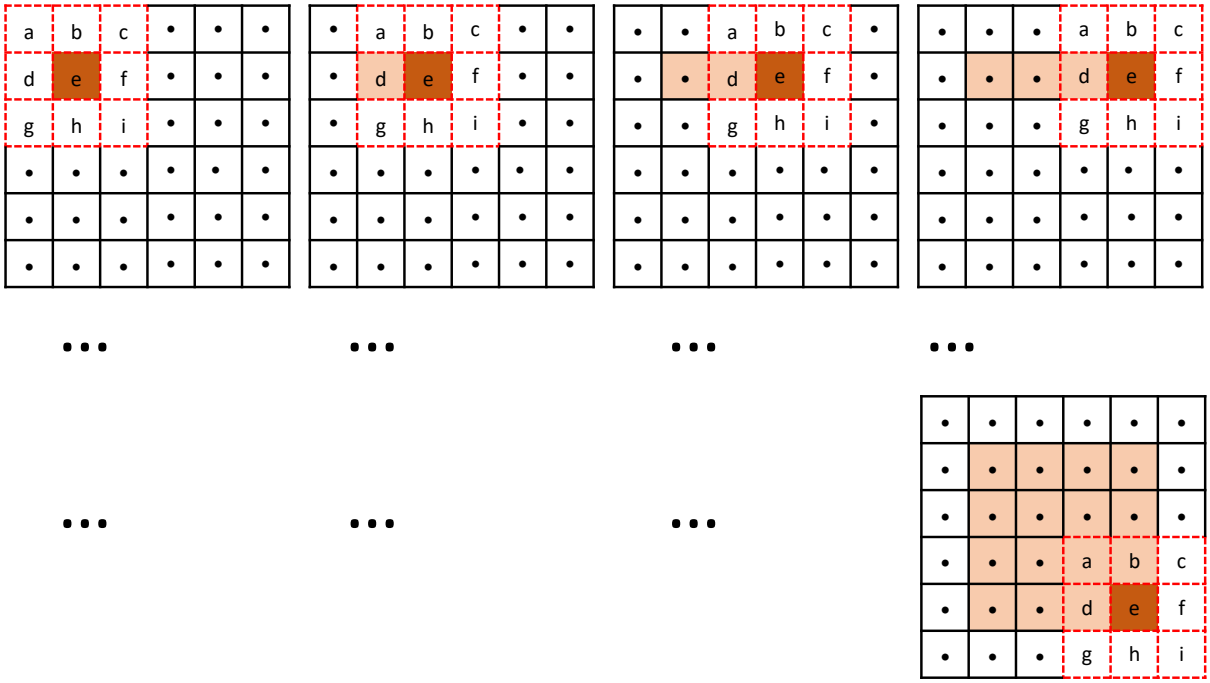
$$\frac{dz}{dy} = \frac{4\cdot(g + 2h + i) - 4\cdot(a + 2b + c)}{8\cdot s},$$

ahol az a … i (41. ábrán) betűk jelölik az adott pontban lévő pixel számszerinti értékét ‘s’ pedig a domborzatmodell cellamérete.

Ezen két segédmennyiségből kiszámítható a lejtés:

$$lejtés = \tan^{-1}\left(\sqrt[2]{\left(\frac{dz}{dx}\right)^2 + \left(\frac{dz}{dy}\right)^2}\right)$$

Az így kapott lejtés az ‘e’ cella lejtésértéke. Ez után a számítógép eggyel jobbra lépteti az egész 3×3-as maszkot és a következő középponti pixelre is elvégzi ezt a számítást. Ekkor az a … i betűk már más cellákat jelölnek, de a műveletsort ugyanúgy, a maszk középponti pixeléhez viszonyítva kell elvégezni (az ‘a’ mindig egy lépéssel balra, és egy lépéssel felfelé helyezkedik el, a ‘b’ egy sorral felfelé…).



42. ábra. A lejtőszögszámítás menete. Az egyszerűség kedvéért a raszter celláinak számértékét ‘•’ jellel helyettesítettük. Ugyanígy, a köztes lépéseket ‘…’ jelöli. Az aktuális ‘e’ cellát, melyre a számítás éppen megtörténik a ‘■’ szín, míg a már kiszámított (és egy másik, új raszterre beírt) eredményeket a ‘■’ szín jelöli. A piros, szaggatott vonal az aktuális 3×3-as maszk kiterjedése.

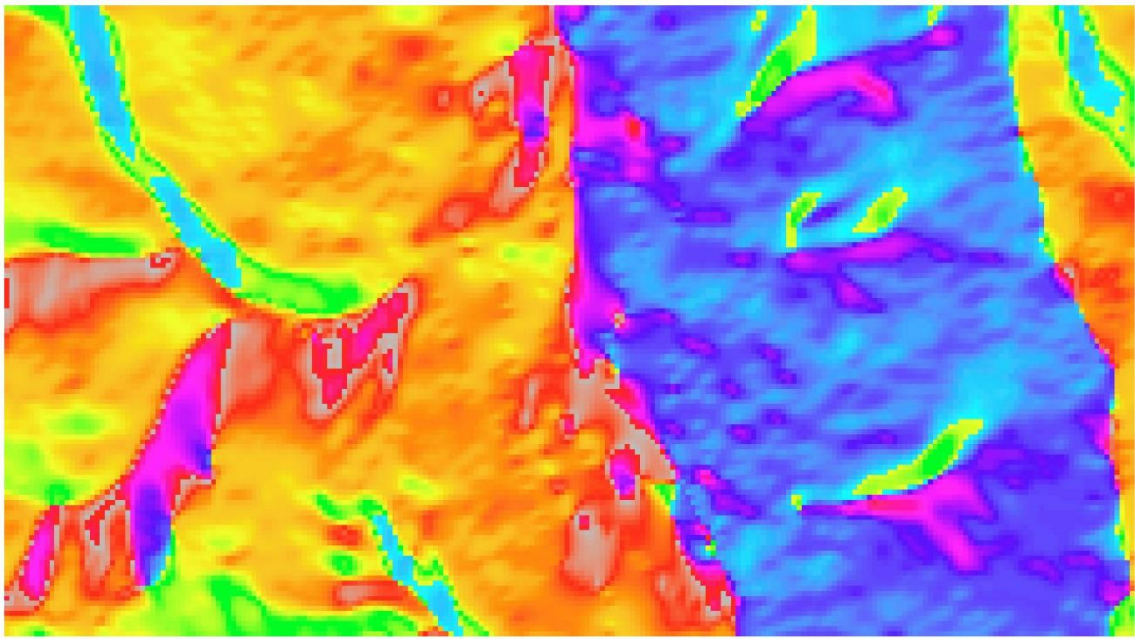
A léptetés mindig átfedéssel történik: a korábbi ‘b’ az új ‘a’, a korábbi ‘c’ az új ‘b’. Ha a számítógép eléri a sor végét, a sor utolsó cellája az ‘f’-re esik. Ez után átfedéssel, egy sorral lejjebb ugrik, majd ismét végighalad a soron. A 3×3-as maszk soha nem lóghat le az eredeti raszterről. A számítások eredményét a szoftverek egy újonnan létrehozott raszteren tárolják el.

A lejtésszámítás módszerének egyik következménye, hogy az eredmény-raszter két sorral és két oszloppal kevesebbet tartalmaz, mint a bemeneti raszter (különben a 3×3-as mintánk a raszter szélén „lelógna”).

Tehát a lejtés ábrázolásához a háttérben egy új rasztert kell létrehoznunk az általunk kiszámított értékekkel, majd ezt a hagyományos, korábban látott ábrázolási eljárások egyikével megjeleníteni.

8.7 Kitettség

A kitettséget – a lejtőszöghöz hasonlóan – számolni kell. A kitettség megadja, hogy egy adott cella mely földrajzi irányba lejt. A lejtést negatív körüljárási irány szerint (óramutató járásával megegyezően) az északi iránytól mérjük, fokban, ahol az északi irány a 0°, a keletit a 90°, a délit a 180°, a nyugatit pedig a 270° jelöli ([0°; 360°[∈ ℝ). A terepmodellünkön olyan részek is lehetnek, melyek tökéletesen vízszintesek, így nem lejtenek semelyik irányba sem. Az ilyen cellákat a legtöbb térinformatikai program ' −1'-el jelöli. A kitettséget rendszerint folytonos színskálán, vagy osztályba soroltan ábrázolják úgy, hogy a piros jelöli az északias, a sárga a keleties, a ciánkék a délies, a kék pedig a nyugatias értékeket, miközben a sík területek szűrkék (43. ábra):



43. ábra. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából.

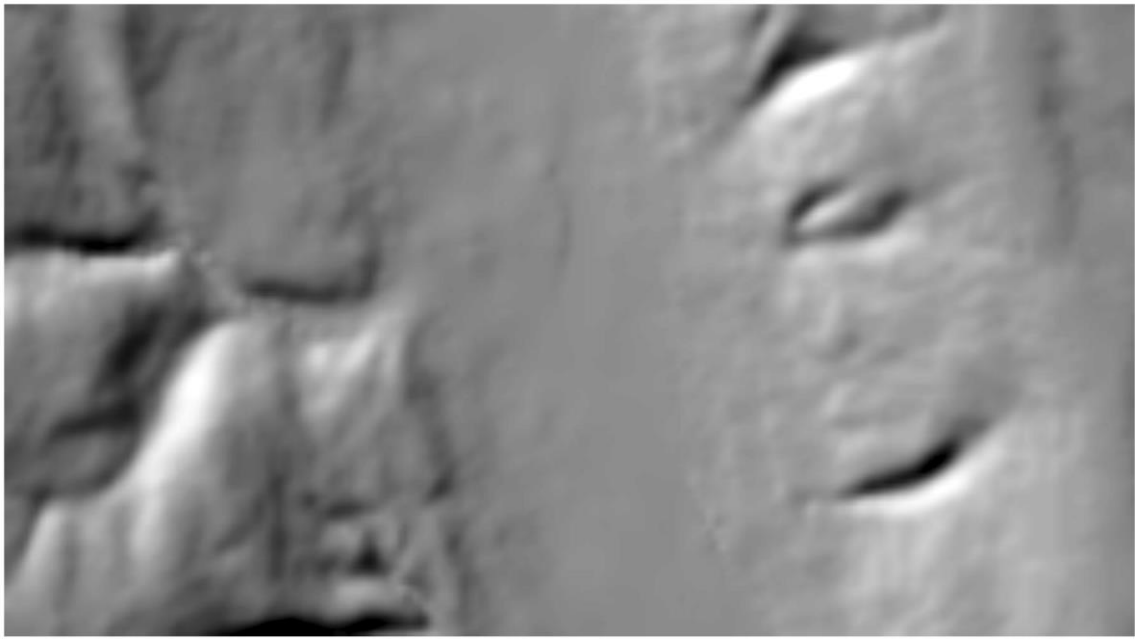
A kitettség a lejtőszögköz hasonlóan, az ott bemutatott két segédmennyiség alapján az alábbi képlettel készül:

$$kitettség = atan2\left(\frac{dz}{dx}, -\frac{dz}{dy}\right),$$

ahol az 'atan2' az arkusztangens, koordináta-negyedeket figyelembe vevő informatikai megvalósítása.

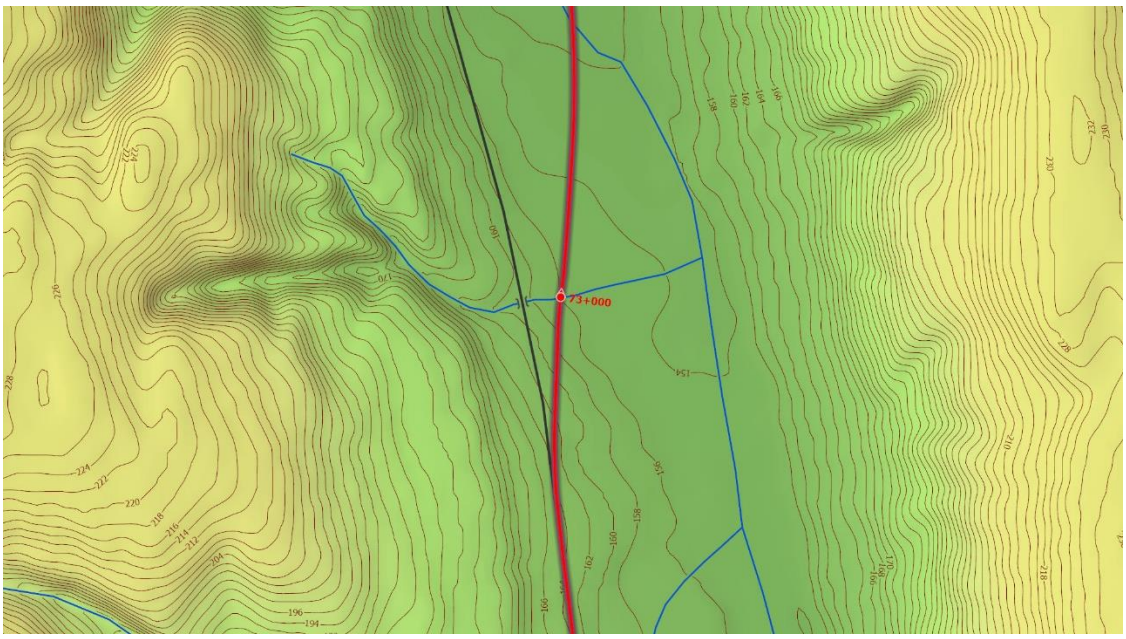
8.8 Domborzatárnyékolás

Az idáig megismert eszközök közül a leglátványosabb és vizuálisan a térképolvasót legjobban segítő megjelenítési forma a domborzatárnyékolás (44. ábra):



44. ábra. Analitikus-/hagyományos domborzatárnyékolás 315°/45°-os megvilágítási iránnyal. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából.

A domborzatárnyékolást rendszerint nem csak önmagában, hanem más térképi, domborzati tematika felett átlátszóan szokták alkalmazni (45. ábra):



45. ábra. Többirányú domborzatárnyékolás szín fokozatos domborzatárnyékolással és szintvonalakkal kiegészítve. Alapadatok forrása: HydroDEM 10 m – a Dél-dunántúli Vízügyi Igazgatóság jóvoltából.

A domborzatárnyékolás számítási megoldása a kitettségből és a lejtőszögből származik két segédmennyiség bevonásával.

Az első segédmennyiség a fényforrás, például a Nap helyi horizont feletti magassága ($\alpha_\odot$). Ezt a segédmennyiséget át kell váltanunk zenit-távolságra az alábbi képlet szerint: $\theta_\odot = 90° - \alpha_\odot$.

Második segédmennyiségünk a végtelenben lévő fényforrás azimutja ($\gamma_\odot$), melyet át kell számolnunk a következő képlet szerint $\mu_\odot = 360° - \gamma_\odot + 90°$.

Ezen raszterszinten konstans segédmennyiségek ismeretében a domborzatárnyékolás mértéke pixelenként kiszámítható az alábbi képlet szerint:

$$L = (\cos\theta_\odot \cdot \cos S) + (\sin\theta_\odot \cdot \sin S \cdot \cos[\mu_\odot - A]),$$

Ahol 'L' keresett árnyékolási érték, 'S' az adott pixelre korábban kiszámított lejtés, 'A' pedig az adott pixelre kiszámított kitettség mértéke (természetesen minden mennyiséget radiánban kell megadnunk). A kapott 'L' érték az adott pixelre vonatkoztatott lebegőpontos szám, mely a [0; 1] intervallumba esik. A térinformatikában a domborzatárnyékolást azonban rendszerint nem lebegőpontos számként, hanem jelöletlen, 8-bites egészszámként tárolják, melyet úgy állíthatunk elő, hogy az 'L' értékét megszorozzuk 255-el és a kapott eredményt egésszé csonkoljuk.

A domborzatárnyékolásnál elvárás, hogyha a raszterünk valamilyen vetített koordináta-rendszerben tárolja a horizontális pozíció-adatokat, akkor a horizontális- és a vertikális mértékegységnek meg kell egyezni, például, ha a horizontális vonatkoztatási rendszerünk metrikus, akkor a magasság egységének is méternek kell lenni. (Így például torz eredményt hoz, ha egy metrikus horizontális vonatkozgatási rendszerhez lábban adunk meg magasságot.)

Külön technikai problémát jelent, hogyha a horizontális vonatkoztatási rendszerünk földrajzi- vagy ellipszoidi koordináta-rendszerben rögzített: ekkor a szélesség és a hosszúság lineáris magasság-egységgé váltása nem konstans, de még nem is lineáris. Ilyenkor különböző cellaméretet kell használni az 'x' és 'y' irányban és az adott földrajzi pozíciónak megfelelően kiszámítani a függőleges torzítás mértéket. Ezt rendszerint a térinformatikai szoftverek elvégzik helyettünk.

A klasszikus/analitikus domborzatárnyékolás során a fényforrás azimutját 315°-nak (észak-nyugat), a horizont feletti magasságot pedig 45°-nak választják. Ezen számok azonban meglehetősen furcsák a 315° a 45°-os horizont feletti magassággal lehetetlen kombinációt alkot az északi féltekén. Így a megvilágítási iránynak megfelelően a terepmodell kiemelkedéseinek észak-nyugati oldalai fényesek, a déli- délkeleti lejtők pedig sötétek, így a klasszikus-, északi tájolású térképeken a felénk eső lejtők sötétek – mégis így ad tökéletes domborzat-érzetet.

Ez a jelenség túlmutat a földrajz matematikai-, informatikai aspektusain, a választ az emberi észlelés milyenségében kell keresni: emberi szemlélőként feltételezzük, hogy a dolgok sötét része a tárgy alja, amely kevesebb fényt kap, így sötét részeket feltételezzük stabilnak, a talajhoz kötöttnek. Így még akkor is ezen számokat érdemes használni, ha földrajzi értelemben ilyen megvilágítási irány a féltekénken lehetetlen. Ha kísérletet teszünk az irányok megfordítására, például az azimutot 180°-ra (délre) állítjuk, miközben a zenittávolságot 45°-on hagyjuk, akkor egy jellegzetes, invertált domborzatot kapunk, ahol a hegyek látszólag besüllyednek, a völgyek pedig kiemelkednek.

A hagyományos-/analitikus domborzatárnyékolásnak több hátulütője is van: a meredek részek elsötétülnek és a domborzat plasztikája ott már nem érződik. Ezt a jelenséget látjuk a 44. ábrán, ahol a fő-, kvázimeridionális völgybe merőlegesen befutó mellékvölgyek déli oldala elsötétedik és a valódi domborzati szituáció nem látható és nem érthető meg – pusztán a képet szemlélve.

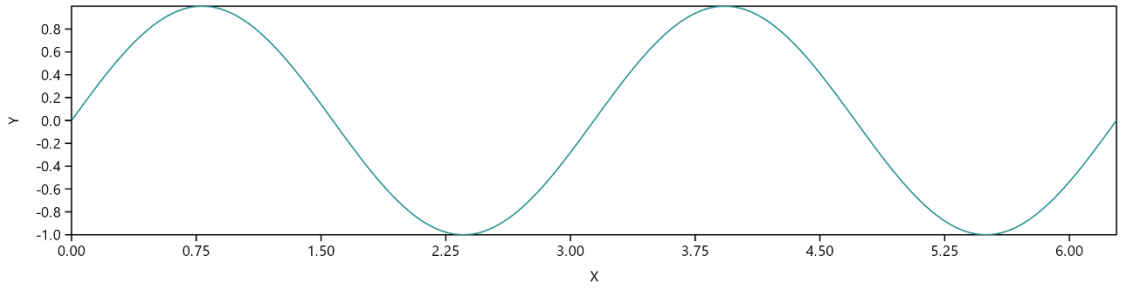
Ezen probléma kiküszöbölésére vezették be a többirányú domborzatárnyékolást, amely számítás-technikai értelemeben megegyezik az előbb leírtakkal, csak egyszerre több fényforrást használ, melyeket valamilyen, egynél kisebb súlytényezővel szoroz, úgy, hogy összegzés után a kapott eredmény egy legyen (mely 0–255-re skálázható). Az ilyen, többirányú domborzatárnyékolási megoldások általában hat megvilágítási irányt használnak, melyekben az észak-nyugatias komponensek nagyobb súllyal vesznek részt, mint a délies összetevők. A fő komponenst 315° helyett, egyes implementációkban 337,5°-nak választják. Ilyen, többirányú domborzatárnyékolást mutat be a 45. ábra.

További technikai nehézség, hogy a domborzatmodell skálafüggő jelenség: a raszterünk adott méretű cellákat tartalmaz és megjelenítő eszközünk – legyen az papír, vagy monitor – korlátozott felbontású. Ha terepmodellt nem natív felbontásában nézzük – amikor egy terepmodell pixel egy monitor pixelnek felel meg – hanem kisebb méretarányokon, akkor a domborzatárnyékolás elkezd „ellaposodni” és „szőrösödni” egyszerre. Ezen problémák elkerülése végett az adott terepmodellt újra kell mintavételezni, hogy az adott képernyő-pixelméretnek megfelelő cellaméretű legyen, valamint a magasságértékeket meg kell szorozni az alábbi empirikus képlet szerint: $Z = Z_0 \cdot P^{0,664} \cdot 0{,}024$, ahol ‘P’ a monitor-pixel terepi mérete.

8.9 Fourier–transzformáció

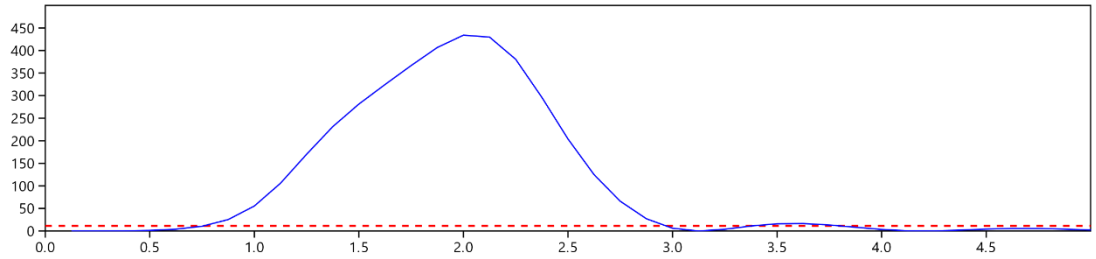
A Fourier–transzformáció egy olyan matematikai eljárás, melynek keretében egy adatsorból helyreállítható, hogy milyen frekvenciájú függvénykomponensek szuperpozíciójaként jött létre. Ez a definíció ebben a formában nehezen hasznosítható térinformatikai szemszögből, ezért nézzünk rá példákat!

Adott egy *f(x) = sin(2·x)* függvényünk, ahol $x \in [0; 2 \cdot \pi]$. Mivel digitális környezetben dolgozunk, a függvényre tekintsünk lebegőpontos számok sokaságaként és ábrázoljuk függvényként (46. ábra):



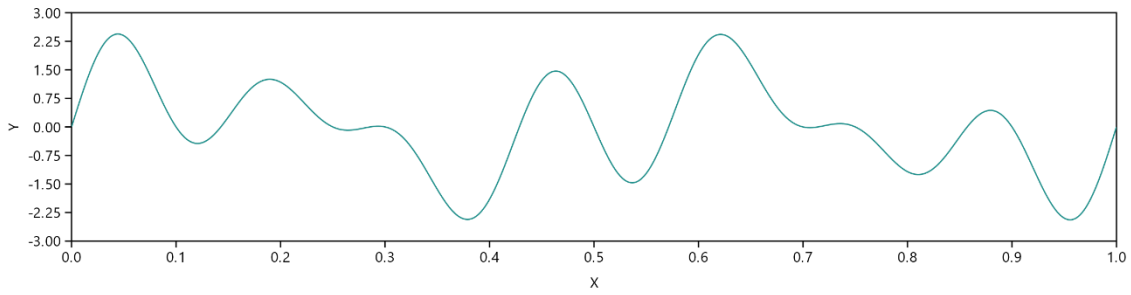
46. ábra. Az f(x) = sin(2·x); x ∈ [0; 2·π] függvény képe.

Végezzünk ezen az adatsoron egy gyors Fourier–transzformációt! Az eredmény egy frekvencia–intenzitás számsor, amely szintén függvényként ábrázolható, melyet periodogramnak nevezünk (47. ábra):



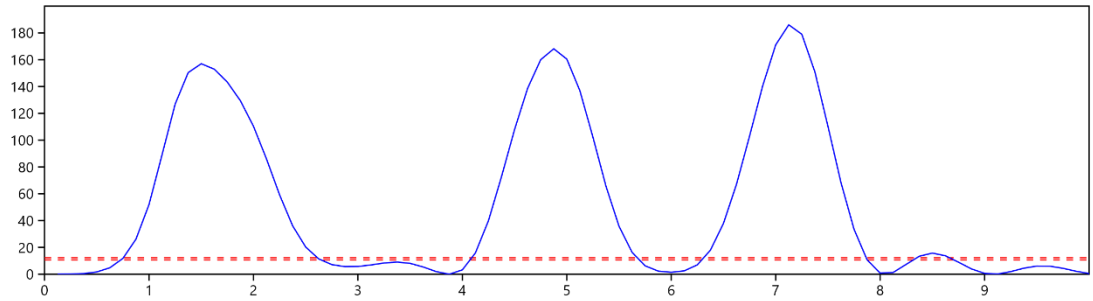
47. ábra. Az f(x) = sin(2·x) függvény periodogramja. A csúcsérték 2,0-nél van, a horizontális tengelyen. Az itt látható Lomb–periodogram a PAST 4.17-es verziójában készült (Hammer, Harper, & Ryan, 2001).

A periodogramon látható, hogy a csúcsérték 2,0-nél van. Így a Fourier–transzformáció segítségével egy hosszú adatsorból kinyertük, hogy milyen frekvenciájú szinusz hullámból tevődik a függvény össze. Tegyünk próbát egy összetetteb, *f(x) = sin(2·x) + sin(7·x) + sin(5·x)* függvény szerint létrehozott adatsorral (48. ábra):



48. ábra. Az f(x) = sin(2·x) + sin(7·x) + sin(5·x) függvény képe a [0; 2·π] intervallumon.

Ennek az adatsornak a képe meglehetősen komplex, ha nem ismernénk előre az összetevő függvényeket, munkaigényes lenne mindegyiket külön-külön meghatározni. De futtassunk egy gyors Fourier–transzformációt! Az eredmény meggyőző (49. ábra):

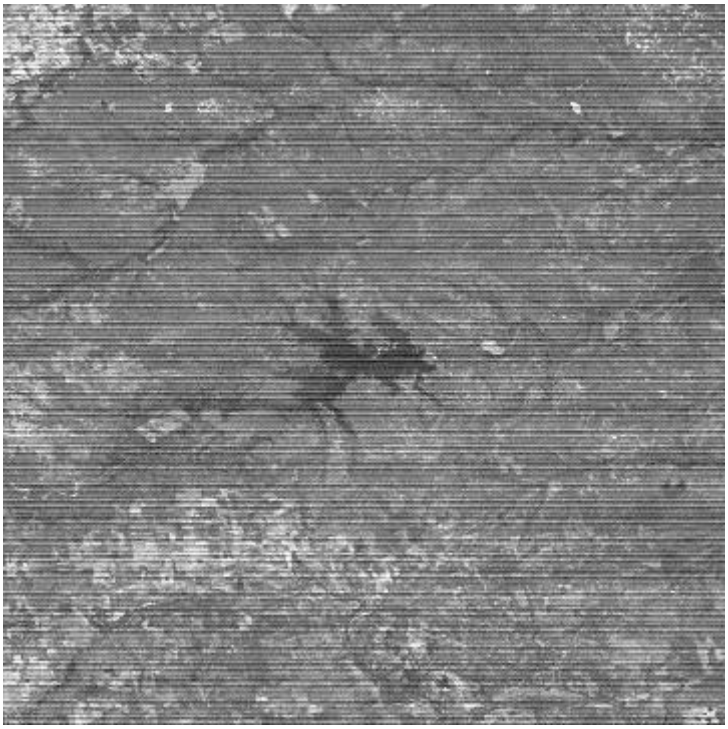


49. ábra. Az f(x) = sin(2·x) + sin(7·x) + sin(5·x) függvény periodogramja. Az adatsor rövidsége miatt a csúcsoknál némi elcsúszás látható. A két piros, szaggatott vonal a periodogram megbízhatóságát jelöli: a határérték alatti csúcsok nem megbízhatóak, mert jelenlétük vagy statisztikailag a véletlen műve is lehet, vagy a Nyquist–frekvencia közelében vannak, így megjelenésük csak látszólagos – a mintavételi frekvenciának köszönhetően (lásd: Moiré–jelenség). Ezen a periodogramon is ilyen csúcsokat láthatunk a három és a négy, illetve a nyolc és a tíz között. Az itt látható Lomb–periodogram a PAST 4.17-es verziójában készült (Hammer, Harper, & Ryan, 2001).

Az ábrán a csúcspontok az adatsor rövidsége miatt egy kissé elcsúsztak, de a három komponens ténye és azok aránya tisztán látszik. Tehát a gyors Fourier–transzformáció egy olyan eszköz mellyel fel lehet tárni, hogy egy adatsor milyen függvénykomponensek szuperpozíciójaként állt elő.

A Fourier–transzformáció invertálható is: a 50. ábrán látható periodogramból az 54. ábrán látható, eredeti adatsor (majdnem tökéletesen) helyreállítható.

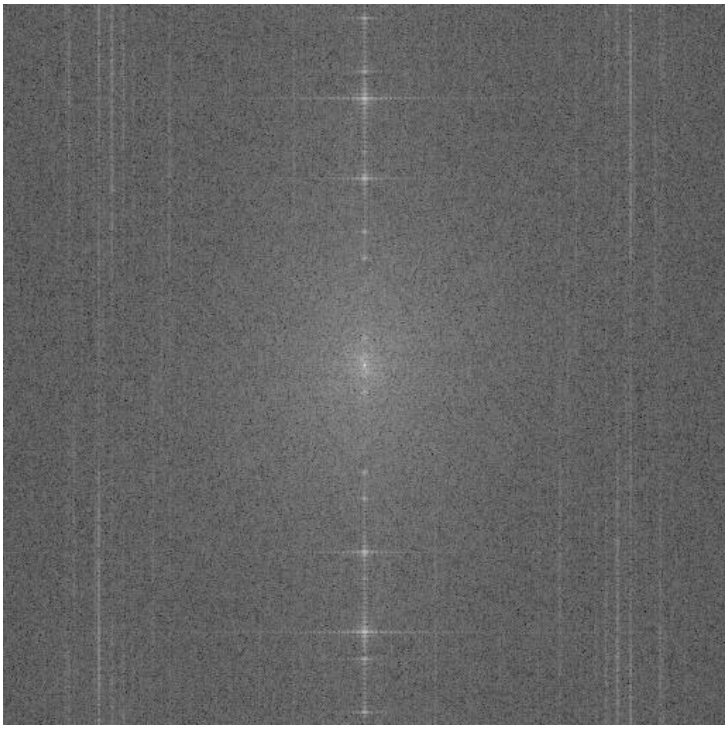
De mire alkalmas térinformatikai környezetben? Nézzünk egy régi Landsat MSS felvételt (51. ábra)! Ez egy 360×360 pixeles, 8-bites, egy sávos műholdfelvétel, melyen az egymást követő sorok kis időbeli eltéréssel kerültek a képre. Az egyes soroknak más-más „fekete” értékük, ezért a kép jellegzetesen csíkozott.



51. ábra. Feldolgozatlan (Level 0) Landsat MSS felvétel. Forrás: United States Geological Survey (2024).

Ha képet közelebbről megnézzük a csíkozás meglehetősen szerencsés: az egyes csíkoknak más és más az intenzitása, de a jelenségek (például a középen lévő, sötét tó) követhetők az egymást követő csíkokon keresztül is, így az egyik csík világosabb, a másik sötétebb, de adatvesztés nem történt.

Készítsük el a kép periodogramját a Fourier-transzformáció segítségével (52. ábra)!



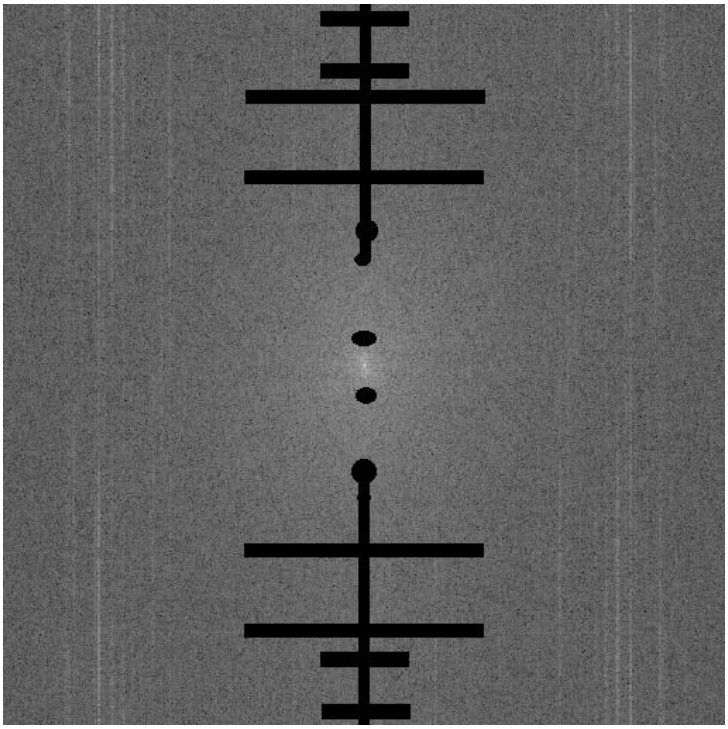
52. ábra. A feldolgozatlan Landsat MSS felvétel periodogramja. A periodogram az ImageJ alkalmazásban készült (https://imagej.net/software/imagej/).

Mivel az eredeti bemeneti kép kétdimenziós, ezért a periodogram is két dimenzióban jelenik meg, jelen esetben 512×512 pixeles formában, ahol az egyes frekvencia-komponensek intenzitása a pixel világosságában tükröződik.

A periodogramon a középponti rész körül gyülekeznek a kis frekvenciájú, nagy hullámhosszú komponensek, a szélek felé pedig egyre magasabb frekvenciák láthatók. A függőleges tengely mentén látható nagyobb intenzitású pixelek jelölik a periodikus jelenség (csíkozás) irányát. Ezen a tengelyen különböző intenzitáshoz tartozó csomópontok alakulnak ki a csíkok szélességnek függvényében. Ezek a csomópontok al- és felharmonikusokként ismétlődhetnek is.

A kép nem szisztematikus tartalma a függőleges tengelytől eltávolodva, zajszerűen jelenik meg: nem köthető hozzá intenzív periodicitás.

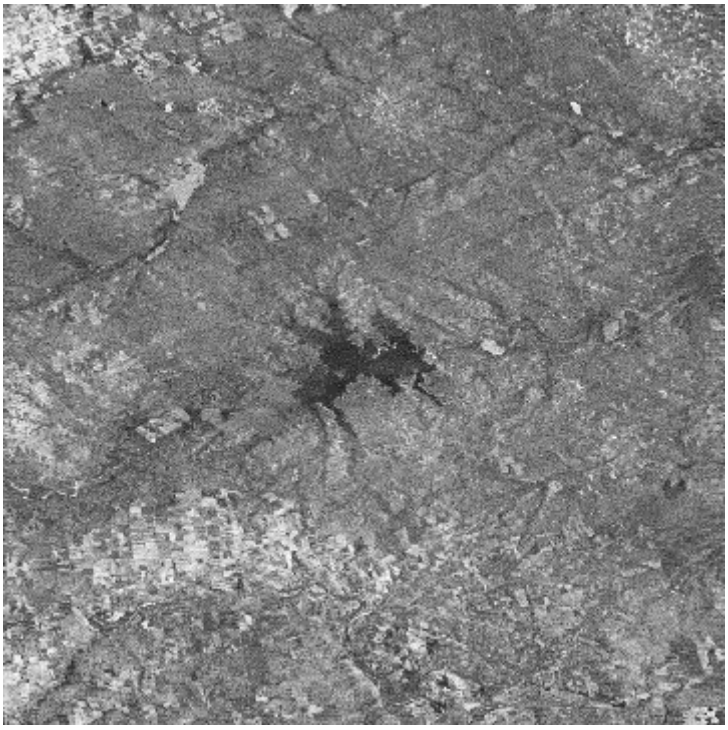
Maszkoljuk ki az intenzív periodicitással rendelkező komponenst (53. ábra)!



53. ábra. Manuálisan maszkolt periodogram.

A maszkolással a kivonni kívánt értékeket egyszerűen nullára állítjuk. Hasonlítsuk össze a maszkolás előtti és a maszkolás utáni állapotot!

Ha a maszkolás után invertáljuk a Fourier-transzformáció kimenetét, egészen használható eredményt kapunk (54. ábra):



54. ábra. Az 53. ábrából invertált transzformáció.

Látható, hogy a csíkok eltávolítása közel tökéletes és csak az eredeti, kissé zajos jelenet maradt vissza.

■

9 Hivatkozások

.NET Foundation. (2024). UTF8Encoding. Forrás: https://source.dot.net/#System.Private.CoreLib/src/libraries/System.Private.CoreLib/src/System/Text/UTF8Encoding.cs

Conrad, O., Bechtel, B., Bock, M., Dietrich, H., Fischer, E., Gerlitz, L., . . . Boehner, J. (2015). System for Automated Geoscientific Analyses (SAGA) v. 2.1.4. *Geosci. Model Dev., 8,*, 1991-2007. doi:10.5194/gmd-8-1991-2015.

ESRI. (2023). World Countries Generalized represents generalized boundaries for the countries of the world. Forrás: https://www.arcgis.com/home/item.html?id=2b93b06dc0dc4e809d3c8db5cb96ba69

ESRI. (2024). Esri Projection Engine Database Documentation. Redlands, CA, U.S.A. Forrás: https://github.com/Esri/projection-engine-db-doc/tree/main

Gade, K. (2010). A Non-singular Horizontal Position Representation. *The Journal of Navigation, 63*, 395–417. doi:10.1017/S0373463309990415

Halmai, Á. (2024). Alternative implementation for Lerc Encoding/Decoding in C#. Forrás: https://github.com/Esri/lerc/blob/master/OtherLanguages/CSharp/LercCS_Impl_B.cs

Hammer, Ø., Harper, D., & Ryan, P. (2001). PAST: Paleontological statistics software package for education and data analysis. *Palaeontologia Electronica 4(1)*.

Hazay, István (szerk.). (1956). *Geodéziai Kézikönyv* (I.. kötet). Budapest: Közgazdasági és Jogi Kiadó.

IEEE Computer Society. (2024). *Werner Buchholz*. Retrieved from https://www.computer.org/: https://www.computer.org/profiles/werner-buchholz

Lechner Tudásközpont. (2012). Orthofotó 2012. Budapest: Lechner Tudásközpont.

Maurer, T., Gao, P., & Backer, P. (2015). *U.S.A Szabadalom száma: US 9,002,126 B2.*

MÉM Országos Földügyi és Térképészeti Hivatal. (1975). *Vetületi szabályzat az Egységes Országos Vetületi Rendszer alkalmazására.* Budapest: MÉM Országos Földügyi és Térképészeti Hivatal.

Office of Geomatics, National Geospatial-Intelligence Agency. (1996). Earth Gravitational Model 96 (EGM96). Letöltés dátuma: 2024. 08 21, forrás: https://earth-info.nga.mil/php/download.php?file=egm-96interpolation

Oualline, S. (1997). *Practical C Programming* (3.. kiad.). 103A Morris St. Sebastopol, CA, United States: O'Reilly Media, Inc. Forrás: https://www.oreilly.com/library/view/practical-c-programming/1565923065/apa.html

Pirisi, G., Trócsányi, A., & Hajnal, K. (2011). Általános társadalom- és gazdaságföldrajz. Pécs. Forrás: https://tamop412a.ttk.pte.hu/files/foldrajz2/index.html

Snodgrass, R., & Camp, V. (1922). *Radio Receiving for Beginners.* New York: The Macmillan Company. Retrieved from https://www.worldradiohistory.com/Archive-Early-Radio-Assorted/Radio-Receiving-for-Beginners-Snodgrass-1922.pdf

Takács, B., & Siki, Z. (2017). Centiméter pontosságú ETRS89-EOV/Balti átszámítás nyílt forráskódú környezetben. In B. (. Balázs, *Az elmélet és a gyakorlat találkozása a térinformatikában VIII. = Theory meets practice in GIS* (old.: 355–362). Debrecen.

Trócsányi, A. (2010). A kultúrgeográfia alapjai. In J. Tóth, *Világföldrajz* (old.: 619–645). Budapest: Akadémiai Kiadó.

United States Geological Survey. (2024). Example of striping in Landsat 4 Multispectral Scanner (MSS) Level-0 data. Letöltés dátuma: 2024. 08 21, forrás: https://d9-wret.s3.us-west-2.amazonaws.com/assets/palladium/production/s3fs-public/thumbnails/image/detector-striping_true_mss.jpg

US National Geospatial-Intelligence Agency. (2024). WGS 84 (G2296) Terrestrial Reference. 7500 GEOINT Drive, Springfield, Virginia 22150, U.S.A. Forrás: https://earth-info.nga.mil/php/download.php?file=WGS%2084(G2296).pdf