

Az ionosféra GPS jelekben okozott késleltető hatásának meghatározása tomografikus úton

Szerző: Csönde Gergely

Konzulens: Dr. Rózsa Szabolcs

Budapest, 2013. október 16.



Tartalomjegyzék

1. Bevezető	2
1.1 Az abszolút helymeghatározás áttekintése	2
1.2 Szabályos hibaforrások.....	3
2. Az ionoszféra hatása GNSS műhold jelek terjedésére.....	7
2.1 Fázis- és csoportsebesség	7
2.2 Törésmutatók	8
2.3 A késleltetés meghatározása	11
3. Az ionoszférikus késleltetés valós idejű meghatározásának lehetőségei.....	14
3.1 A késleltetés meghatározása kétfrekvenciás vevők esetén.....	14
3.2 A késleltetés meghatározása általánosságban.....	15
4. A cellaalapú ionoszféra modell	18
4.1 A számítási modell	18
4.2 Az ionoszféra felosztása cellákra.....	21
5. A modell számítógépes megvalósítása	26
5.1 A modell megvalósításának áttekintése.....	26
5.2 A RINEX formátum áttekintése	26
5.3 A RINEX parser modul megvalósítása	31
5.4 A Grid model modul megvalósítása	33
5.5 Az Ionosphere modeler modul megvalósítása.....	45
6. Eredmény.....	49
6.1 A felhasznált adatok bemutatása	49
6.2 Az első kiegyenlítés eredményei	49
6.3 Az algoritmus tesztelése szimulált mérési adatokkal	51
6.4 Első eredmények.....	52
7. Összefoglalás.....	56
8. Irodalomjegyzék.....	58



1. Bevezető

1.1 Az abszolút helymeghatározás áttekintése

Napjainkban a globális helymeghatározás az egész világon elterjedt, a mindennapjaink része lett elsősorban az autós navigáció révén, és egyesek talán már el sem tudnák képzelni nélküle az életet. A legtöbb okostelefon már képes venni a GPS műholdak jeleit, így elmondható, hogy a modern világban a legtöbb ember GPS vevővel a zsebében sétál az utcán. Habár általában a készülékek nem alkalmasak tényleges helymeghatározásra, mivel a különböző alkalmazások leginkább csak egy térképen tüntetik fel a pozíciót a pontos koordináták közlése nélkül, mégis elmondhatjuk, hogy az életünk szerves részévé vált ez a technológia. Ennek ellenére sokan nem tudnak semmit magáról a technológiáról, vagy csak nagyon keveset. Ha megállítanánk az utcán embereket, és megkérdeznénk tőlük, hogy tudják-e hány műhold kell minimálisan ahhoz, hogy lehetséges legyen a helymeghatározás, nem egyszer kapnánk valami olyan választ, hogy nulla, hiszen ott van a GPS a telefonban. Valószínűleg kevesen vannak tisztában azzal, hogy több helymeghatározási módszer is létezik, mindegyiknek megvannak a saját előnyei és hátrányai, illetve, hogy az autós GPS készüléken és az okostelefonokon kívül léteznek másfajta vevők is, amelyek másfajta technológiával mérnek, másmilyen szintű pontosságot lehet elérni velük, és más árkategóriába, tipikusan nagyobbba, tartoznak. A pontosságot említve megjegyezném azt is, hogy feltehetőleg sokan még ezzel sincsenek tisztában, csak az a fontos nekik, hogy a térképen a kis nyilacska mutassa, merre kell tekerni a kormányt. A GPS csupán egy a több műhold alapú helymeghatározó rendszer közül, így át is térnék az általánosabb GNSS (Global Navigation Satellite System) kifejezésre, amely ezeket összefogja. A GNSS alapú helymeghatározási módszereket sok szempontból fel lehet osztani, én itt most csak a számunkra érdekeseket tekintem át.

A helymeghatározás során a mérési adatokból vektorokat határozunk meg, és ezekből a vektorokból kapunk földrajzi koordinátákat. Aszerint, hogy ezek a vektorok milyen két pont között futnak, megkülönböztetünk relatív és abszolút helymeghatározást. Abszolút helymeghatározás esetén egy háromdimenziós derékszögű geocentrikus koordináta-rendszerben határozzuk meg az origó és a vevő közötti vektort, azaz a vevő X , Y , Z



koordinátáit. Ezzel szemben relatív helymeghatározásnál egy ismert koordinátájú pont (egy bázis), és a vevő között határozunk meg egy vektort. Ezekből az adatokból transzformációk útján kapunk földrajzi szélességet, hosszúságot és tengerszint feletti magasságot. A második esetben a meghatározott vektorok hossza lényegesen rövidebb, a pontos nagyságrend a mérési technikától függ, de még a leghosszabbak is jóval kisebbek mint az abszolút helymeghatározásnál, hiszen ott a Föld sugarának (~6380 km) nagyságrendjében járunk. Ebből is látszik, hogy a két mérés pontossága között nagyon nagy eltérés lehet, de természetesen ezt még sok másik tényező is befolyásolja. Relatív esetben centiméteres vagy akár nagyobb pontosság is elérhető, köszönhetően annak, hogy számos szabályos hiba kiejthető, míg abszolút esetben ugyanez méteres nagyságrendű.

Megemlítenék még egy további felbontást. Aszerint, hogy a vevő mit mér, beszélhetünk fázismérésről illetve kód mérésről. Utóbbinál a beérkező műholdjel terjedési idejét mérjük, majd ez alapján a fénysebesség segítségével ki tudjuk számítani a terjedési úthosszt. Egy kis matematikával mindenki utána járhat, hogy ha van egy nanoszekundum pontossággal mérő óránk, akkor azzal ~0,3 m méter pontosan tudjuk meghatározni a műhold-vevő távolságot legfeljebb, és ebben az esetben még hibátlan mérésekről beszélünk. Persze a pontosság meghatározása nem ennyire egyszerű, ezt a példát csupán a nagyságrend érzékeltetésére mutattam. Fázis mérés esetén a beérkező műholdjel fázisát mérjük, és az egész hullámok számából illetve a mért fázisból megkapjuk a műholdvevő távolságot. A jelek hullámhossza nagyjából 20 cm, így már ebből azonnal látszik, hogy a pontosság jobb lesz, mint kód mérés esetén. Itt a gond, az egész hullámok számával, illetve annak meghatározásával van. Ezt nevezik ciklustöbbleteltértségnek. Ennek a feloldása időigényes lehet, így a fázismérés maga tovább tarthat, mint a kód mérés. Én az abszolút helymeghatározáson belül is a kód méréssel foglalkoztam. Ilyen módon végzik a helymeghatározást a legegyszerűbb (és egyben legolcsóbb) vevőkészülékek, illetve az okostelefonok is.

1.2 Szabályos hibaforrások

A következőkben a GNSS jelek méréseivel kapcsolatos főbb szabályos hibaforrásokat, illetve azok nagyságrendjét tekintem át röviden. A szabályos hibákat sok esetben ki tudjuk ejteni megfelelő mérési módszerekkel, illetve ki tudjuk számítani a mértékét, és ennek megfelelő korrekciókat alkalmazunk. Sajnos, mint látni fogjuk, vannak olyan esetek amikor, nem rendelkezünk kellő ismerettel a hibákról, és kiejteni sem tudjuk őket, így azok benne



maradnak a mérésekben, és rontják a pontosságot. A hibaforrásokat a műholdtól a vevő felé haladva veszem sorra.

Elsőnek nézzük a műholdak órapontosságát. Mint láthattuk azt korábban, a méréseknél minden nanoszekundum számít, ezért a műholdakon van egy nagyon pontos atomóra. Ezeknek az óráknak a frekvenciája nem állandó, hanem időben változik, viszont ennek a változásnak a mértéke ismert, és néhány tényezővel megadható. A műholdak navigációs üzenetei tartalmazzák ezeket, így számítható a szükséges korrekció. Ennek nagyságrendje akár 1,5-2 méter is lehet.

A következő hibaforrás a műholdak pályahibája. Mivel a helymeghatározás azon alapul, hogy kiszámítjuk a műhold és a vevő távolságát, elengedhetetlen, hogy kellő pontossággal ismerjük a műhold pozícióját, ezt pedig a pályából tudjuk meghatározni. Ezt kiejteni nem tudjuk, viszont a pályahiba mértékét különböző számítási módszerekkel le tudjuk csökkenteni olyan mértékben, hogy a helymeghatározásra gyakorolt hatása kis nagyságrendben lesz.

A műholdakat elhagyva a konstellációt vizsgáljuk meg. A helymeghatározás pontosságának szempontjából nem mindegy, hogy a műholdjaink az égbolton milyen irányban vannak. Ennek az oka, hogy a pozíciót tipikusan több műhold távolsága alapján gömbök metszéspontjainak meghatározásával számítjuk. Közel azonos irányban levő műholdak esetén ezeket a metszéspontokat sokkal pontatlanabban tudjuk meghatározni. Rendelkezésre állnak almanachok, melyekből közelítő pályaadatokat lehet számítani. Ezek ismeretében meg lehet határozni a műholdgeometria hatását, és lehet úgy tervezni a mérés időpontját, hogy minél előnyösebb legyen az.

Ezután nézzük a relativisztikus hatásokat. A műholdak meglehetősen nagy sebességgel haladnak a Föld körül, tehát a vevőhöz képest is. Ezen felül más gravitációs térben mozognak, illetve más gyorsulások hatnak rájuk. Ezek miatt a műhold órák máshogy járnak a földi órákhoz képest. Ez az eltérés meghatározható, és a műhold frekvenciájának módosításával van korrigálva.

Ezt követően a műhold jel elhagyja a műholdat, és terjed az űrben fénysebességgel, majd eléri az ionoszférát. Mint azt tudjuk, az elektromágneses hullám csak vákumban terjed fénysebességgel, így az ionoszférán áthaladva kódolás esetén késleltetést szenved, fázis mérés esetén pedig sietni fog. Az eltérés elsősorban a szabad elektrontartalomtól függ. Mivel az ionoszféra állapota időben és térben is viszonylag gyorsan változik, jelenleg nem igazán állnak rendelkezésre módszerek arra, hogy valós időben lemodellezzük. Ennek ellenére van mód a hatás kiküszöbölésére. A késleltetés mértéke az elektrontartalmon kívül függ még az elektromágneses hullám frekvenciájától is, és ezt kihasználva ki lehet ejteni ezt a hibát kétfrekvenciás vevőkészülékekben. Az érdekesebb eset az egyfrekvenciás. A dolgozatomban



ezzel foglalkozok, ennek a részletes tárgyalása a további fejezetekben olvasható. Annyit megjegyeznék még itt, hogy ezt a késleltető hatás akár több mint 10 méter is lehet.

Az ionoszféra után a következő fontos réteg a troposzféra, ahol a légkör időjárási jelenségei zajlanak. Ez a réteg már jóval összetettebb, és több fizikai jellemzőtől függ az okozott késleltetés. Nagyobb a sűrűség, a hőmérséklet, és már nem ionok, hanem molekulák alkotják. Ennek megfelelően a legfontosabb figyelembeveendő tényezők, a légnyomás, a hőmérséklet, és a vízgőz tartalom. Azért a vízgőz, mert 1-nél jóval nagyobb a törésmutatója, így jelentősen módosítja a terjedési sebességet. A troposzférikus késleltetés meghatározására is vannak egyszerűbb és bonyolultabb módszerek. Az, hogy melyiket használják, általában attól függ, hogy milyen jellegű és gyakoriságú mérési adatok állnak rendelkezésre. A késleltetés nagyságrendje pár méteres.

A jelterjedésben van még egy hibatényező, ez pedig a többutas terjedés. Itt egyszerűen arról van szó, hogy a műhold jel a talajról, illetve a környező tereptárgyakról visszaverődve hosszabb úton érkezik be a vevőbe. Ez kód mérés esetén tud nagyobb hibát okozni, fázismérésnél csupán néhány centiméter a hatása. Árnyékoló lemezekkel, speciális festéssel és további módszerekkel lehet kiküszöbölni.

Elérkeztünk a vevőoldali hibákhoz. Nézzük a ciklusugrást. Fázismérés esetén először a vevőnek meg kell határoznia a ciklustöbbsértelműséget. Amint ez megvolt, onnantól kezdve folyamatosan lehet mérni, amíg kapcsolat van a vevő és a műhold között, és periodikusan jönnek az adatok. Probléma akkor adódik, amikor valami miatt megszakad a folyamatos kapcsolat (Pl. egy dubai felhőkarcoló pont kitakar pár műholdat), és kimarad egy, esetleg több mérés. Ilyenkor a ciklustöbbsértelműség számítását újra el kell végezni. Ha ezt elmulasztjuk, az komoly hibát okozhat. Kiküszöbölni azzal tudjuk, hogy úgy választjuk meg az álláspontot, hogy ne legyen kitakaró objektumok. Előfordul olyan eset is, hogy erre nincsen módunk, viszont ebben az esetben sem feltétlenül kell újra várni a ciklustöbbsértelműség meghatározására, mivel a jelfeldolgozó programok észre tudják venni a ciklusugrásokat, és bizonyos esetekben ki is tudják javítani azokat.

Utolsó hibaforrásként említeném meg az antenna fáziscentrumának külpontosságát. A mérések során mi egy pontnak a koordinátáit szeretnénk meghatározni, viszont az antennáknak kiterjedésük van. Ez a kiterjedés összemérhető a jelek hullámhosszával, így fázismérésnél egyáltalán nem mindegy, hogy az antenna mely pontjában veszi a beérkező jelet. Ezt a pontot nevezzük fáziscentrumnak, és ez általában nem esik egybe az antenna geometriai centrumával, aminek a koordinátáit meghatározzuk. Ez a külpontosság több milliméter lehet. Szerencsére típusonként ugyanaz, így antennamodellekkel kiküszöbölhető. További nehézség, hogy a geometria középpont és a fáziscentrum közötti különbség térben



nem állandó. Ezen kalibrációs eljárásokkal lehet segíteni. A mérések utófeldolgozása során, szoftveres úton lehet kiküszöbölni ezt a hatást. Az utóbbi két hiba csak a fázismérések esetén szignifikáns, kódéréseknél elhanyagolható a mértékük.

A szabályos hibaforrásokat áttekintve elmondható, hogy a legtöbb kiküszöbölhető, vagy a szükséges mértékig számítható és korrigálható. Több olyan eset is van azonban, ahol a hiba meghatározására ugyan vannak módszerek, viszont azokat csak bizonyos mérési eljárások esetén lehet használni, vagy a módszer alkalmazásához nem áll rendelkezésre elegendő információ. Így egyáltalán nem, vagy csak pontatlanabb számítással juthatunk valamilyen eredményre. Ezek közül kiemelném az ionoszférikus késleltetést, mivel ennek az értéke igen jelentős, és egyfrekvenciás vevőkészülékkel való méréshez csak durva közelítő módszerek adóttak. A fejezetben leírtak részletesebben megtalálhatóak az [1] és [2] forrásokban.



2. Az ionoszféra hatása GNSS műhold jelek terjedésére

2.1 Fázis- és csoportsebesség

Az előző fejezetben már volt róla szó, hogy az ionoszféra szabad elektrontartalma miatt a GNSS műhold jelek sebessége megváltozik, eltér a fénysebességtől. Ennek a részletes tárgyalásához meg kell nézni, hogy pontosan mit is értünk a műhold jelek sebessége alatt. Két egymástól eltérő sebességfogalmat kell definiálnunk. Az egyik egy önálló elektromágneses hullám sebessége, amely az alábbi jól ismert összefüggés írható fel:

$$v_f = \lambda f, \quad (2.1)$$

amit fázissebességnek is nevezünk. A GNSS rendszerekben a műholdjelek sebességét fázismérés esetén ezzel a képlettel kell figyelembe venni. Egymástól kismértékben eltérő frekvenciájú jelek egy csoportjának a terjedési sebességére egy másik összefüggés használatos, mégpedig a következő:

$$v_{cs} = -\frac{df}{d\lambda} \lambda^2. \quad (2.2)$$

Ezt nevezzük csoportsebességnek, és a kódmérések esetén számolunk vele. A kétfajta sebesség közötti kapcsolat meghatározható. Először írjuk fel a fázissebesség teljes differenciálját:

$$dv_f = f d\lambda + \lambda df. \quad (2.3)$$

Ezt át tudjuk rendezni a következő formára:

$$\frac{df}{d\lambda} = \frac{1}{\lambda} \frac{dv_f}{d\lambda} - \frac{f}{\lambda}. \quad (2.4)$$



Ezt vissza tudjuk helyettesíteni a csoportsebesség képletébe (2.2), és így azt kapjuk, hogy:

$$v_{cs} = -\lambda \frac{dv_f}{d\lambda} + f\lambda. \quad (2.5)$$

Ebben a képletben felfedezhetjük a fázissebesség definícióját (2.1), és azt beírva, a két sebesség közti összefüggés a következő formát ölti:

$$v_{cs} = v_f - \lambda \frac{dv_f}{d\lambda}. \quad (2.6)$$

2.2 Törésmutatók

A következő lépésben vizsgáljuk meg az elektromágneses hullámok törésmutatóját a két különböző esetben. Egy adott közegre általánosan felírható, hogy:

$$v = \frac{c}{n}, \quad (2.7)$$

ahol c a fénysebesség, v az elektromágneses hullám terjedési sebessége a közegben, n pedig a közeg törésmutatója vákuumhoz viszonyítva. Ezt felírhatjuk mindkettő sebességre:

$$v_f = \frac{c}{n_f}, \quad (2.8)$$

$$v_{cs} = \frac{c}{n_{cs}}. \quad (2.9)$$

A fázissebesség képletének mindkét oldalát deriváljuk a hullámhossz szerint:

$$\frac{dv_f}{d\lambda} = -\frac{c}{n_f^2} \frac{dn_f}{d\lambda}. \quad (2.10)$$



Az utolsó három képletet visszahelyettesítve a sebességek közti (2.6) összefüggésbe arra jutunk, hogy:

$$\frac{c}{n_{cs}} = \frac{c}{n_f} + \lambda \frac{c}{n_f^2} \frac{dn_f}{d\lambda}, \quad (2.11)$$

vagy a fénysebességgel egyszerűsítve:

$$\frac{1}{n_{cs}} = \frac{1}{n_f} \left(1 + \lambda \frac{1}{n_f} \frac{dn_f}{d\lambda} \right). \quad (2.12)$$

Mivel a zárójelben levő kifejezés nagyon közel van egyhez alkalmazható az alábbi közelítés:

$$\frac{1}{1 + \varepsilon} = 1 - \varepsilon, \quad (2.13)$$

és így ezzel azt kapjuk, hogy:

$$n_{cs} = n_f \left(1 - \lambda \frac{1}{n_f} \frac{dn_f}{d\lambda} \right). \quad (2.14)$$

A zárójelet felbontva:

$$n_{cs} = n_f - \lambda \frac{dn_f}{d\lambda}, \quad (2.15)$$

majd végül a hullámhosszról a frekvenciára áttérve:

$$n_{cs} = n_f + f \frac{dn_f}{df}. \quad (2.16)$$



A fázis törésmutató sorba fejthető a következő formában:

$$n_f = 1 + \frac{c_2}{f^2} + \frac{c_3}{f^3} + \frac{c_4}{f^4} + \dots \quad (2.17)$$

A $c_2, c_3, c_4 \dots$ együtthatók csak az N_e elektronsűrűségtől függnak. A másodfokúnál nagyobb tagokat elhanyagolva az marad, hogy:

$$n_f = 1 + \frac{c_2}{f^2}. \quad (2.18)$$

Ezt az egyenletet differenciálva:

$$dn_f = -\frac{2c_2}{f^3} df. \quad (2.19)$$

(2.18)-at és (2.19)-et behelyettesítve a (2.16)-ba azt kapjuk csoport törésmutatóra, hogy:

$$n_{cs} = 1 - \frac{c_2}{f^2}. \quad (2.20)$$

Már most látszik az az érdekesség, hogy a fázis és csoport törésmutatók közül, az egyik biztosan 1-nél nagyobb, a másik biztosan 1-nél kisebb. c_2 becslült értéke:

$$c_2 = -40,3N_e \quad [\text{Hz}^2]. \quad (2.21)$$

Ezek szerint a csoport törésmutató nagyobb, mint a fázis törésmutató, ebből következően a fázissebesség nagyobb, mint a csoportsebesség. Továbbá azt is kijelenthetjük, hogy a fázis siet, a kód pedig késik, azaz a mért pszeudotávolságok fázismérés esetén kisebbek a valódinál, míg kód mérés esetén nagyobbak.



2.3 A késleltetés meghatározása

A Fermat-elv szerint az elektromágneses hullámok mindig a legrövidebb idejű útvonalon jutnak el egy adott pontba, ennek megfelelően az s mért távolságokra az alábbi integrál írható fel:

$$s = \int n ds . \quad (2.22)$$

Ebben a képletben n a törésmutató, ds az elemi pályaszakasz, és az integrálást a teljes terjedési útra kell elvégezni. Hasonló integrál írható fel az s_0 tényleges műhold-vevő távolságra is. Ilyenkor úgy vesszük, hogy a hullám egyenesen terjed, tehát nincsen eltérítő közeg, azaz a törésmutató értéke 1, ennek megfelelően:

$$s_0 = \int ds_0 . \quad (2.23)$$

Ezekből már felírható a mért és a valódi távolságok közötti eltérés:

$$\Delta^{iono} = \int n ds - \int ds_0 . \quad (2.24)$$

Ismét kettébonthatjuk a vizsgálódást fázis és kód mérés szerint. A képletbe behelyettesítve a megfelelő törésmutatók előző pontban meghatározott értékeit a következőkre jutunk:

$$\Delta_f^{iono} = \int \left(1 + \frac{c_2}{f^2} \right) ds - \int ds_0 , \quad (2.25)$$

$$\Delta_{cs}^{iono} = \int \left(1 - \frac{c_2}{f^2} \right) ds - \int ds_0 . \quad (2.26)$$



Alkalmazzuk azt az közelítést, hogy az első integrált is a geometriai távolságon integráljuk, így a képletben szereplő ds helyett ds_0 -t írhatunk, és az összefüggések az alábbira egyszerűsödnek:

$$\Delta_f^{iono} = \int \frac{c_2}{f^2} ds_0, \quad (2.27)$$

$$\Delta_{cs}^{iono} = - \int \frac{c_2}{f^2} ds_0. \quad (2.28)$$

Ezekbe behelyettesítve a c_2 (2.21) szerinti értékét, azt kapjuk, hogy:

$$\Delta_f^{iono} = - \frac{40,3}{f^2} \int N_e ds_0, \quad (2.29)$$

$$\Delta_{cs}^{iono} = \frac{40,3}{f^2} \int N_e ds_0. \quad (2.30)$$

Vezessük be a teljes elektron tartalom fogalmát, azaz a TEC-et (total electron content):

$$TEC = \int N_e ds_0. \quad (2.31)$$

Ezek után felírhatjuk az ionosféra fázis- és csoportképletelésének végleges alakjait:

$$\Delta_f^{iono} = - \frac{40,3}{f^2} TEC, \quad (2.32)$$

$$\Delta_{cs}^{iono} = \frac{40,3}{f^2} TEC. \quad (2.33)$$

A TEC-ről érdemes még néhány szót ejteni. Definíció szerint úgy kapjuk meg, hogy az elektronsűrűséget integráljuk a geometriai távolság mentén. Ez azt jelenti, hogy azt adja meg, hogy mennyi elektron van összesen a műhold és a vevő közötti úton (illetve egy tetszőleges vonal mentén) egy 1 m^2 -es sávban. Ebből is látszik, hogy a mértékegysége [elektron/ m^2]. A fenti képletekben a bal oldali és a jobb dimenziók látszólag nem egyeznek, de a 40,3-es



szorzótényezőnek is van mértékegysége, mégpedig $[m^3/s^2]$, és így minden a helyére kerül. Gyakorlatban a vertikális irányú elektron tartalmat (TVEC) szokták meghatározni, amit utána lehet ábrázolni hőtérképszerűen az egész Föld területén. Az általános irányú elektrontartalomra a szakirodalmak ferde elektron tartalomként (STEC) szoktak hivatkozni. Mivel a TEC értékek nagyon nagyok, bevezették a TEC egységet (TECU):

$$1 \text{ TECU} = 10^{16} \text{ elektron} / m^2. \quad (2.34)$$

A Föld éjszakai oldalán egy átlagos TVEC érték lehet például 10 TECU, míg a nappali oldalon akár 70 TECU is lehetséges. A fejezet levezetései megtalálhatóak az [1] és [2] forrásokban.



3. Az ionoszférikus késleltetés valós idejű meghatározásának lehetőségei

3.1 A késleltetés meghatározása kétfrekvenciás vevők esetén

Az előző fejezetben meghatároztuk az ionoszféra okozta késleltetés nagyságát a frekvencia és a teljes elektron tartalom függvényében. GNSS rendszereknél a frekvencia ismert, így az egyenletünkben két ismeretlen szerepel. Kétfrekvenciás vevőknél egyszerre két jel érkezik a műholdról közelítőleg ugyanazon az úton, így a kódérés pszeudotávolságai alapján ezekre két független egyenletet tudunk felírni, amiből már meg tudjuk határozni a TEC értéket az adott vonalra az adott időben. A TEC értéket visszaírva a késleltetés képletébe, bármelyik frekvenciára kiszámíthatjuk azt. Legyen R a geometriai távolság, R_I a kisebb frekvencián mért távolság és R_{II} a nagyobb frekvencián mért távolság. Mivel kódérésről van szó, így a mért távolságok nagyobbak lesznek, mint valójában, és a késleltetés pozitív értékeket vesz fel. Ezek után (2.33) alapján felírhatjuk a következő egyenleteket:

$$R_I - R = \frac{40,3}{f_I^2} TEC, \quad (3.1)$$

$$R_{II} - R = \frac{40,3}{f_{II}^2} TEC. \quad (3.2)$$

A (3.1) egyenletből fejezzük ki a valódi távolságot:

$$R = R_I - \frac{40,3}{f_I^2} TEC, \quad (3.3)$$



majd helyettesítsük be a (3.2) egyenletbe:

$$R_{II} - \left(R_I - \frac{40,3}{f_I^2} TEC \right) = \frac{40,3}{f_{II}^2} TEC. \quad (3.4)$$

A képletet átrendezve TEC-re azt kapjuk, hogy:

$$TEC = \frac{(R_{II} - R_I) f_I^2 f_{II}^2}{40,3(f_I^2 - f_{II}^2)}. \quad (3.5)$$

Ezzel megadtuk a TEC értékét két ugyanabban az időpillanatban, de különböző frekvencián végzett pszeudotávolság mérés függvényében. A fenti képlet abban az esetben érvényes, ha a mért távolságok közötti különbség csak a frekvenciák eltéréséből adódik. Gyakorlatban ez nem így van. A műholdak a két különböző frekvenciájú jelet nem ugyanabban az időpillanatban adják ki. Ezt a különbséget nevezzük differenciális kód eltérésnek, röviden DCB-nek (differential code bias). A hazai terminológiában egyszerűen hardver késésnek nevezik ezt a hatást. Ennek a nagysága mérhető, és napi illetve havi átlagban szokták megadni. Ugyanilyen eltérés van a vevőoldalon is. A DCB-k nanoszekundumban vannak megadva, ezért fénysebességgel való szorzással át kell számítani őket hosszúságra. A műhold oldali késleltetést b_{st} -vel, a vevőoldali b_{rc} -vel jelölve a (3.5) egyenlet a következőre módosul:

$$TEC = \frac{(R_{II} - R_I - b_{st} - b_{rc}) f_I^2 f_{II}^2}{40,3(f_I^2 - f_{II}^2)}. \quad (3.6)$$

A DCB-k nagyságrendje $\pm 1-20$ nanoszekundum, tehát az ebből származó késleltetések $\pm 0,3-6$ méter között vannak. Ez kisebb, mint az ionoszféra okozta késleltetés, így azt mondhatjuk, hogy a TEC érték számítására a (3.5) képlet is alkalmazható, csak kevésbé pontos eredmény érhető el vele.

3.2 A késleltetés meghatározása általánosságban

Azokban az esetekben, amikor nem áll rendelkezésünkre információ két külön frekvencián végzett mérésről, a helyzet korántsem ennyire derűs. Vannak ugyan módszerek az



elektron tartalom mérésére, és ezeknek az eredményét közölni is szokták, de maguk a mérések és azok feldolgozása túl sokáig tart ahhoz, hogy valós időben fel lehessen használni. Az egyetlen lehetőségünk az, hogy valamilyen közelítő modell szerint vesszük figyelembe az ionoszférát.

A tökéletes megoldás természetesen az lenne, hogy ismerjük az ionoszféra elektronsűrűségét minden időpontban minden helyen. Ilyenkor a műhold és a vevő közötti vonal mentén az út szerint integrálva az elektronsűrűséget megkapjuk a keresett TEC értéket, innen pedig már számítható lenne a késleltetés. Sajnos ez csupán fikció.

Gyakori megoldás, hogy a teljes ionoszféra tartalmát egy a Föld felszíne felett 350 km-rel levő gömbfelületre vetítve képzeljük el, a felület elektron tartalma pedig a helyi vertikális elektrontartalomnak felel meg. Ez a modell csupán zenitirányú műholdak esetén ad jó megoldást. Egyéb esetekben egy ferdeségi szorzóval módosítják a TVEC értéket. Az így kapott eredmény már közelebb áll a valósághoz, de nem veszi figyelembe azt, hogy az elektronsűrűség vertikális és horizontális irányban nem állandó, így még mindig pontatlan.

Ennél nem sokkal számításigényesebb a Klobuchar-féle ionoszféramodell. Itt az éjszakai órákban egy konstans zenitirányú késleltetést feltételezünk, napközben pedig egy koszinusz függvénnyel adjuk meg azt. A függvény paraméterei a helyi idő, illetve a geomágneses szélesség, mivel a légkör ionizáltsága nagymértékben függ attól, hogy a napsugarak milyen szögben érkeznek be, és mennyi ideig érik az adott légköri réteget. Mivel itt is zenitirányú eredményt kapunk, ezt is módosítani kell egy ferdeségi szorzótényezővel.

Ezektől eltérő megközelítést alkalmaz a NeQuick modell. Havi medián elektronsűrűség térképeket felhasználva végez számításokat, ennek köszönhetően figyelembe tudja venni a napfolttevékenységet is. A Galileo rendszer használja ezt a modellt. A műholdak a navigációs üzenetekben küldik a szükséges paramétereket, a vevő pedig ismeri az elektronsűrűség eloszlások havi medián térképeit, és ezekből számítja a késleltetést.

Az eddigi felsorolásból látszik, hogy az ionoszféra késleltetésének hatékony, és megfelelő pontosságú modellezése egy kihívásokkal teli kutatási terület. Több kísérlet és módszer volt már arra, hogy az ionoszféra állapotát tomografikus úton határozzák meg. Ez annyit jelent, hogy az ionoszférán áthaladó műholdjelek terjedési idejéből megpróbáljuk feltérképezni annak fizikai állapotát, elsősorban az elektronsűrűséget.

A [3] forrásban egy olyan modellt mutatnak be, ahol az elektronsűrűséget gömbfüggvényekkel írják le. A gömbfüggvényt integrálva a műhold és a vevő közötti úton meghatározható a TEC érték. A másik oldalról tekintve a TEC értékek meghatározhatóak kétfrekvenciás mérések alapján, és így a két összefüggésből egy egyenletrendszerhez jutunk,



amelyben a gömbfüggvény együttható az ismeretlen paraméterek, és ezeknek az értéke valamilyen kiegyenlítéssel meghatározható.

Más megközelítést alkalmaznak az [4] forrásban. Az ionoszférát cellákra osztják, és egy adott cellában konstans elektronsűrűséget feltételeznek. Ebben a felbontásban igaznak kell lennie annak, hogy a műhold és a vevő közötti ferde irányú elektrontartalom megegyezik azzal, hogy szummázzuk a műhold-vevő trajektória egyes cellákba eső szakasz hosszainak és a cellák elektronsűrűségeinek szorzatát. Ezen összefüggés alapján egy algebrai rekonstrukciós módszerrel állítják elő az elektronsűrűséget.

Mi úgy döntöttünk, hogy a kettő módszert ötvözzük, és egy olyan közel valós idejű modellt alkotunk, ahol az ionoszférát cellákra osztjuk, az egyes cellákban konstans elektronsűrűséget feltételezünk, és kétfrekvenciás mérések alapján kiegyenlítéssel, egészen pontosan a legkisebb négyzetek módszerével határozzuk meg a cellák elektronsűrűségét. A közel valós idejű alatt azt értem, hogy a modell elkészítéséhez nem kell több idő, mint ameddig az érvényesnek tekinthető. A modellt a következő fejezetben ismertetem.

4. A cellaalapú ionoszféra modell

4.1 A számítási modell

A modell kitalálása során az volt az elsődleges szempont, hogy kis lépésekkel haladjunk, minél egyszerűbb legyen, és majd ha az egyszerűbb modell működik, akkor lehet tovább finomítani. Ezért döntöttünk amellett, hogy azt a megközelítést használjuk, amiben az ionoszférát cellákra osztjuk, és egy cellában konstans elektronsűrűséget feltételezünk. A számítások során permanens állomások kétfrekvenciás méréseiből indulunk ki. Ezekből a mérésekből felírhatjuk a teljes elektron tartalmat a (3.6) összefüggés alapján. Ez a teljes elektron tartalom felírható a műhold-vevő vektor által metszett cellák adott vonal mentén vett teljes elektron tartalmainak összegeiként:

$$\sum_i TEC_i = \frac{(R_{II} - R_I - b_{st} - b_{rc}) f_I^2 f_{II}^2}{40,3(f_I^2 - f_{II}^2)}. \quad (4.1)$$

A teljes elektrontartalom felírható az elektronsűrűség vonal menti integráljaként:

$$\sum_i \int N_{ei} ds = \frac{(R_{II} - R_I - b_{st} - b_{rc}) f_I^2 f_{II}^2}{40,3(f_I^2 - f_{II}^2)}, \quad (4.2)$$

ahol az integrálást az adott cellában található vektorszakasz mentén kell elvégezni. Mivel a modell szerint az elektronsűrűség cellánként állandó, így ez tovább egyszerűsíthető:

$$\sum_i N_{ei} S_i = \frac{(R_{II} - R_I - b_{st} - b_{rc}) f_I^2 f_{II}^2}{40,3(f_I^2 - f_{II}^2)}. \quad (4.3)$$

A mérési eredmények helyett áttérünk a kettő különbségére, és ezt tekintjük mérési eredménynek, ez a számítás eredményét nem befolyásolja döntően, de egyszerűsíti a menetet. Az egyenletet nullára rendezve megkapjuk a modellünk alapegyenletét:



$$\frac{40,3(f_I^2 - f_{II}^2)}{f_I^2 f_{II}^2} \sum_i N_{ei} s_i - \Delta R + b_{st} + b_{rc} = 0. \quad (4.4)$$

Ha van m darab mérési eredményünk, akkor m darab ilyen egyenletet írhatunk fel. Ha az ismeretlen paraméterek számát k -val jelöljük, akkor legalább k darab független egyenletre van szükségünk arra, hogy a kapott egyenletrendszer megoldható legyen legkisebb négyzetek módszerével II. kiegyenlítési csoportot használva. Az erre vonatkozó közvetítő egyenletek általános alakja a következő formát ölti:

$$\underline{v} = \underline{A}\underline{x} - (\underline{L} - \underline{A}\underline{X}_0 - \underline{C}), \quad (4.5)$$

ahol $\underline{v}$ a mérési eredmények ismeretlen javításait tartalmazó vektor, $\underline{x}$ az ismeretlen paramétereket javításait tartalmazó vektor, $\underline{A}$ az alakmátrix, $\underline{L}$ a mérési eredményeket tartalmazó vektor, $\underline{X}_0$ a paraméterek előzetes értékeit tartalmazó vektor, $\underline{C}$ pedig az egyenletekben található többi konstans. A zárójelben szereplő kifejezést szokás összevonni, $\underline{l}$ betűvel jelölni, és tisztatag vektornak hívják.

A mi esetünkben az ismeretlen paraméterek száma, és ezzel együtt a minimálisan szükséges független egyenleteké a rendelkezésre álló adatoktól függ. Az elektronsűrűségek mindig ismeretlenek, míg a DCB értékekről lehet információnk, így a következő 4 lehetőséget kell, hogy megkülönböztessük. Mindkét DCB ismeretlen, a műhold DCB ismeretlen, az állomás DCB ismeretlen és mindkét DCB ismert. Ennek megfelelően rendre a következő 4 egyenletet írhatjuk fel:

$$\frac{40,3(f_I^2 - f_{II}^2)}{f_I^2 f_{II}^2} \sum_i n_{ei} s_i + b_{st} + b_{rc} - \left(\Delta R - \frac{40,3(f_I^2 - f_{II}^2)}{f_I^2 f_{II}^2} \sum_i N_{ei0} s_i - B_{st0} - B_{rc0} \right) = r, \quad (4.6)$$

$$\frac{40,3(f_I^2 - f_{II}^2)}{f_I^2 f_{II}^2} \sum_i n_{ei} s_i + b_{st} - \left(\Delta R - \frac{40,3(f_I^2 - f_{II}^2)}{f_I^2 f_{II}^2} \sum_i N_{ei0} s_i - B_{st0} - B_{rc} \right) = r, \quad (4.7)$$

$$\frac{40,3(f_I^2 - f_{II}^2)}{f_I^2 f_{II}^2} \sum_i n_{ei} s_i + b_{rc} - \left(\Delta R - \frac{40,3(f_I^2 - f_{II}^2)}{f_I^2 f_{II}^2} \sum_i N_{ei0} s_i - B_{st} - B_{rc0} \right) = r, \quad (4.8)$$

$$\frac{40,3(f_I^2 - f_{II}^2)}{f_I^2 f_{II}^2} \sum_i n_{ei} s_i - \left(\Delta R - \frac{40,3(f_I^2 - f_{II}^2)}{f_I^2 f_{II}^2} \sum_i N_{ei0} s_i - B_{st} - B_{rc} \right) = r. \quad (4.9)$$

A fenti képletekben a nullás indexek jelölik az előzetes értékeket a tisztatag vektorban, és az ezeknek megfelelő kisbetűk pedig az adott paraméterhez tartozó javításokat, ΔR a mérési eredményt, r pedig az ehhez tartozó mérési javítást. Minden, ami más, az előre ismert konstans. Az egyenletrendszerünk felírása után már meghatározható az alakmátrix, és a tisztatag vektor, ezek ismeretében pedig már elvégezhető a kiegyenlítés. A paraméterek javításának a képletét, és a kiegyenlített paraméter értékeket csak általános alakban írom fel itt:

$$\underline{\underline{x}} = \underline{\underline{N}}^{-1} \underline{\underline{n}} = (\underline{\underline{A}}^* \underline{\underline{P}}_{LL} \underline{\underline{A}})^{-1} (\underline{\underline{A}}^* \underline{\underline{P}}_{LL} \underline{\underline{l}}), \quad (4.10)$$

$$\underline{\underline{X}} = \underline{\underline{X}}_0 + \underline{\underline{x}}. \quad (4.11)$$

A képletben szereplő P_{LL} a mérésekre vonatkozó súlymátrix. Mivel azzal a feltételezéssel éltünk, hogy minden mérés egyszeri, független, és ugyanolyan módszerrel lett végrehajtva, ez egységmátrixnak vehető ebben az esetben. Ugyanakkor be lehetne vezetni egy magassági szögtől függő súlyozást, mivel a horizonthoz közelebb műholdaknál nagyobb lehet a több utas terjedés hatása. Az N betűvel jelölt mátrixot szokás normálmátrixnak is nevezni. Látható, hogy a számítás során ezt a mátrixot invertálni kell, azonban előfordulhat, hogy szinguláris lesz. Ebben az esetben valamilyen általánosított inverz (pl. Moore-Penrose féle pszeudinverz) felhasználásával juthatunk eredményre. Ezt követően a paraméterek javítási értékeit felhasználva meg tudjuk határozni a mérési javításokat is, illetve a javított méréseket:

$$\underline{\underline{v}} = \underline{\underline{A}} \underline{\underline{x}} - \underline{\underline{l}}, \quad (4.12)$$

$$\underline{\underline{U}} = \underline{\underline{L}} + \underline{\underline{v}}. \quad (4.13)$$

Amennyiben a kapott javítások egy általunk megválasztott szintet meghaladnak, úgy a kiegyenlítést újra el kell végezni, úgy hogy most az előzőben kapott kiegyenlített paraméter értékek lesznek az előzetes paraméter értékek. A számítás konvergens, így az iterációt addig



kell folytatni, amíg a változások elég kicsik nem lesznek. A mi esetünkben jó eséllyel szükség is lesz több iterációra, mivel nincs információnk az előzetes értékekről, és legjobb esetben is csak nagyságrendileg tudjuk felvenni őket.

A kiegyenlítést követően az eredmény egy cellamodell, amiben minden cellára van egy ismert elektronsűrűségünk. Ezt a teljes egészében a modellen áthaladó műhold-vektoroknál tudjuk felhasználni hibakorrekcióra, azaz az ionoszférikus késleltetéstől mentes pszeudotávolság visszaállítására. Az adott vektornak meghatározzuk, hogy melyik cellákba mekkora szakasza esik. Ezután az elektronsűrűségek ismeretében meg tudjuk határozni a cellánkénti TEC értéket, majd ezeket szummázva megkapjuk az egész vektorra vonatkozó elektron tartalmat. Végül pedig (2.33) alapján kiszámítjuk a kívánt frekvencián a késleltetést, és levonjuk a megfelelő pszeudotávolságból.

A modell az ionosféra egy pillanatnyi állapotát tükrözi, viszont az ionosféra gyorsan változik, ezért a felhasználás időtartam korlátos. Ebből következik, hogy a modell valós idejű használhatóságának érdekében szükséges az, hogy jóval rövidebb ideig tartson elkészíteni azt, mint amennyi ideig érvényes. Az érvényességi idő növelhető, ha nem tekintjük időben állandónak az adatokat, hanem mondjuk az előző pár modell alapján extrapolálunk valamit. Hasonló módon növelhető a pontosság, ha térben sem tekintjük állandónak a modellt, hanem az egyes cellák között interpoláljuk az elektron sűrűséget. Ezekkel a lehetőségekkel már nem foglalkoztam, hiszen mint írtam, az volt az elsődleges szempont, hogy kis lépésekkel haladjunk.

4.2 Az ionosféra felosztása cellákra

A számítási modell után a következő fontos dolog, amit át kell gondolni, hogy milyen módon osszuk fel az ionoszférát cellákra. Ezzel kapcsolatban sok kérdés felmerül. Lokális vagy globális modellt akarunk? Amennyiben lokális, úgy akarjuk-e közelíteni a Föld alakját síkkal? Milyen alakúak legyenek a cellák? Milyen méretűek legyen a cellák, illetve ami ezzel összefügg, hogy hány darab cella legyen? Ezen kérdések eldöntésénél is az elsődleges szempont az egyszerű megvalósíthatóság volt.

Mivel előfordulhat, hogy az elsőre elkészített cellamodell nem megfelelő, így mindenképpen valamilyen paraméterezhető megoldást akartunk elkészíteni. Ezzel gyakorlatilag eldöntöttük, hogy egy lokális modell is tetszőlegesen skálázható, akár elvihető globális szintig is. Az egyetlen dolog, amire figyelni kell, hogy a globális szint elérésénél az



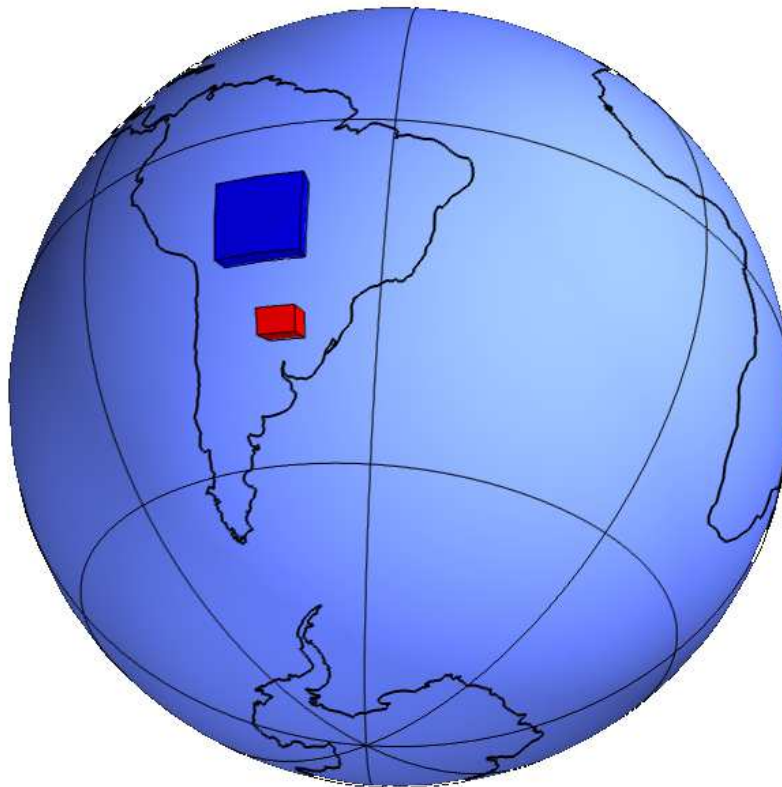
ionoszféra minden térrészletét csak 1 cellával fedjük le. Ezek után már csak a síkkal való közelítés lehetősége csábít a szigorúan lokális modell felé. Végiggondoltuk a lehetőségeket, és arra jutottunk, hogy többet ér a jóval nagyobb szabadság egy globális méretű modell esetén, mint amennyivel egyszerűbb lenne sík alapfelülettel számolni valamilyen görbe helyett. Végül arra jutottunk, hogy modellünket úgy vesszük fel, hogy két ellipszoid határolja alulról és felülről, nagyjából az ionoszférea alsó és felső szélénél. Ezt a két ellipszoidot úgy vesszük fel, hogy a középpontja egybeesik a WGS84 ellipszoidéval, az nagytengelyük hossza pedig az alsó illetve felső ionoszféra határok geocentrumtól vett közelítő távolsága. Az excentricitások szintén megegyeznek a WGS84-ével. Azért a WGS84-et választottuk, mert ez a GNSS rendszerek alapfelülete, és azért az excentricitás egyezik meg, mert számításokban főként ezt használjuk, nem pedig a lapultságot. Horizontális irányban a modellünket a geocentrumon átmenő síkokkal határoljuk le.

A cellák alakjának kiválasztásánál azt kellett figyelembe venni, hogy milyen irányokban változik az ionoszféra állapota, hiszen ezekben az irányokban kell felosztani részekre. A „fő” változások a szélességi és hosszúsági körök mentén illetve vertikális irányban vannak. Tehát az a legpraktikusabb választás, ha vertikálisan koncentrikus azonos excentricitású ellipszoidokkal osztjuk fel az ionoszférát rétegekre, horizontálisan pedig a meridián síkok valamint a Föld középpontja és a szélességi körök által meghatározott kúp palástok mentén darabolunk. Az így kapott térelemek tesszeroidok lesznek. Ezeket szemlélteti általánosságban a 4.1-es ábra.

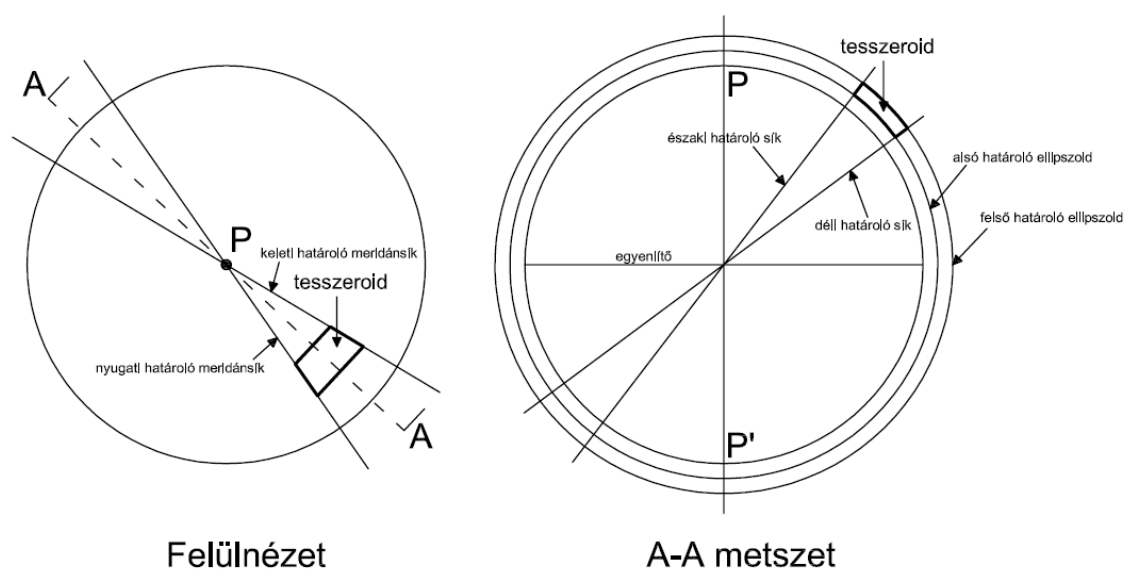
Egy ilyen általános tesszeroidot kezelni nem annyira egyszerű, ezért mi kicsit eltérünk ettől és közelítést alkalmazunk. Egészen pontosan a vertikális határoló elemek a mi esetünkben is koncentrikus ellipszoidok azonos excentricitással, illetve az észak-dél irányú határoló elemek is a meridián síkok lesznek, az eltérés a kelet-nyugat irányú határoló felületekben van. Ezek egy kúppalást helyett olyan síkok lesznek, amelyek átmennek a Föld középpontján illetve a WGS84 ellipszoid és a két meridián sík által meghatározott meridián kívánt szélességű pontjain. Ezt szemlélteti a 4.2 ábra két vetületből. Az eddig leírtakból látszik az is, hogy a cellák méretét egy szélességi szög különbséggel, egy hosszúsági szög különbséggel és (valamelyik tengely irányú) rétegvastagsággal tudjuk megadni.

Ezzel a cellatípussal viszonylag egyszerű számításokat végezni, ugyanakkor több szempontból is nagy pontatlanságot tud okozni. Az egyik, hogy ha túl szélesre választjuk a cellákat, akkor a szélességek mentén való felosztás már nagyon eltávolodik szélességi köröktől, és inkább a szélességi körök húrjai mentén lesz. Szerencsére ez nem gond, mert az ilyen nagyméretű cellákból álló modellel amúgy sem tudnánk mit kezdeni, kisebb

cellaméretek esetén pedig a kelet-nyugat irányú határoló felületek egészen jól közelítik a szélességi köröket.



4.1. ábra: Tesszeroidok a Föld felszínén



4.2. ábra: A tesszeroidok határoló felületei



A másik pontatlanság, ami keletkezik az, hogy a cellák méreteit szélességi és hosszúsági szög különbséggel adjuk meg, viszont a hosszúsági szög valódi geometriai hossza függ a tényleges szélesség értékétől is, így két ugyanolyan szögméretekkal és vastagsággal rendelkező cella közül egy egyenlítő közelinek a valódi térfogata jóval nagyobb lesz, mint egy sarkvidékinek, így az egyenlítő környékén jóval pontatlanabb lesz a modell. Szerencsére minket ez sem zavar, mert a sarkvidéki cellák amúgy sem számítanak annyira, mivel a műholdak zenit irányban nem érnek el akkora szélességeket. Ettől még lehetnek olyan horizonthoz közeli műhold-vevő vektorok, amelyek a sarkvidék felett haladnak át, de elenyésző mennyiségben.

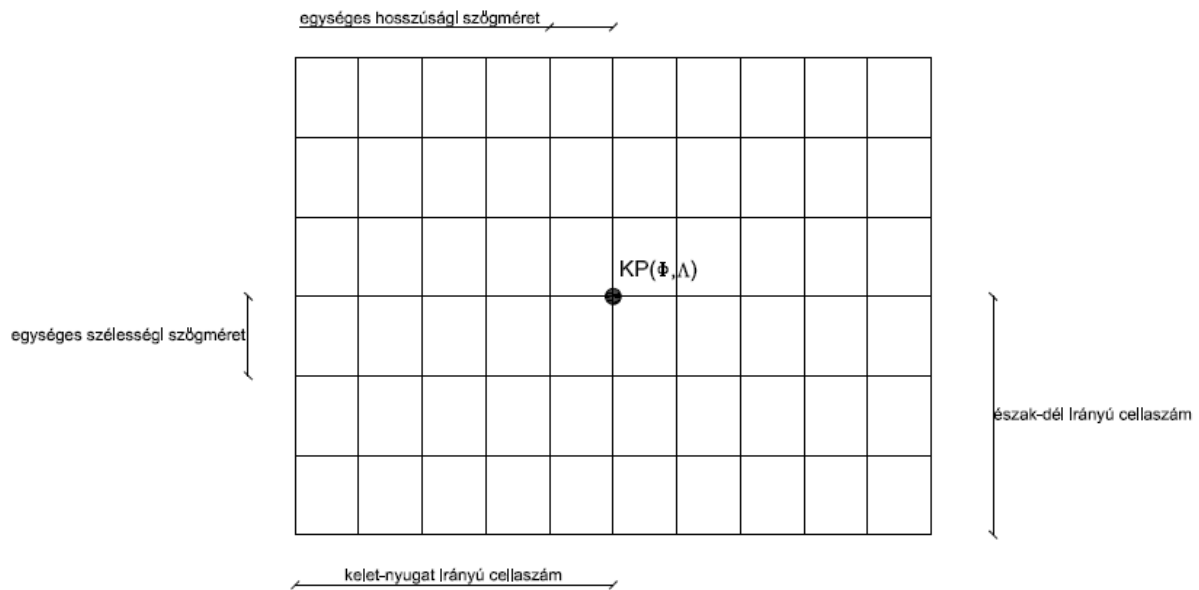
A harmadik pontatlanság teljesen hasonló az előzőhöz, és abból fakad, hogy a szélesség és a hosszúság valódi geometriai mérete függ a geocentrumtól való távolságtól is, így két ugyanolyan szögméretekkal és vastagsággal rendelkező cella közül, a magasabban elhelyezkedőnek nagyobb lesz a valódi térfogata, így nagyobb magasságban pontatlanabb lesz a modell. Ezen viszont könnyedén tudunk segíteni eltérő rétegvastagságok használatával. Ugyanakkor ez a különbség arányaiban jóval kisebb az előzőnél, így figyelmen kívül is hagyható.

Végül foglalkozunk a cellaméretekkel. Ezt alapvetően két dolog határozza meg. Az egyik a modell mérete, a másik az a modellen belüli állomások száma, amelyekről van mérési adatunk. Ha fix méretű cellákkal dolgozunk, akkor egy nagyobb modellben több cellára van szükségünk. A több cella több ismeretlent jelent, tehát nagyobb alakmátrixot, és ez által nagyobb számítási igényt. Márpedig ez egy jelentős korlát. Ennek a vizsgálatával később részletesen foglalkozok, itt csak annyit jegyeznek, hogy ezek szerint minél nagyobb a modell, annál nagyobb cellaméretekre van szükség, ezáltal csökkenni fog a pontosság is, vagy fordítva minél kisebbek a cellák, annál kisebb lehet a modell maximális mérete. A másik szempont az állomások száma. Mint azt már az előző fejezetben említettem, legalább annyi független egyenletre, tehát független mérésre, van szükség, mint ahány paraméterünk, azaz cellánk van. A mérések száma függ a modellen belül található állomások számától, így ez egy felső korlátot ad a modell celláinak lehetséges számára, és így egy alsó korlátot a modell celláinak méretére.

Az eddigi megfontolások alapján döntöttünk úgy, hogy a modellt a következő módon fogjuk felállítani. Először is szükség van egy fix pontra az alapfelületen, ezt hosszúsággal és szélességgel adjuk meg, ezt lesz a modellünk közepe. Maga a modell „kiterítve” egy téglatest lesz felosztva sok kisebb téglatestre, és a középpont az alsó lap közepén fog elhelyezkedni. Ezután megadjuk az egyes cellák hosszúsági és szélességi szögméreteit. Ez minden cellára



ugyanakkora lesz. Ezt követően megadjuk a szükséges rétegek számát, vertikálisan ennyi egyenlő vastagságú rétegre fogjuk felosztani modellt. Ezekkel az adatokkal már egyértelműen meg van adva a cellák szögmérete. Ami kérdés még, az a modell mérete. Ezt pedig úgy adjuk meg, hogy a középponttól nézve megmondjuk, hány darab cella legyen kelet-nyugat, illetve észak-dél irányban. Ezeknek az adatoknak a segítségével már meghatározható az összes cella összes sarokpontja, illetve a határoló felületek, tehát a modell adottnak vehető. A bekezdésben leírtakat szemlélteti a 4.3-as ábra, ahol a modell egy rétege látható.



4.3. ábra: A modell szélességi és hosszúsági paramétere

Ennek a modellnek előnyei a számítógéppel való könnyű kezelhetőségben rejlenek. Magukat a cellákat egy háromdimenziós mátrix elemeiként tudjuk tárolni, viszonylag kevés adat megadásával könnyedén tudunk, egy egészen rugalmas modellt felépíteni, és könnyű a szomszédsági viszonyokat szemléltetni a maradékosztályok segítségével, ez pedig elengedhetetlen a műhold-vevő vektorok egyes cellákba eső részeinek hatékony számításához.



5. A modell számítógépes megvalósítása

5.1 A modell megvalósításának áttekintése

A modellt megvalósító programot C nyelven írtam meg Linux operációs rendszerre, mivel a legtöbb szakmailag releváns szoftver is így készült. Több modult készítettem az egyes elvégzendő feladatok szerint. Az egyes modulokat osztott könyvtárakként fordítottam, kivéve a fő programot, ennek köszönhetően azok könnyedén újrafelhasználhatóak más programokban is. A három fő rész:

- RINEX parser: A méréseket tartalmazó fájlok feldolgozását végzi.
- Grid model: A cellalapú modell előállítását és az azzal kapcsolatos számításokat végzi.
- Ionosphere modeler: Ez a főprogram. Létrehozza a kívánt modellt, beolvas mérési eredményeket, és ezek alapján elvégez egy kiegyenlítést.

A számítások végénél használtam még Matlabot is, mivel jóval egyszerűbb benne mátrix műveleteket implementálni. A fent leírtakon kívül a két osztott könyvtárhoz készítettem még tesztprogramokat is, hogy a működésük helyességét ellenőrizni tudjam.

5.2 A RINEX formátum áttekintése

Többször említettem már, hogy a modell működése kétfrekvenciás vevők méréseinek feldolgozásain alapul, ezért a legelső feladat, amit meg kell oldani, hogy hozzájussunk ezekhez, illetve be tudjuk olvasni. Létezik több GNSS állomás hálózat például a GNSSnet.hu vagy az EPN (EUREF Permanent Network). Ezek fix, állandó állomásokból állnak, amelyek folyamatosan végeznek méréseket, akár több frekvencián is több műhold rendszert használva. Ezeket a méréseket elérhetővé teszik utólagosan, illetve valós időben is. Ami számunkra lényeges, hogy milyen formában lehet hozzáférni az adatokhoz. A GNSS mérések leírására a



RINEX (Receiver Independent Exchange Format) fájl formátum használatos. Ez egy szöveges formátum.

A RINEX formátumnak több verziója is van, folyamatosan fejlesztik, de az állomások nem mindig a legújabbat használják. Ennek valószínűleg az oka, hogy a régebbi verzió is megfelel a célokra, és ha tovább lépnek az újabbra, akkor a felhasználóknak, is újítani kell a feldolgozó szoftveren. Nekünk 2.11-es verziójú EUREF hálózati adatok álltak rendelkezésünkre. Ennek megfelelően egy olyan feldolgozó programra volt szükség, ami a 2.11-es RINEX fájlokat dolgoz fel. Mivel ez egy globálisan használt szabvány, így az első gondolatunk rögtön az volt, hogy keresünk az interneten egy programot, ami ezt csinálja. Találtunk is, de sajnos csak C++-ban illetve más összetettebb nyelveken írt programokat, C-ben írtat sajnos nem. Lehetett volna azt is csinálni, hogy letöltjük a C++-osat, és írunk rá egy C-s wrappert, de általában ebben az irányban jóval nehezebb ez a feladat, mint az egyszerűbb nyelvtől a bonyolultabb felé. Így született az a döntés, hogy írunk egy saját feldolgozó programot. Röviden áttekinteném a 2.11-es verziójú RINEX állományok formátumának fontosabb elemeit. Maga a pontos specifikáció elérhető az [5] webcímen.

A RINEX fájloknak több típusuk van, a 2.11-es verzióban konkrétan 7:

- Observation Data
- Navigation Message
- Meteorological Data
- GLONASS Navigation Message
- GEO Navigation Message
- Satellite and Receiver Clock Date
- SBAS Broadcast Data

Az egyes fájl típusokra elnevezési konvenció is van, mégpedig a következő:

ssssdddf.yyt

ssss – 4 betűs állomásazonosító

ddd – az első mérés napjának sorszáma az évben

f – több fájlból álló méréssorozatok esetén a fájl sorszámának azonosítója

yy – az évszám utolsó két jegye

t – a megfelelő fájltypus azonosítója

Mi egyelőre, továbbra is a kis lépések elvet követve, csupán GPS adatok feldolgozásával foglalkoztunk, így csak az első három fájltypus feldolgozását készítettük el, azt is kicsit



szűkítve, mivel az observation fájlok közösek a műholdrendszerekre, így egyelőre csak olyan fájlok feldolgozása lehetséges, amikben csupán GPS és GLONASS műholdak találhatók. Valójában még ezzel is erősen túllóttunk a célon, mert a tényleges adat, amire szükségünk van, csak az observation fájlokban található meg, és elenyésző része az egésznek.

A különböző típusú fájlok szerkezete egészen hasonló, ezért a teljesség igénye nélkül itt csak a mérési fájlokat mutatom be egy példán. A többi fájl pontos formátuma megtalálható a specifikációban. Az 5.1-es ábrán látható egy observation fájl első néhány sora. Ez a példa a specifikációból lett kimásolva, és lehet nevezni állatorvosi lónak, mert szinte minden speciális eset megtalálható rajta, és a valósághoz szinte semmi köze, gyakorlatban alig fordulhat elő ilyen. Az első, ami feltűnhet, hogy minden sor maximum 80 karakterből állhat. Ha megvizsgáljuk a fájlt látható, hogy két részből áll, egy fejléc és egy adat részből. A fejléct onnan lehet felismerni, hogy a fájl elején kezdődik, a sorok utolsó 20 karakterében mindig van valami, és a végét egy „END OF HEADER” címke jelzi.

A fejléc meta adatokat tartalmaz a méréseinkről. Az utolsó 20 karakter mindig egy címke, ami azt hivatott jelezni, hogy az előtte levő 60 karakter mire vonatkozó adatokat tartalmaz milyen formátumban. Itt mondanám el, hogy a fájl formátuma nagyon kötött (nem csak a fejlécben, hanem az adat részben is), minden egyes adatnak pontosan megvan a helye, megvan az, hogy hányadik karaktertől hányadik karakterig tarthat, sőt még az is kötött, hogy milyen adat típus van ott. Tehát például, ha a fejlécben van egy „TIME OF FIRST OBS” címke, akkor tudjuk azt, hogy az első 6 karakteren egy integer típusú szám fog állni, vagy hogy a 31. karaktertől a 43. karakterig egy float típusú szám lesz. Az ilyen szintű kötöttség nagyban segíti a feldolgozást, mivel nem kell az adott sort felbontani elemekre, és az elemeket külön-külön vizsgálni, hogy vajon ez mi lehet, hanem rögtön lehet, azt mondani, hogy ettől a karaktertől eddig a karakterig az adat kerüljön bele ebbe a változóba.



```

-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     TABLE A7                                     |
|                               GPS OBSERVATION DATA FILE - EXAMPLE                               |
|-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23|24|25|26|27|28|29|30|
|-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 2.11          OBSERVATION DATA      M (MIXED)          RINEX VERSION / TYPE |
| BLANK OR G = GPS, R = GLONASS, E = GALILEO, M = MIXED COMMENT |
| XXRINEXO V9.9  AIUB                  24-MAR-01 14:43 PGM / RUN BY / DATE |
| EXAMPLE OF A MIXED RINEX FILE (NO FEATURES OF V 2.11) COMMENT |
| A 9080 MARKER NAME |
| 9080.1.34 MARKER NUMBER |
| BILL SMITH ABC INSTITUTE OBSERVER / AGENCY |
| X1234A123 XX ZZZ REC # / TYPE / VERS |
| 234 YY ANT # / TYPE |
| 4375274. 587466. 4589095. APPROX POSITION XYZ |
| .9030 .0000 .0000 ANTENNA: DELTA H/E/N |
| 1 1 WAVELENGTH FACT L1/2 |
| 1 2 6 G14 G15 G16 G17 G18 G19 WAVELENGTH FACT L1/2 |
| 0 RCV CLOCK OFFS APPL |
| 5 P1 L1 L2 P2 L5 # / TYPES OF OBSERV |
| 18.000 INTERVAL |
| 2005 3 24 13 10 36.00000000 TIME OF FIRST OBS |
| 05 3 24 13 10 36.00000000 0 4G12G09G06E11 END OF HEADER |
| 23629347.915 .300 8 -.353 23629364.158 -.123456789 |
| 20891534.648 -.120 9 -.358 20891541.292 |
| 20607600.189 -.430 9 .394 20607605.848 |
| .324 8 .178 7 |
| 05 3 24 13 10 50.00000000 4 4 |
| 1 2 2 G 9 G12 WAVELENGTH FACT L1/2 |
| *** WAVELENGTH FACTOR CHANGED FOR 2 SATELLITES *** COMMENT |
| NOW 8 SATELLITES HAVE WL FACT 1 AND 2! COMMENT |
| 05 3 24 13 10 54.00000000 0 6G12G09G06R21R22E11 COMMENT |
| 23619095.450 -53875.632 8 -41981.375 23619112.008 -.123456789 |
| 20886075.667 -28688.027 9 -22354.535 20886082.101 |
| 20611072.689 18247.789 9 14219.770 20611078.410 |
| 21345678.576 12345.567 5 |
| 22123456.789 23456.789 5 |
| 65432.123 5 48861.586 7 |
| 05 3 24 13 11 0.00000000 2 1 |
| *** FROM NOW ON KINEMATIC DATA! *** COMMENT |
| 05 3 24 13 11 48.00000000 0 4G16G12G09G06 -.123456789 |
| 21110991.756 16119.980 7 12560.510 21110998.441 |
| 23588424.398 -215050.557 6 -167571.734 23588439.570 |
| 20869878.790 -113803.187 8 -88677.926 20869884.938 |
| 20621643.727 73797.462 7 57505.177 20621649.276 |
| 3 4 |
| A 9080 MARKER NAME |
| 9080.1.34 MARKER NUMBER |
| .9030 .0000 .0000 ANTENNA: DELTA H/E/N |
| --> THIS IS THE START OF A NEW SITE <-- COMMENT |
| 05 3 24 13 12 6.00000000 0 4G16G12G06G09 -.123456987 |
| 21112589.384 24515.877 6 19102.763 3 21112596.187 |
| 23578228.338 -268624.234 7 -209317.284 4 23578244.398 |
| 20625218.088 92581.207 7 72141.846 4 20625223.795 |

```

5.1. ábra: RINEX mérési fájl

Tekintsük át a fontosabb címkéit a fejlécnek:

RINEX VERSION / TYPE

információ a RINEX verzióról

illetve a fájlban található műholdak
típusáról

PGM / RUN BY / DATE

az előállító program, a program



COMMENT	futtatója, a futtatás dátuma megjegyzés
MARKER NAME	az állomás neve
MARKER NUMBER	az antennajelző sorszáma
OBSERVER / AGENCY	az észlelést végző személy illetve szervezet neve
REC # / TYPE / VERS	a vevő sorszáma, típusa és verziója
ANT # / TYPE	az antenna sorszáma és típusa
APPROX POSITION XYZ	a jelző közelítő pozíciója WGS84 koordinátákban
ANTENNA: DELTA H/E/N	az antenna külpontossága a ponthoz képest (antennamagasság, és keleti északi külpontosság)
# / TYPES OF OBSERV	a különböző észlelések száma illetve azok típusai
TIME OF FIRST OBS	az első észlelés időpontja

Ezek után térjünk rá az adat részre. Minden egyes adat bejegyzés egy felvezető sorral kezdődik. Ez a felvezető sor alapvetően két féle lehet. Az egyik lehetőség, hogy észlelések következnek. Ilyenkor a felvezető sor először az észlelés epocháját tartalmazza. Ezután van egy flag, ami azt jelzi, hogy volt-e valami valamilyen üzemszünet az előző mérés óta. Ezt követi az, hogy az adott epochában hány darab műholdra volt észlelést, majd a műholdak kerülnek felsorolásra az azonosítójukkal. Előfordulhat, hogy a műholdak nem férnek ki egy sorba, ilyenkor a következő sorba folytatódik a felsorolás. A felvezető soroknál a másik lehetőség, hogy valamilyen esemény történt, például elmozgatták az antennát, vagy kimaradt egy epocha a periodikus mérésekből (ciklusugrás volt), és ezt jelzik. Ilyenkor is lehet epocha, de nem feltétlenül szükséges. Ami biztosan szerepel a sorban, az esemény kódja, illetve a műholdak számának a helyén az, hogy hány darab speciális sor következik. Ezek a sorok lehetnek commentek, vagy más meta adat, a fejlécnek megfelelően kell őket feldolgozni.

Amennyiben a felvezető sor szerint észlelés következnek, úgy a következő sorokban minden egyes műholdra számadatok következnek. Az, hogy melyik számadat milyen mérést is takar, a fejléc „# / TYPES OF OBSERV” címkéjéből derül ki. Nagyjából ezek a legfontosabb jellemzői az observation fájloknak. Ezekből nekünk a modell számítása során az egész adatmennyiség töredékére van tényleges szükségünk, így egy jóval egyszerűbb logikával is meg lehetett volna oldani a feladatot. Egészen pontosan a fejlécből csak a „# /



TYPES OF OBSERV” címkére van szükség, az észlelésekből pedig az epochákra, a műhold azonosítókra, illetve a megfelelő mérésekre. Ennek ellenére a kód feldolgozza az egész fájlt, így a modul használható más feladatokra is.

5.3 A RINEX parser modul megvalósítása

Ebben a pontban a modul programozás technikai részleteit ismertetném. Az adatok feldolgozásánál azt érdemes először megvizsgálni, hogy milyen módon kívánjuk tárolni az észleléseket. Az előző pontban láttuk, hogy a mérési eredményeket az epochával, a műhold azonosítójával, és a műhold mérések sorrendjében vett pozícióval tudjuk egyértelműen megadni. A tárolást is ennek megfelelően kell implementálni. Sajnos a C nyelvben nem állnak rendelkezésre generikus konténer osztályok, így két lehetőség áll előttünk. Vagy írunk egy specifikus konténer osztályt, vagy valamilyen kevésbé optimális, de egyszerűbb tárolási módszert alkalmazunk. Az utóbbi mellett döntöttünk. Az adatokat egy kétdimenziós struktúrában tároljuk. Az első dimenzió egy fix méretű tömb, aminek az elemeit a műholdak sorszámaival lehet indexelni. Minden műhold rendszernek hagytunk 100 műholdnyi helyet, ha esetleg időközben bővítenék a konstellációt. 0-99-ig vannak a GPS műholdak, 100-199-ig a GLONASS műholdak, és mivel több rendszerrel nem foglalkozunk, így egyelőre ennyi van összesen. A tömb elemei dinamikus tömbök, melynek az indexei az adott műholdhoz tartozó epochák növekvő sorrendben. A tömb mindig pontosan akkora, mint amekkorának lennie kell, és az elemek epochák szerint növekvő sorrendben vannak. Maguk az epochák egy *long*-ként vannak tárolva, mivel ez a Unix UTC időtárolási formátuma. Ez tekinthető a második dimenziónak. minden egyes epocha egy *GNSSObservation* típusú változót tartalmaz. Mivel ez egy dinamikus tömb ismerni kell a mindenkori méretét, erre egy *int* tömb szolgál.

A *GNSSObservation* struktúra tárolja, egy adott epochára egy adott műhold észleléseit *double* típusként, illetve az ehhez kapcsolódó flageket, és egyéb értékeket egész számként. Majd később látni fogjuk, hogy a *double* pontosság több helyen fejtörést okozott. Az előző pontban írtam, hogy az észlelések önmagukban nem sokat érnek, szükség van a fejlécre a megfejtésükhöz. Éppen ezért az egész fejléctet letároljuk egy másik struktúrában, ez a *GNSSObservationHeader*.

Minden észlelés tartalmaz egy pointert egy ilyen fejléc struktúrára. Azért van szükség a pointerre, mert nagyon sok észleléshez ugyanaz a fejléc információ tartozik, és igen nagy pazarlás lenne, ha mindig külön letárolnánk ugyanazt. Viszont nem minden észleléshez tartozik ugyanaz a fejléc. Ha történt valami esemény, akkor az addig érvényes állapot



megváltozik, kibővül. Ilyenkor új fejléceket kell létrehozni, és onnantól kezdve az észleléseknek arra kell hivatkozni. A fejléceket láncolt listába szervezve kezelem. Magának a *GNSSObservationHeader* struktúrának több alstruktúrája van a fejléc adatok megfelelő tárolásának céljából, ezeket nem részletezem.

Az adatok feldolgozását végző függvény input/output paraméterként kapja meg az észlelések kétdimenziós tömbjét, az észlelések számait tartalmazó tömböt, a fejlécek listáját, a feldolgozandó fájl elérési útját, illetve a fejlécek darabszámát. A visszatérési érték 1, ha sikeres volt a feldolgozás, és 0, ha sikertelen.

Maga az algoritmus, nem túl bonyolult, inkább csak hosszadalmas gépies kódolás volt, mivel nagyon sok esetet kellett lekezelni. A fájlkon szintaktikai ellenőrzést nem hajtunk végre, nem egy „RINEX fordító” írása volt a cél. Csupán olyan szintű ellenőrzéseket végzünk, hogy megfelelő inputra lett-e meghívva a függvény, tehát nem mondjuk GPS navigációs fájlra, illetve, hogy tartalmaz-e olyan adatokat, amiket nem kezelünk le (például Galileo műholdak). Futás során a program először neki áll a kiindulási fejléc információk feldolgozásának. Ezt soronként olvassa be, és a sor végén levő címke alapján egy hatalmas if/else elágazással dönti el, hogy mit kell kezdeni az adott sorral, és a megfelelő értékeket beleteszi a fejléc megfelelő tagváltozóiba. A fejléc végét a megfelelő címkével tudjuk detektálni.

A kiindulási fejléc feldolgozása után, következnek az adatok. Minden adatbejegyzésnél elsőnek beolvassuk a felvezető sort. Ebből azonnal kiderül, hogy észlelésekről, vagy valamilyen eseményről van-e szó. Amennyiben esemény jön, tudjuk azt is, hogy hány sor tartalmazza a rá vonatkozó információkat. Ilyenkor létrehozunk egy új fejléceket, ami az előző másolata, és a megfelelő számú sorra ismét lefuttatjuk a fejléc feldolgozást kibővítve az újonnan létrehozott struktúrát. Ettől kezdve ez lesz az érvényes. Ha észlelések következnek, akkor megint csak pontosan meg tudjuk határozni, hogy hány sor adatot kell beolvasni, és a megfelelő műholdak észlelései közé behelyezzük azokat. Ezt a lépést ismételjük ciklikusan, amíg el nem érjük a fájl végét. Ha nem volt menet közben hiba, 1-gyel térünk vissza.

Ezt az algoritmust elég sok 2.11-es verziójú fájlra (több száz) teszteltem le, mivel a modell alkotása során is ezt használtam. Tesztelés közben jött elő egy-két hiba, ezek javítva lettek. Jelen állás szerint, a legtöbb fájl sikeresen be tudja olvasni, amit pedig nem, azokban valóban van szintaktikai hiba, illetve lekezeletlen adatokat tartalmaznak, viszont ezekben az esetekben sem dől össze a program, hanem egyszerűen 0-val tér vissza a függvény. Mint említettem korábban, ezen kívül elkészítettünk még a meteorológiai és a GPS navigációs fájlokat feldolgozó algoritmust is. Mivel a fájlok szerkezete is hasonló, így az algoritmus is az, ezért ezeket külön nem részletezném. Ezeket csupán néhány fájlra próbáltam ki, az



azokban előjövő hibákat kijavítottam, de valószínűleg a kimerítő tesztelés hiánya miatt, még maradtak bennük. Ezt pusztán statisztikai alapokon feltételezem.

A hatékonyságról szólnék még pár szót. A modell felépítéséhez kb. 60-70 állomás egész napos fél percenkénti mintavételezésű RINEX állományait használtam fel. Memória használat szempontjából a program igazán nagyvonalú, csupán néhány megabájtot használ. Az esetleges memóriavesztéseket a Valgrind analízáló program ([6] forrás) segítségével feltártam, és kijavítottam. A futási idő már korántsem ennyire rózsás. Az említett fájlok feldolgozása egy átlagosnak mondható asztali gépen nagyjából 5 percet vett igénybe. Ez első ránézésre nagyon soknak tűnik, ugyanakkor ne felejtjük el, hogy egész napos mérésekről van szó. Nekünk legfeljebb 15 percnyi adatra van szükségünk, mivel ha ennél nagyobb intervallumból veszünk észleléseket, az már túl pontatlan modellt fog eredményezni, esetleg ellentmondásosak lesznek az adatok. Szerencsére létezik egy parancssori program a TEQC ([7] forrás), amivel a RINEX állományokon transzformációkat lehet végrehajtani, többek között például időintervallumra szűrni, így elő tudunk állítani a napi mérésekből olyan fájlt, ami csak a nekünk érdekes adatokat tartalmazza. Ezekkel a megritkított fájlokkal a program már pillanatok alatt lefut. De ha még ez sem lenne elég, van még egy gyorsítási lehetőség. Jelenleg a feldolgozás egy szálon fut, viszont a mai számítógépekben általában legalább négy fizikai processzor van, tehát a többszálú feldolgozással még negyedére lehetne csökkenteni a futási időt, de mint láttuk erre egyelőre nincsen szükség. Összességében elmondható, hogy a mérési adatok beolvasása semmilyen módon sem gátolja a valós idejű modellszámítás lehetőségét.

5.4 A Grid model modul megvalósítása

A cellalapú modell elméleti hátterét már ismertettem a 4.2-es fejezetben. Itt most az implementáció részleteire térnék ki. Mivel ez a modul jóval bonyolultabb, és több matematikai számítást tartalmaz, mint a RINEX parser modul, így ebben jobban el fogok merülni.

A modellben elsősorban koordináta-geometriai számítások elvégzésére van szükség, így elengedhetetlen, hogy rendelkezünk az ezeknek megfelelő geometriai primitívekkel. Az internetről is lehet letölteni ilyen könyvtárakat, de vagy körülményes a használatuk, vagy nem tudnak mindent, amire nekünk szükségünk van. Mivel nekünk nem kell túl sok minden, így úgy döntöttünk, hogy magunknak megírjuk ezeket.



A számításokban gyakran kell ellenőrizni valaminek a nullával való egyenlőségét, például két vektor merőlegességének a vizsgálatához a skaláris szorzatukat kell, hogy megnézzük. Lebegőpontos számokkal dolgozunk, és ezeknek általában nem lehet a nullával való egyenlőségét ellenőrizni, mivel a számítások eredményei nulla helyett csak nagyon kicsi számok. Ezért definiáltunk egy konstans epszilont, aminek az értéke 10^{-6} . A nullával való egyenlőség ellenőrzését pedig úgy végezzük, hogy megnézzük, hogy az adott változó $\pm 10^{-6}$ között van-e. Természetesen, az hogy ez a 10^{-6} megfelelő-e, nagyban függ az alkalmazott mértékegységektől illetve a megkívánt pontosságtól is.

Az első és legalapvetőbb geometriai primitív, amit elkészítettünk, az a vektor volt. Ezt vegyesen használjuk pontok, és irányok megadására. Általánosságban ismertek a vektor két végpontja közötti X, Y, Z koordináta különbségek illetve a vektor hossza. Pontoknál ugyanez igaz, csak a vektor kiindulása mindig az origó, a vége pedig a pont. Készítettünk hozzá egy legyártó függvényt is, ami automatikusan kiszámolja a koordináták alapján a hosszát. Az ellentmondások elkerülése végett célszerű ezt használni. Hasonló módon, a többi manipulációhoz is készítettünk függvényeket. A struktúra tagváltozóit ajánlott a függvényeken keresztül manipulálni, nem pedig közvetlenül. A megvalósított függvények: skalárszorzás, keresztszorzás, normalizálás, két vektor összeadása, egy vektor kivonása egy másiktól és skalárral való szorzás.

A következő alakzat a forgási ellipszoid, amit úgy kapunk, hogy egy ellipszist az egyik tengelye körül körbeforgatunk, azaz forgásszimmetrikus, így elég az ellipszis adatait ismerni. Ezek az adatok a középpont, a két tengely és az excentricitás. Az utolsó háromból kettő elég a harmadik meghatározásához, így ez is redundáns. Emiatt a vektorhoz hasonlóan ehhez is írtunk egy létrehozó függvényt. Ezt a függvényt többféle módon is lehet paraméterezni, attól függően, hogy melyik két adatot adtuk meg.

Az elméleti modellnek megfelelően elkészítettünk még egy sík struktúrát is. Egy síkot a térben legegyszerűbben az egyenletének az együtthatóival tudunk megadni, azaz az (A, B, C, D) számokkal. Ebből az (A, B, C) számhármassal a sík normálvektora. Ezen felül tároljuk még a sík egy tetszőleges pontját, mert egy ponttal és egy normálvektorral egyszerűbb és szemléletesebb a sík megadása. Ismét redundancia keletkezik, ismét létrehozó függvénnyel kezeljük.

Létrehoztunk még egy egyenes vonal típust is, hiszen műhold-vevő trajektóriákkal szeretnénk számításokat végezni. Az egyenest két pontjával tudjuk megadni legegyszerűbben, ezen felül tároljuk még az orientációját is, mivel számításoknál erre is szükség lehet, így egyenest is célszerű a létrehozó függvényével definiálni.



Végül, de nem utolsó sorban definiáltunk egy tesszeroid típusú struktúrát is, hiszen a modellünk rácsa ilyen alakzatokból fog állni. Ezt a határoló felületeivel adjuk meg. Ezek egy alsó és egy felső határoló ellipszoid, továbbá néhány sík. Ez a megadás még nem egyértelmű, és nem is ellentmondásmentes. Az ellentmondás mentességet az garantálná, ha a biztosítanánk, hogy az ellipszoidoknak azonos az excentricitásuk és a középpontjuk, illetve a síkok mind átmennek ezen a közös középponton, és az ellipszoidok felületén zárt ellipszoidi alakzatot metszenek ki. Ennek az ellenőrzésével külön nem foglalkoztunk, a hívóra bízunk a helyes adatok használatát. Ellentmondásmentes megadás esetén sem egyértelmű még, hogy milyen tesszeroidról van szó, hiszen mivel a síkok is átmennek a közös középponton, így az egész alakzat forgásszimmetrikus, tehát az ellipszoidok két átellenes oldalán két tesszeroidunk van. Ezen többértelműség miatt letároljuk még a tesszeroid csúcsainak koordinátáit is, bár ezeket ténylegesen nem fogjuk hasznosítani.

A geometriai primitívekhez írtunk még néhány közös függvényt is, amelyek a koordináta geometriai alapszámításokat végzik:

- egy vektor melyik oldalán van egy síknak
- egy vektor melyik oldalán van egy ellipszoidnak
- vonal és ellipszoid metszése
- vonal és sík metszése

A fenti megadása a tesszeroidnak túl általános, a mi modellünk esetén csupán négy illetve három oldalsíkkal rendelkező alakzatokra szorítkozunk, ezek közül is azokra, amelyeknek a megfelelő oldalakon levő csúcsaiknak megegyeznek a szélességi illetve hosszúsági koordinátái. Éppen ezért írtunk egy függvényt, ami ilyen tesszeroidot állít elő (az oldalak számát nem kötöttük meg). Ennek a bemenő paraméterei a következők:

- alsó és felső határoló ellipszoidok, melyek geocentrikusak, és az excentricitásuk megegyezik a WGS84-ével
- legalább három pont a WGS84 alapfelületen megadva földrajzi szélességgel, és hosszúsággal
- a fenti pontok száma

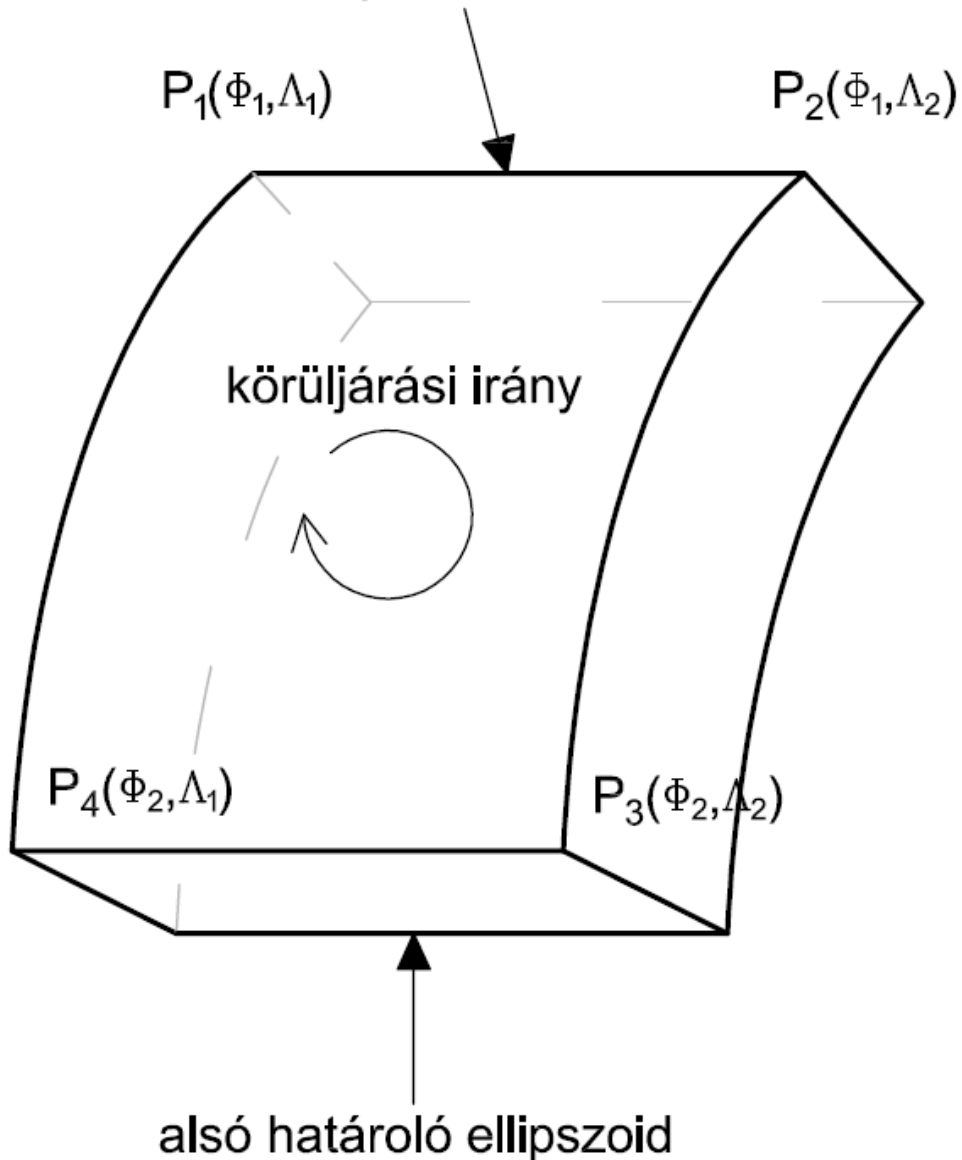
A pontokat az ellipszoidok külső felületén óramutató szerinti körüljárási sorrendben adjuk meg. A paraméterek megadásának módját szemlélteti az 5.2-es ábra.



A létrehozás során először meghatározzuk a földrajzi koordinátákkal adott pontok alapfelületi (X, Y, Z) koordinátáit. Az ezeken a pontokon illetve az origón áthaladó egyenesek már kimetszik a két határfelületből a tesszeroidunk csúcsait. Rögtön kettőt is a forgásszimmetria miatt. Azt, hogy melyik a helyes, úgy tudjuk meghatározni, hogy képezzük az alapfelületi pont és az eredmények skalárszorzatát, amelyik pozitív, az lesz a helyes. A csúcsok meghatározása után következnek a határoló síkok. Megadás szerinti sorrendben végigmegyünk a csúcsok vektorain, mindegyiknek vesszük a következővel való vektoriális szorzatát, az utolsó csúcsnak pedig az első csúccsal vett vektoriális szorzatát. Az így kapott vektorok lesznek a határoló síkok normálvektorai. Ezen kívül szükségünk van egy pontra még. Ez lehet bármelyik csúcs, amit a normálvektor számításakor felhasználtunk, de mint említettem, az összes sík áthalad az origón, így a legegyszerűbb megoldás, ha ezt választjuk az ismert pontnak.

A tesszeroidjainkat becsomagoltuk egy cella struktúrába, ami a modellen belül való elhelyezkedésről tartalmaz információkat. A modell ismertetésénél írtam, hogy úgy kell elképzelni kiterítve, mint egy téglatestet, ami több kicsi téglatestre van felosztva, ezek jelképezik a cellákat. Ezeket három dimenzió segítségével tudjuk kiválasztani. Az első az, hogy vertikálisan lentől felfelé hányadik rétegen tartózkodunk, a másik, hogy észak-dél irányban hányadik sorban van a cella, a harmadik pedig, hogy nyugat-kelet irányban hányadik oszlopban. Mivel ez a megadás diszkrét, és véges, így valójában egyetlen számmal is meg tudnánk adni a cellákat, mint ahogyan a számítógép is egy dimenziós tömbként tárolja a memóriában a háromdimenziós tömböket. Ennek a megadásnak az lenne az előnye, hogy minden cella egyetlen közvetlenül elérhető azonosítóval rendelkezik, és az egyenletek megoldása után egyszerűbb lenne azonosítani, hogy melyik érték melyik cellára vonatkozik. Ugyanakkor a hátulütője, hogy mi szemléletesen három dimenzióval tudunk megadni egy cellát, így ezt a három indexet mindig át kellene számolni az egyszeres indexre, hogy hozzáférjünk az elemekhez. További hátrány, hogy az egyszeres indexszel bonyolultabb a szomszédossági viszonyok meghatározása, így maradtunk a háromdimenziós megadásnál, de írtunk egy függvényt is, ami adott modellre a háromkoordinátás megadást átszámolja egyszeresre.

felső határoló ellipszoid



5.2. ábra: A tesszeroid paraméterei

A fenti három adaton kívül, minden cellához tárolunk egy mutatót a szomszédjaira egy hatelemű tömbben. A tömbben levő pozíció jelenti, hogy melyik szomszédról van szó konkrétan. Ezek az indexek szerint 0-5-ig a következő sorrendben vannak: alsó, felső, északi, keleti, déli, nyugati. Előfordulnak olyan helyzetek is, ahol valamelyik irányban nincsen szomszéd. Például ha egy modell szélén levő celláról van szó, illetve van még egy olyan speciális eset, hogy a cella az északi vagy a déli sarkon van (nyilván nem túl praktikus, de



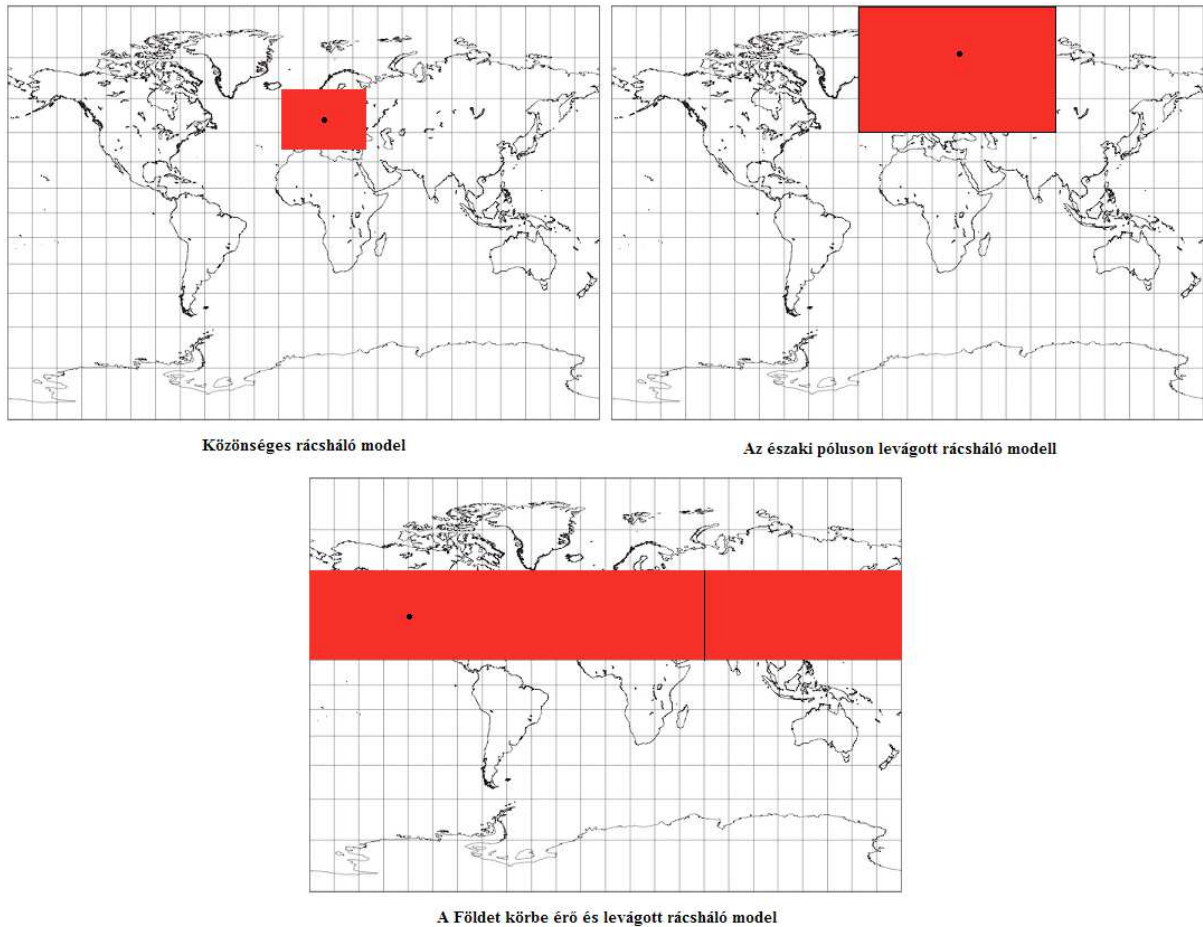
lehetőség van arra is, hogy egyszerre az északi és a déli sarkon). Ilyenkor a megfelelő irányú szomszédnál egy null pointert tárolunk.

Most, hogy már megvannak a modellünk építőkövei, nekiállhatunk felhúzni azt. Az ezt a feladatot ellátó függvény bemenő paraméterként a következő adatokat kapja meg:

- a középpont földrajzi koordinátái az alapfelületen
- a cellák egységes szélességi szögmérete
- a cellák egységes hosszúsági szögmérete
- a rétegek száma
- a szélességi körök mentén a középponttól (tehát keletre illetve nyugatra) hány cellát akarunk
- a hosszúsági körök mentén a középponttól (tehát északra illetve délre) hány cellát akarunk

Az ionoszféra alsó és felső határa konstansként van definiálva. A modellt vertikálisan ebben az intervallumban osztjuk fel annyi közel egyenlő vastagságú rétegre, amennyit megadtunk. Ezen a megkötésen viszonylag kevés átírással tudunk szükség esetén változtatni. A cellák száma egy irányra vonatkozik, például ha a szélességi körök mentén hármat adunk meg, akkor az keletre és nyugatra is három lesz, tehát összesen kétszer annyi, azaz hat. Elképzelhető, hogy lettek volna praktikusabb megadások is, ezt nem tekintettük annyira központi kérdésnek. Ez a megadásmód sem ellentmondásmentes. A földrajzi szélességek -90 foktól $+90$ fokig illetve a hosszúságok -180 foktól $+180$ fokig terjednek. A hosszúsági határokat a kezdőponthoz relatívan nézzük, míg a szélességi határokat abszolút értelemben (tehát a sarkpontok adják). Elképzelhető olyan megadás, amiben ennél nagyobb (vagy kisebb) szögértékeket kapunk eredménynek. Például ha azt mondom, hogy a szögmérték legyen 10 fok, és legyen 100 cellánk minden irányban. Ilyen esetekben úgy kezeljük a helyzetet, hogy a modellt ennek megfelelően hozzuk létre, majd a határokat elérve levágjuk azt. Ez annyit jelent, hogy a modellben az a cella lesz a szélső, amelyiknél elérjük a ± 180 fokot relatív hosszúságban, illetve a ± 90 fokot abszolút szélességben. Ebből következik, hogy az utolsó cellák szögmérete legfeljebb annyi lehet, mint amennyit megadtunk. Ugyanez igaz a cellák számára is. Ennek a megadásnak a lehetőségeit szemlélteti egy réteggel hengervetületen az 5.3-as ábra. Még egy dologra oda kell figyelni, mégpedig arra, hogy amennyiben a modell eléri kelet-nyugat irányban a határokat, a szélső cellákat (a legnyugatabbra és legkeletebbre levőket) szomszédossá kell tenni. A sarkvidéki cellák esetében ezzel nem kell foglalkozni,

mivel gyakorlatban nulla annak a valószínűsége, hogy egy műhold-vevő trajektória pontosan a pólusok felett haladjon át.



5.3. ábra: Különböző elhelyezkedésű rács modellek

A függvény egy rács struktúrát hoz létre, ami a modellt reprezentálja. Ez a következőket tárolja:

- a modell középpontját
- a modell celláit egy háromszoros mutatóban (azaz itt eltértünk a háromdimenziós tömb elvétől, de ez csak memóriabeli tárolásnál jelent különbséget)
- a cellák számait a négy égtáj irányában
- a rétegek számát
- a cellák egységes szélességi és hosszúsági szögmértékeit



Ezen felül a rács tartalmazza még a határoló ellipszoid felületeket is, mivel ezek nagyon sok cellára azonosak. Ezzel memóriát spórolunk, hogy nem tároljuk le őket minden cellához külön, hanem csak egy pointert adunk rájuk.

Miután létrehoztuk a rács struktúrát, feltöltjük a már ismert adatokkal. Elvégezzük az ellenőrzést, hogy a megadott paraméterek elérik-e a modell határait, és ha igen, akkor levágjuk a modellt, és módosítjuk a cellák számát a szükséges módon. Ezt követően jön a cellák létrehozása. Végigiterálunk a rétegeken, illetve az észak-dél és kelet-nyugat irányú felosztáson, és a földrajzi koordinátákat mindig a megfelelő módon eltolva adjuk meg a cellákhoz szükséges paramétereket. Arra kell figyelni, hogy ha a modell eléri a határszögeket, akkor ne egységnyi cellaméretet alkalmazzunk, hanem a fennmaradó résznek megfelelőt. Ezen felül fontos még, hogy a pólusoknál csak 3 földrajzi koordinátát használunk, és figyelni kell a sorrendre is, hiszen körüljárás szerint nem mindegy, hogy az északi vagy a déli sarkon vagyunk. Ha készen vannak a cellák, akkor a szomszédsági viszonyokat kell felállítani. Ez viszonylag egyértelmű a háromkoordinátás megadásnak köszönhetően. Egyetlen különlegesség van benne, amit már korábban említettem, hogy ha a modell körbe ér a Földön, akkor a legkeletebbi és legnyugatabbi cellákat szomszédossá kell tenni. Ezzel a lépéssel a rács modell elkészült és lehet számításokra használni.

Az alakmátrixunk felállításához szükség van az adott műhold-vevő vektorok egyes cellákba eső szakaszainak a meghatározására. Ezeket a szakaszokat egy vonal szektor struktúrában tároljuk, és láncolt listába szervezve kezeljük őket egy adott trajektóriához. A listában az első elem a vektornak abba a cellába eső szakasza, amelyiknél belép az ionoszférába, matematikailag fogalmazva eléri a legfelső határoló ellipszoidot. A lista következő eleme mindig a soron következő cellákba eső szakasz. Mivel a szomszédsági viszonyok ismertek, így annak a meghatározásával, hogy melyik határoló felületen hagyja el a vektor a cellát, rögtön megkapjuk azt is, hogy melyik lesz a következő cella. A lista végét egy null pointer jelzi. Ezeknek megfelelően a struktúrában letárolt adatok a következők:

- a cella három modellkoordinátája
- a vektor cellába eső hossza
- a vektor kilépési és belépési koordinátája
- pointer a következő vonal szektorra

Ezeket a vonalszakaszokat szintén egy függvény állítja elő a modellből, a műhold és a vevő koordináták ismeretében. Az algoritmus lépéseit először nagyvonalakban ismertetném, majd a fontosabb részletekre külön kitérek. Meghatározzuk a műholdon és a vevőn átmenő



egyenest. Ez után meghatározzuk, hogy a legfelső rétegben melyik cellában lép bele a modellbe a vektor. Ezt úgy tesszük, hogy a legfelső rétegen végigiterálunk, és minden cellára megvizsgáljuk, hogy az egyenes, és a legfelső határoló ellipszoid metszéspontja belesik-e a cellába. Az első találatot jelöljük ki belépési cellának. Ha nincs ilyen, az adott vektort eldobjuk, mert rossz módon, vagy egyáltalán nem metszi a modellünket. Ezután meghatározzuk a vektor cellából való kilépési koordinátáját, illetve ebből azt, hogy az melyik oldal. Az oldal alapján tudjuk, hogy melyik a következő cella, és arra megismételjük ezt a számítást, miközben letároljuk az előző cella adatait. Ha az egyik cellába a kilépési oldalon nincsen szomszéd, az azt jelenti, hogy a vektor a modell oldalán lép ki, és ilyenkor eldobjuk. Helyes esetben a számítás akkor ér véget, ha a legalsó rétegnél tartunk, és a cellából az alsó határoló ellipszoidon lépünk ki.

A legelső cella megtalálásánál először meghatározzuk az egyenes és a legfelső ellipszoid metszéspontját. Ennek muszáj kettő eredményt adnia, különben a modell alkotásakor valamit elrontottunk (például a vevőállomás a Holdon van). A két megoldás közül az lesz a helyes, amelyik közelebb van a műholdhoz. A cellákat oldalról határoló síkok normálvektorai mind a cellából kifelé mutatnak. Emiatt, ha a keresett pont a cellán belül van, akkor a síkok normálvektorainak, és a pontba mutató vektor skalárszorzatainak mind negatívnak kell lennie. Ha találtunk egy ilyet, akkor megvan az első cella, és erre kell meghatározni a szakaszhosszt, illetve a következő cellát. Itt ki kell térnünk egy speciális esetre. Előfordulhat, hogy ugyan a metszéspont egy adott cellán belül van, de nagyon közel egy vagy két határoló síkhoz, mondhatni rajta. Ilyenkor, ha az egyenes orientációja valamelyik sík normálvektorának az irányába mutat, tehát a skaláris szorzatuk pozitív, a műhold-vevő vektornak csak elenyésző része fog a cellába esni. Ilyenkor jobban járunk, ha azt mondjuk, hogy nem találtuk meg a keresett cellát, és továbbmegyünk. Mivel a számításaink a megkövetelt pontosságon belül azt az eredményt adták, hogy rajta van a pont a síkon, így a cella szomszédjánál is rajta lesz, tehát belső pontként lesz felismerve. Ebbe a cellában már ténylegesen használható vonalszakasz fog esni. Ezt a speciális esetet szemlélteti az 5.4-es ábra.

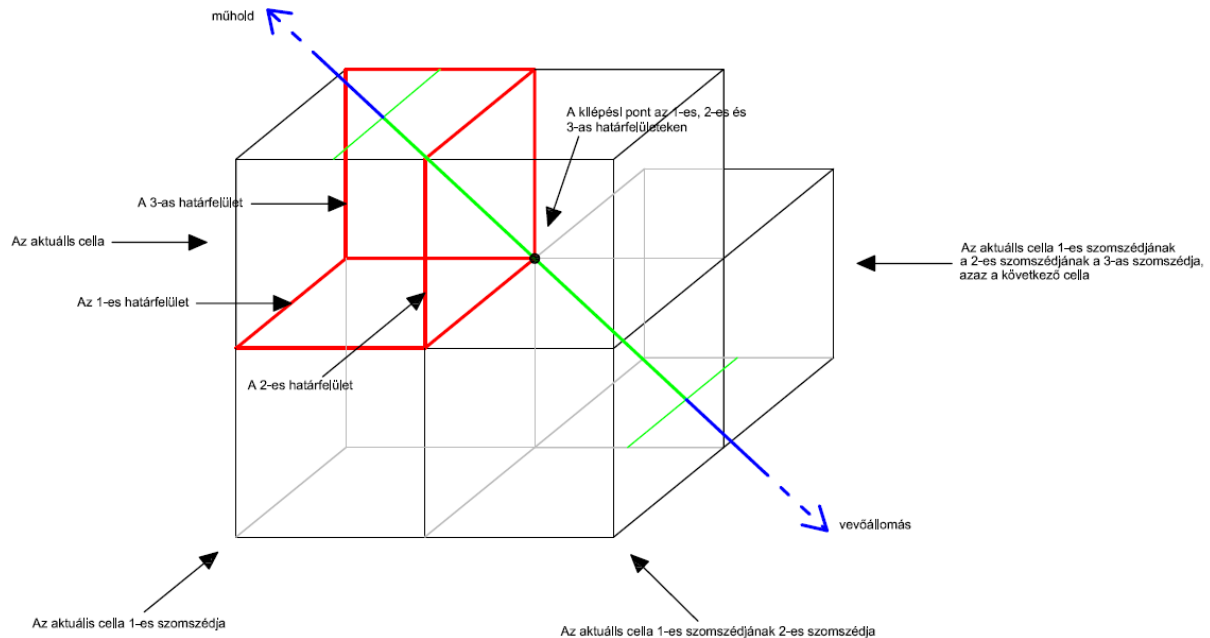


A következő cellát a kilépési ponthoz tartozó oldalakból kapjuk meg. A legegyszerűbb eset az, ha ebből csak egy darab van, ilyenkor közvetlenül vehetjük az adott oldali szomszédot. Viszont itt is figyelni kell az előző fejezetben említett speciális esetekre. Ha a kilépési pont egynél több határoló felületen van rajta, olyankor bármelyik irányba is mennénk tovább, az adott cellába elenyésző szakaszhossz jutna. Például, ha a kilépési pont az északi és a keleti határoló felületen van, olyankor az északi és a keleti szomszédon nem megy át a vonal, hanem csak az északkeleti szomszédon. Hasonló módon elképzelhető az a helyzet is, amikor a kilépési pont egybeesik a cella egy csúcsával, tehát három határoló felületen van rajta (két oldalsík és egy ellipszoid). Könnyen látható, hogy ezekben az esetekben úgy járhatunk el helyesen, ha a szomszédok szomszédjait vesszük. Formálisan megfogalmazva, ha a kilépési pont rajta van az a_1 , a_2 , a_3 irányú határoló felületeken, akkor a következő cellát úgy kaphatjuk meg, hogy vesszük az a_1 irányú szomszéd a_2 irányú szomszédjának az a_3 irányú szomszédját. Tulajdonképpen még a sorrend sincs megkötve, mindenhogyan ugyanoda jutunk. Ezeket a speciális eseteket szemlélteti az 5.5-ös ábra. Ha megvan a következő cella, arra is elvégezzük ugyanezt a számítást, és így tovább, egészen addig, amíg el nem hagyjuk a modellt. Ha jó helyen (az alsó határoló ellipszoidon át) hagytuk el a modellt, akkor visszaadjuk a kiszámított szakaszhosszakat tartalmazó lista gyökérelemére mutató pointert, egyébként meg nullával térünk vissza.

A Grid model modul nagyjából ennyiből áll. A tesztprogramban létrehoztunk egy modellt, amit felparamétereztünk, és ellenőrzéskép megvizsgáltuk, hogy a modell adatai helytállóak-e. Ezt elvégeztük a speciális esetekre is, amikor le kell vágni a modell széléit. Néhány adatra ellenőriztük pontosan az értékeket, de sok adatot néztünk meg, hogy nagyságrendileg jók-e. Hasonló ellenőrzéseket végeztünk el a vonalszakaszok számítására is fiktív, de reálisnak mondható műhold és állomás koordinátákkal.

Az ellenőrzések során több hiba is előjött, amiket javítottunk. Ezek közül a legsúlyosabbat a *double* típus pontossága okozta. Itt futottunk ebbe bele először. Alapvetően elég nagy számokkal dolgozunk. SI mértékegységeket használunk, így az állomás koordinátáink hat-hétmilliós nagyságrendben vannak, a műhold koordináták még ennél is nagyobbak, húsz-huszonötmillió méteresek. A határoló síkok normálvektorait alapfelületi koordinátákba mutató vektorok keresztszorzataként állítjuk elő, ami máris 10^{13} nagyságrendben van. Ha erre még ráteszünk egy elhelyezkedés vizsgálatot, az még egy skalárszorzás, és máris 10^{20} körül járunk, ami már túl van a *double* 16 számjegyes pontosságán. Így történhetett meg az, hogy az egyik alkalommal vizsgáltuk, hogy egy pont rajta van-e egy síkon, és rajta kellett volna, hogy legyen, azaz a számítás eredménye 0-át kellett volna, hogy adjon. Ehelyett 4000 lett. Ha megnézzük, a 4000 a 10^{20} -onhoz képest

valóban elhanyagolható, viszont mi az ellenőrzést az előre definiált $\epsilon=10^{-6}$ -hoz képest végezzük el. Ez nyilvánvalóan hibás.



5.5. ábra: A következő cella meghatározásának speciális esete

Alapvetően két jó megoldás jutott eszünkbe erre. Az első az egy végtelen (előre megadható) pontosságú számtípus létrehozása. Ilyet C++-ban könnyedén lehet írni, sőt valószínűleg az internetről is lehet ingyen letölteni, viszont C-ben igazán körülményes lenne ilyet használni, ezért a nyelvi kötöttségek miatt ezt a módszert elvetettük. A másik lehetőség, hogy kivesszük az előre definiált epsilon értéket, és minden helyen, ahol használtuk, egy arra a konkrét esetre érvényes epsilonra végezzük el az ellenőrzést. Ez nem túl elegáns megoldás, így egy harmadik, nem 100%-os megoldás mellett döntöttünk. A síkok és az egyenes létrehozásakor a normál- és az irányvektorokat lenormalizáljuk. Ezzel a problémát okozó számok nagyságrendje 0 és 1 közé került. Jelenleg úgy tűnik, hogy ez a megoldás működik, és kezelhető nagyságrendben tartja a számokat. Magát a modult jobban kiteszteltük a tényleges számítások során. Jelenleg nincsen benne ismert hiba. Teljesítmény szempontjából jónak mondható. A RINEX parser modul több erőforrást igényel, tehát nem ez lesz a szűk keresztmetszet a valós idejű alkalmazhatóság szempontjából.



5.5 Az Ionosphere modeler modul megvalósítása

A harmadik része a megírt programnak a tényleges számításokat végzi. A feladata, hogy a RINEX parser segítségével beolvasson GPS mérési adatokat, egyéb fájlokból beolvasson még néhány adatot, a Grid model használatával létrehozzon egy modellt, és ezek alapján elvégezzen egy kiegyenlítést. Ez már egy futtatható program, melyet parancssori argumentumokkal tudunk felparaméterezni. A paraméterei:

- a RINEX állományok könyvtára
- a DCB fájlok könyvtára
- az állomás koordináták fájlja
- almanac fájl a műholdak közelítő pozíciójának meghatározásához
- kezdő időpont
- vég időpont
- intervallum

A fájlokat és könyvtárakat abszolút vagy relatív elérési úttal kell megadni. Az idő paraméterekkel azt tudjuk meghatározni, hogy a RINEX fájlból milyen epochájú mérésekre vagyunk kíváncsiak. Ezeket GPST másodpercben kell megadni. Ennek az eredménye az lesz, hogy minden olyan mérést beolvasunk, ami a kezdő és a vég időpont között található (azokat is beleértve), és az utolsó mintavétel óta legalább az intervallumban megadott másodperc eltelt, kivéve a kezdő időponttól vett első mérést, mert azt automatikusan feldolgozzuk.

Mivel itt már nagyon sok különböző típusú adattal szükséges dolgozunk, melyek összefüggnek egymással, így az átláthatóság érdekében a konténerek alkalmazása elkerülhetetlenné vált. Első sorban valamilyen map vagy bináris fa típusra van szükségünk, hogy sorba rendezett kulcs-érték párokat tudjunk hatékonyan kezelni. Mivel C-ben nincs lehetőség generikus típusok használatára, ezért minden egyes releváns adattípushoz el kellett készíteni egy konténert, a hozzátartozó kezelőfüggvényekkel együtt. Mi a bináris fák mellett döntöttünk. Ennek a részleteibe nem mennék bele, hiszen az informatikában egy egészen alapvető fogalom, akinek mégis homályos folt, az interneten nagyon gyorsan informálódhat ezzel kapcsolatban.

Az első struktúra a GPS műholdak koordinátáit tárolja. A kulcs az epocha, amire vonatkoznak az értékek GPST másodpercben megadva, az adat pedig az összes műhold koordinátája egy vektor tömbben tárolva. A tömbben a műholdak a sorszámuk szerint helyezkednek el, tehát a 1-es műhold a 0-ás elem, a 2-es műhold az 1-es elem, az utolsó 32-es



műhold pedig a 31-es elem. A második struktúra a vevőállomások koordinátáit tárolja. Minden állomásnak van egy négy karakterből álló egyedi azonosítója. Logikusan következik tehát, hogy ennek a fának ez az azonosító legyen a kulcsa. A tárolt adat az állomás koordinátája vektorban megadva. Ez a két struktúra klasszikus bináris faként lett megvalósítva.

A következő bináris fákat a kiegyenlítés paramétereinek tárolására hoztam létre. Ennek három típusa van. Vevőállomás DCB, műhold DCB és cella elektronsűrűség. Az első esetben a kulcs az állomás azonosítója, az érték pedig a DCB méterre átszámolva. A második esetben a kulcs a műhold sorszáma, az érték pedig itt is a DCB méterben. A harmadiknál a kulcs a cella egyszeres indexelésű sorszáma, az érték pedig a cellához tartozó elektronsűrűség. Ezeknek a bináris fáknak az extra tulajdonsága, hogy az elemek tartalmazzak egy változót, ami azt hivatott jelezni, hogy az adott elem a fában inorder bejárás szerint hányadik (a forráskódban ezt nagyon szerencsétlenül sikerült preordernek nevezni, ez a javítás még várat magára). Az indexelés nullától indul. Ezt a sorszámozást egy függvényhívással lehet létrehozni miután felépítettük a fát. Ha módosítunk rajta (mi ilyet nem csinálunk), akkor újra le kell futtatni ezt a függvényt.

Ennek az egésznek a jelentősége a kiegyenlítés kiértékelésében rejlik. A számítások elvégzése után nekünk eredményül egy vektorunk van. Ebből még nem derül ki, hogy melyik érték milyen változóra vonatkozik. Viszont ha az alakmátrixunk oszlopait ennek az indexelési sorrendnek megfelelően állítjuk össze, akkor az eredményvektorban is ugyanebben a sorrendben kapjuk vissza az értékeket, és máris egyértelmű megfeleltetésünk van a paraméterazonosítók és a vektor elemei között. Ezeken felül van még egy változó a kiegyenlítésben, ez pedig a mérési eredmények javítása. Ez a mi számunkra irreleváns abból a szempontból, hogy melyik mérési eredményre vonatkoznak, így ezekre nem is hoztam létre bináris fát, egyszerűen csak egy dinamikus tömbben tárolom őket. A kiegyenlítés változóinak összefogására hoztam létre még egy struktúrát, amely a bekezdésben említett típusokat tartalmazza, így egy helyen lehet kezelni azokat.

Létrehoztunk még egy láncolt listát is a mérési eredmények tárolására is. A RINEX parsernél írtam, hogy kicsit túllőttünk a célon, és egy általánosabb feldolgozót készítettünk, ami minden eredményt kieszed a fájlokból, és ezeknek csak a töredékére van szükségünk. Éppen ezért a számunkra fontos észleléseket kigyűjtjük egy ilyen mérési struktúrába. Ez a struktúra a következő adatokat tartalmazza:

- az észlelés időpontja GPST másodpercben
- az észlelt műhold sorszáma



- az észlelő állomás négybetűs azonosítója
- A két frekvencián levő észlelések
- egy mutató az észleléshez tartozó vonalszektorokra
- egy mutató a következő mérési adatra

A futás során először beolvassuk a parancssori argumentumokat. Ezután első lépésben beolvassuk az állomás koordinátákat a megfelelő bináris fába. A következő lépésben a RINEX fájlokat dolgozzuk fel egyesével. Miután beolvastunk egyet, a RINEX parser tároló struktúrájából átrakjuk a fontos adatokat (megfelelő epochában megfelelő észlelések) a mérési eredmények listájába, és a beolvasás közvetlen eredményét eldobjuk, mivel az jóval nagyobb méretű memóriát használ. Az összes mérés beolvasása után már tudjuk, hogy melyik időpillanatokban milyen műholdakra (és állomásokra) rendelkezünk mérésekkel, így ez alapján már meg tudjuk határozni a szüksége közelítő műhold pozíciókat az almanac fájl segítségével. Ennek az algoritmusnak a működését nem részletezem, az [1] forrásban megtalálható a számítás menete.

A kiindulási adatok feldolgozása alapvetően ennyiből állt, elviekben most már rendelkezésünkre áll minden szükséges információ a kiegyenlítés elvégzéséhez. Maga a rács modell megadása jelenleg a kódba van beleírva a létrehozásnál, mivel elsőre szerettünk volna legalább egy működő felosztást, és nem nagyon akartuk variálni ezt. Természetesen ezt is ki lehetne tenni parancssori argumentumba. Miután létrehoztuk a rács modellt kiszámíthatjuk a vonalszektorokat az egyes mérésekhez. Nagy valószínűséggel lesznek olyan műhold-vevő párok, amelyeknek a vektora hibás módon metszi át a modellt. Ilyenkor a számítás nullát ad vissza. Ezeket a méréseket ki kell szűrní a listából, mivel nem hordoznak használható információt.

Ha megvannak a letisztázott mérési eredmények, kiszámíthatjuk az alakmátrixot. Ennek az első lépése, hogy létrehozzuk változókat tartalmazó struktúrát a mérések alapján, és ezzel meghatározzuk a mátrix szükséges méreteit. Itt eltérő eredményeket kapunk aszerint, hogy van-e információnk a DCB értékekről. A jelenlegi működés szerint ismerjük mind az állomás mind a műhold DCB-ét. Az alakmátrix ennek megfelelően fog alakulni, azaz annyi sora lesz, ahány mérésünk van, és annyi oszlopa, ahány cellára volt értékelhető mérésünk. A mátrix elemeit a 4.9-es képlet szerint határozzuk meg. A frekvenciák állandóként adóttak az egyes szakasz hosszakat pedig az előző lépésben határoztuk meg.

Az alakmátrix létrehozása után megadjuk a kiegyenlítéshez szüksége többi vektort is. Ezek a mérési eredmények vektora, az állandók vektora illetve a paraméterek előzetes értékeinek vektora. Az első az magától értetődő. A második vektor tartalmazza a



paraméterektől és mérésektől független konstans értékeket az egyenletünkben. Jelen esetben ezek a műhold és az állomás DCB-k, amik fájlokban adóttak. Beolvassuk az értékeket és a megfelelő átalakítások után beletesszük őket a vektorba. A harmadik vektort csupa nullának vesszük fel. Ilyenkor a kiegyenlítést eredménye valószínűleg pontatlan lesz, így több iterációt kell alkalmazni. Ezek után ténylegesen elvégezhető a kiegyenlítés. A szükséges adatokat kiírjuk fájlalba, hogy utána Matlabba be lehessen őket importálni, és ott lehessen kiegyenlíteni, de a C programban is implementáltuk az algoritmust, hiszen valódi felhasználáshoz itt kell elvégezni a számítást.

A program futása nagyjából ennyiből áll. Memória használat szempontjából a RINEX parser továbbra is többet használ. A teljesítmény szempontjából a kiegyenlítés legkritikusabb rész, egészen pontosan a benne levő mátrix inverzió. A számításokhoz a *gsl* könyvtárat (GNU Scientific Libraryt) használom. Ez egy GPL lincensszel rendelkező ingyenes C-s matematikai könyvtár. Azért ezt használom, mert C-re összesen két ingyenes mátrixműveleteket elvégezni képes könyvtárat találtam (a másik az ATLAS volt), és a kettőből ennek volt a barátságosabb API-ja. Ennek a könyvtárnak a nagy hátránya, hogy a mátrix inverzió Gauss-eliminációval van implementálva azaz $O(n^3)$ a lépésszáma, ami nem a leghatékonyabb. Egy 1000-szer 1000-es mátrix invertálásának a futási ideje egy mai átlagos számítógépen már tizedmásodpercekben/másodpercekben mérhető. Ez a gyakorlatban egy 10x10x10-es rácsmodellt jelente, ha minden cellára van mérés, és elvégezhető a kiegyenlítés (erről a következő fejezetben részletesen lesz szó). Egy tízszer ekkora modell futási ideje már több perc. Tehát azt lehet mondani, hogy a közel valószerű felhasználhatóság szempontjából igencsak a határon táncolunk, és a modell felbontásának a függvénye, hogy éppen lehetséges-e.



6. Eredmény

6.1 A felhasznált adatok bemutatása

Vizsgálódásainkhoz EUREF állomások méréseit használtuk fel 2013. április 13-áról. Ezek egész napos RINEX állományokként álltak rendelkezésünkre. Éjszakai mérésekkel foglalkoztunk, mivel az ionoszféra állapota ilyenkor jóval homogénebb. Felhasználtuk még a GPS műholdak adott hétre vonatkozó almanac fájlját, ami letölthető az internetről. Továbbá DCB fájlokat is használtunk a kiegyenlítéshez. A fájlokban található értékeket Bernese szoftverrel ([8]-as forrás) vezettük le az összes műholdra és állomásra, mivel ezeket nem lehetett külön letölteni sehonnan sem. Rendelkezésünkre álltak még a berni CODE (Center for Orbit Determination in Europe) központtól is DCB fájlok, de ezek csak az állomásaink egy részére tartalmaztak értékeket.

6.2 Az első kiegyenlítés eredményei

A számítási eredményeink azonnal szembesítettek a vakmerőségünkkel minket. Azon felül, hogy a számok nagyságrendileg sem stimmeltek, a kiegyenlítési folyamat divergensnek bizonyult. Ebből azonnal nyilvánvalóvá vált, hogy valami hibás.

Sok ideig kerestük az algoritmusban rejlő hibákat, de nem sikerült megtalálnunk. Gondoltunk arra is, hogy talán a normálmátrix rosszul kondicionált. Ez akkor vált gyanússá, amikor leellenőriztük a szorzatát az inverzével, és közel sem egységmátrixot kaptunk. Ekkor tértünk át a Matlab használatára, mivel itt jóval egyszerűbben lehet mátrix műveleteket végezni, így rövidebb és átláthatóbb kóddal lehet megvalósítani egy kiegyenlítést. Ennek a segítségével gyorsan meg is találtuk a hibát. Már az első kiegyenlítés során azt a hibaüzenetet kaptuk, hogy az invertálandó normálmátrix rosszul kondicionált, ami annyit jelent, hogy a mátrix legkisebb és legnagyobb abszolút értékű elemei között túl nagy a nagyságrendi különbség, nagyjából 10^{40} . Itt szembesültünk a *double* pontosság hátrányaival másodjára. Mivel minden számhoz 16 számjegyet tárolunk, így az ennél nagyobb nagyságrendi



különbségekkel rendelkező számok összeadásánál, és kivonásánál pontosságot veszítünk. Egy determináns számításánál egyáltalán nem mindegy, hogy az alábbi számítás eredménye 0 vagy -1:

$$6 \cdot 10^{23} + 1 - 6 \cdot 10^{23} - 1. \quad (6.1)$$

Márpedig a *double* pontosság miatt itt az első összeadás eredménye elveszik, míg a két kivonás eredménye megmarad, és így 0 helyett -1-et fogunk kapni. Matlabban leellenőriztük a normálmátrix rangját, és valóban kisebbre adódott, mint a mátrix mérete. Ezek után külön „eredménynek” érzem, hogy sikerült invertálnunk egy szinguláris mátrixot (az más kérdés, hogy ezzel nem az inverzét határoztuk meg).

Ezen problémán ismét tudnánk segíteni egy C++-os végtelen pontosságú számtípussal, de sajnos egyelőre ezt el kellett vessük a már korábban említett okokból kifolyólag. Az első ötletünk az volt, hogy a Moore-Penrose féle pszeudoinverzet próbáljuk ki. Sajnos ez sem hozott megoldást, így az alakmátrixot próbáltuk kicsit szebbé tenni. Az elektronsűrűségek együtthatói, ha megnézzük azokat, viszonylag kicsi számok, míg a DCB együtthatók egyesek. Ezt a nagyságrendi különbséget áthidaltuk azzal, hogy az elektronsűrűség együtthatókat felszorozzuk nagyjából 1 körüli értékre, és majd a számítási eredményeket ennek megfelelően korrigáljuk. Ez körülbelül megfelelt a mátrix kondíció számát, de ez még mindig túl nagy volt.

A következő, amivel elkezdtünk foglalkozni, az állomások és a modell módosítása. Elképzelhetőnek tartottuk, hogy a műhold-vevő vektorok túlságosan egyenlőtlen eloszlása okozza a rosszul kondicionáltságot, így próbáltuk úgy módosítani a modellt, hogy minden cella alá kerüljön legalább egy állomás. Emellett csökkentettük a méretet is. Kitartó próbálkozásunk eredményeként sikerült meghatároznunk egy 6x6x4-es rács modellt Európa felett, amire a normálmátrix sorai függetlennek adódtak. Ez a modell a 6.1-es ábrán látható. A kiegyenlítés ezt követően már konvergensenk bizonyult, viszont a kapott eredmények továbbra sem voltak helyesek. Ami azonnal feltűnt, hogy az egyes cellák elektronsűrűségeinek az előjele rétegenként változó volt, ami nyilván hibás. Ezen felül a nagyságrendek sem stimmeltek.

A következő ötletünk az volt, hogy a DCB értékek kiegyenlítése húzza el az elektronsűrűségek értékeit, ezért ezeket ismertnek vettük, és az áprilisi havi átlagot olvastuk be fájlból. Sajnos ez sem hozott eredményt, viszont egy fontos következtetést le tudtunk belőle vonni. A 4.9-es képletet megvizsgálva láthatjuk, az elektronsűrűség együtthatója negatív lesz, mivel az f_I jelű frekvencia kisebb, mint az f_{II} jelű frekvencia. Ebből következik, hogy a tisztatag vektornak is csupa negatív elemből kell állnia, ez pedig nem így volt. Ezek



paramétereket. A zaj növelése használhatatlan mértékben rontotta a pontosságot. Ezek után megkötöttük a DCB értékeket, ami javított ezen. Milliméteres szórású zaj esetén a felvett és a kiegyenlített paraméter értékek közötti átlagos eltérés 20% körül volt, viszont 5 milliméternél már itt is nagyon nagy lett. A rétegszámok csökkentésével növekedett a még elviselhető zaj szórása. Egy réteg és 1 méteres szórású zaj alkalmazásával az átlagos eltérés nem lépte túl a 12%-ot. Ezek vannak összefoglalva 6.2-es ábrán. A táblázatban az látható, hogy adott cellaszámnál, és adott szórású zaj esetén átlagosan hány százalékos eltérés van a fiktív paraméterek, és a kiegyenlített értékek között.

	0,001 m	0,005 m	0,01 m	0,05 m	0,1 m	0,5 m	1 m
1 réteg	0 %	0 %	0 %	0 %	1 %	6 %	12%
2 réteg	0 %	0 %	0 %	2 %	5 %	25 %	60 %
3 réteg	2 %	10 %	20 %	70 %	100 % <	100 % <	100 % <
4 réteg	20 %	100 % <	100 % <	100 % <	100 % <	100 % <	100 % <

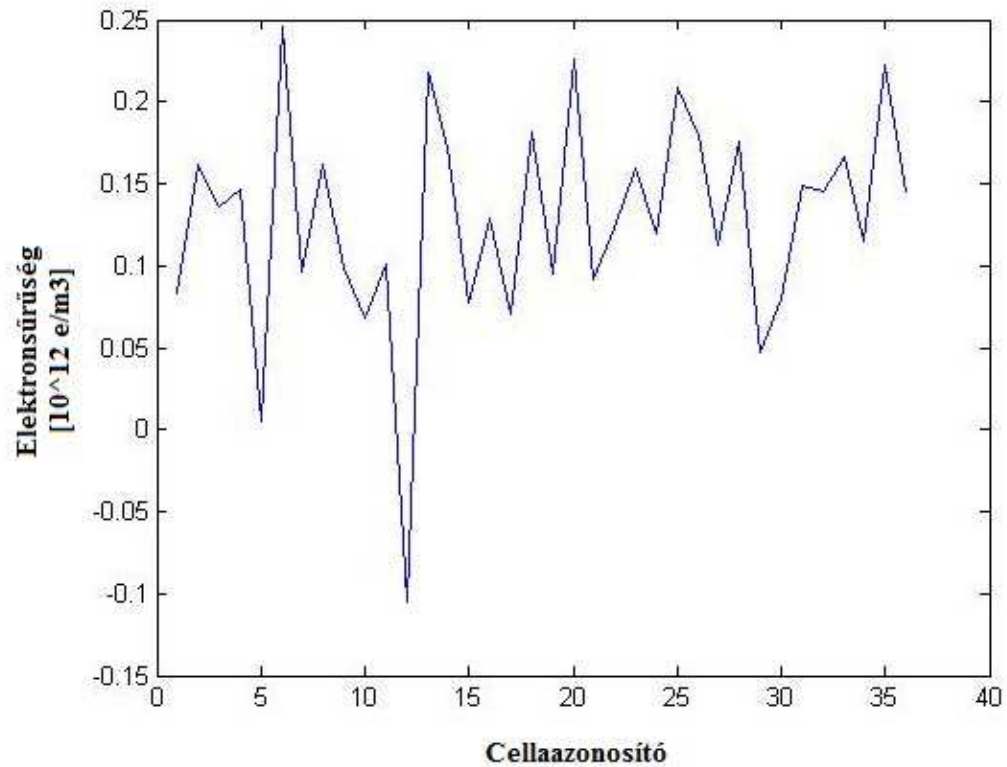
6.2 ábra: A kiegyenlítés pontossága a modellméret és a zaj függvényében ismert DCB értékeket feltételezve

Összefoglalva az eredményeket, zajmentes esetben a módszer helyes eredményt ad. Nagyobb rácshálók esetén már viszonylag kicsi zaj is nagy pontatlanságot eredményez, így nagy modellek esetén a mérésen nem egyenlíthetőek ki a legkisebb négyzetek módszerének alkalmazásával. Egy réteg esetén az 1 méteres zaj ismert DCB értékekkel egészen pontos eredményt ad, ami jó hír nekünk, mivel a GPS kódéréseket is nagyjából ekkora zaj terheli. Ezek alapján elmondhatjuk, hogy kisméretű modelleknél működhet a számítás.

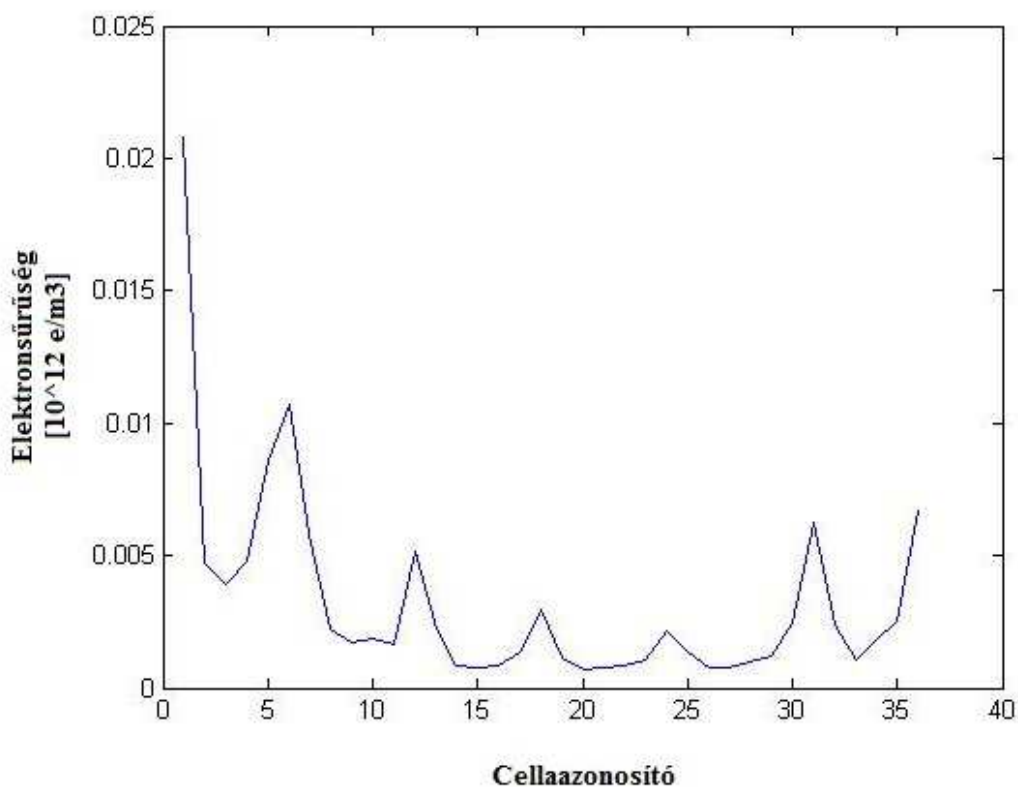
6.4 Első eredmények

Az előző fejezet tanulságai alapján a Bernese programmal elvégeztük a kódérések fázisméréssel történő simítását annak érdekében, hogy a méréseket terhelő zajt csökkenteni tudjuk. Ezt követően végrehajtottuk a tényleges kiegyenlítést egy 6x6-os 1 rétegű modellen, és sikerült egészen reális értékeket kapnunk az elektronsűrűségekre. A kiegyenlített paraméterek láthatóak a 6.3-as ábrán, illetve a paraméterek középhibája a 6.4-esen. A vízszintes tengely a cellaazonosító a függőleges pedig az elektronsűrűség. Mint korábban említettem, ezeket korrigálni kell még, vissza kell szorozni 10^{12} -nel őket, így kapunk elektron/m³-t. Ha az így kapott elektronsűrűségeket felszorozzuk még az ionoszféra zenit irányú vastagságával (ez nálunk 900 km), akkor 5-20 TECU körüli értékeket kapunk, amik

teljesen reálisak a Föld északi oldalára. A középhibákról általában elmondható, hogy két nagyságrenddel kisebbek a paramétereknél, ami egészen jó.

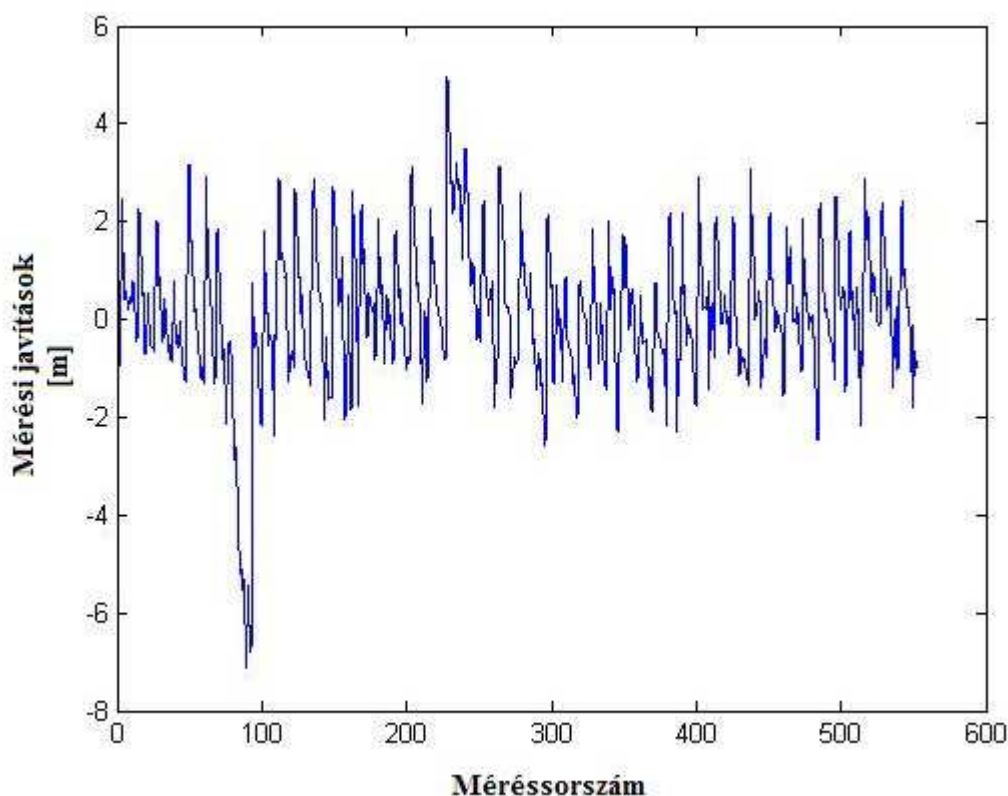


6.3 ábra: Kiegyenlített elektronsűrűségek



6.4 ábra: Kiegyenlített paraméterek középhibái

Látható, hogy van egy negatív tüske a kiegyenlített paraméterek ábráján. Ez valószínűleg azért van, mert az abban a cellában levő egyik állomásról hibás mérési adatokat kaptunk. Ezt támasztják alá a mérési javítások is. Ezek a 6.5-ös ábrán láthatóak. A vízszintes tengely a mérések sorszáma, a függőleges pedig az alkalmazott korrekció méterben. Jól látható, hogy a 90. mérés körül itt is van egy hatalmas kiugrás. Az ugyanarról az állomásról származó mérések egymás után vannak, így ez is megerősíti a gyanúunkat, hogy ott lehetett egy állomás, aminek vagy a mérései, vagy a hozzá tartozó DCB értékek hibásak.



6.5 ábra: A mérések javításai

A középhibánál szintén van egy túske. Ez azzal magyarázható, hogy ezt a cellát csupán egy darab műhold-vevő vektor metszi. Sajnos arra már nem volt időnk, hogy visszaellenőrizzük a modellt. Tehát, hogy a felhasznált észlelésektől független mérésekre meghatározzuk az ionoszférikus késleltetést az elektronsűrűségek segítségével. Emiatt a pontosságról nem tudok bővebben nyilatkozni. Mindenesetre a kiegyenlített értékek biztatóak, és elképzelhető, hogy kis modellek esetén akár gyakorlatban is használható a módszer. Egy durvahiba szűrés valószínűleg el fogja tüntetni a tuskéket, így ezt majd még el kell végeznünk. Amit még megtehattünk volna, de szintén nem volt már rá idő, hogy nappali mérésekkel is vizsgálódunk. Ez azért lehet érdekes, mert az éjszakai mérések esetén a késleltetések nagyon kicsik, és zaj nagyságrendjében vannak, így elképzelhető, hogy nappali méréseknél a zaj hatása még kisebb lesz. A további terveinkben ez is szerepel.



7. Összefoglalás

A dolgozatomban egy olyan számítást, illetve annak megvalósítását mutattam be, amellyel egy közelítő ionoszféra modellt lehet előállítani tomografikus úton GPS kódmerések eredményeit felhasználva. Azt reméltük ettől, hogy a kódérés alapú abszolút helymeghatározás pontosságán tudunk javítani. A kidolgozott módszer kisméretű modellek esetén működik, bár apró problémák adódtak, mivel a felhasznált mérési adatok túlságosan ellentmondásosak voltak. Nagyobb méretű modellek esetében viszont valószínűleg más megoldáshoz kell folyamodni.

Természetesen itt nem adjuk fel a munkát. További vizsgálódások szükségesek, hogy megállapítsuk pontosan milyen méretű modellek esetén használható a módszer. Ezen felül a nagyobb rács hálókhoz szükség van egy új számítás kidolgozására. Köszönhetően a moduláris megvalósításnak nem kell mindent a legelejéről kezdeni, sőt utólagosan azt kell, hogy mondjam, hogy a munka legnagyobb része, a két osztott könyvtár továbbra is használható, és más kiegyenlítési módszerekkel kell próbálkozni, ami nagyobb modell esetén is eltűri GPS kódmeréseket terhelő 1-2 méteres zajt. Ezen felül, ami még fontos következő lépés, hogy megtaláljuk a túl nagy ellentmondásokat a méréseinkben, vagy más ellentmondásmentes adatokat kell használnunk. A harmadik dolog, amit praktikus lenne megvalósítani, az hogy áttérünk C++-ra és el kezdünk használni valamilyen végtelen pontosságú számtípust. Mi ugyan most egy példában meg tudtuk kerülni a *double* pontosság hiányosságait, de általánosságban nem tudjuk garantálni azt, hogy nem lesznek ebből problémák.

Ez egy igazán érdekes kutatási terület, amely sok lehetőséget rejt magában. Ha a fenti dolgokat elérnénk, ott kezdődne az igazi munka. Az említett cikkekben ([3]-as és [4]-es források) sem sikerült nagyságrendileg csökkenteni az ionoszféra okozta hibákat, tehát van még munka bőven a modellek pontosságának területén. A kutatás hasznossága pedig megkérdőjelezhetetlen. Habár azt mondjuk, hogy a jelenlegi navigációs GPS-ek pontossága kielégítő, nekem mégis van több ismerősöm, aki már panaszkodott rájuk. Sőt nekem is volt már kellemetlen tapasztalatom. Volt olyan esetem, hogy két sáv kanyarodott látszólag párhuzamosan, és egyszer csak váratlanul kettéváltak. Ugyan az elágazás ki volt táblázva, de nem túl jól láthatóan, így rossz irányba mentünk, és több kilométert kellett autóznunk, amíg meg tudtunk fordulni. Ha egy nagyságrendet tudnánk javítani a navigációs GPS-ek



pontosságán, akkor elkerülhető lenne egy ilyen eset, mivel a készülék pontosan meg tudná mondani, hogy melyik sávban kell haladni.

Tovább haladva, ha sikerülne elérni a pár 10 centiméteres pontosságot, akkor akár a gyalogos navigáció is elérhetővé válna egyszerű okostelefonokban. Ez a látássérült embereknek lehetne nagy segítség. Én személy szerint egy vakvezető kutyaiban jobban megbíznék, mint egy ilyen okostelefonban, viszont vannak emberek, akik félnek a kutyáktól. Ráadásul sajnos olyan kegyetlen világban élünk, ahol egy vakvezető kutya árából több okostelefont lehet vásárolni. Azt nem állítom 10 éven belül minden látássérült fülbe tehető GPS készülékkel fog az utcán rohangálni, de nem tartom elképzelhetetlennek, hogy pár évtizeden belül ez lesz.

Az említett alkalmazási lehetőségekhez természetesen szükség van a valós idejűsége. Mint láttuk tisztán valós időben nem lehet előállítani a modellt, viszont elképzelhető olyan megoldás, hogy egy nagy teljesítményű számítógép óránként kiszámít egy modellt. A mi kísérleteinkben láttuk, hogy egy egészen részletes modell percek alatt számítható, egy kisméretű modell pedig pillanatok alatt. Ez persze függ a kiegyenlítés módjától is. Viszont ha ezt sikerülne elérni, akkor a készülékek interneten keresztül megszerezhetnék ezeket a korrekciókat, két számítás között pedig valamilyen extrapolációval határoznának meg késleltetési értékeket.

Összefoglalva a kutatás alkalmazási lehetőségeiben igen nagy potenciál van. Habár mi csak egy egyszerű és valószínűleg pontatlan modellel tudtunk eredményre jutni, láttuk, hogy másoknak egészen összetett esetre jött ki használható eredmény. Mi dolgozunk tovább, hogy nagyobb és részletesebb modelleket is sikerüljön felépíteni, mert érdemes a kérdéssel foglalkozni.



8. Irodalomjegyzék

- [1] Ádám J., Rózsa Sz., Takács B.: *GNSS elmélete és felhasználása*, Elektronikus jegyzet, 2012
- [2] Hofmann-Wellenhof, Lichtenegger, Wasle: *GNSS – Global Navigation Satellite Systems*, SpringerWienNewYork, 2008
- [3] Liu Z., Gao Y.: *Ionospheric Tomography Using GPS Measurements*, KIS-2001, 2001
- [4] Jin S., Jin R.: *GPS IONOSPHERIC MAPPING AND TOMOGRAPHY: A CASE OF STUDY IN A GEOMAGNETIC STORM*, Geoscience and Remote Sensing Symposium (IGARSS), 2011 IEEE International
- [5] Gurtner W.: *RINEX: The Receiver Independent Exchange Format Version 2.11*, 2007, <ftp://igs.org/pub/data/format/rinex211.txt>
- [6] Seward J. et al.: *Valgrind Documentation*, 2000-2012, <http://valgrind.org/>
- [7] Estey L., Wier S.: *Teqc Tutorial: Basics of Teqc Use and Teqc Products*, UNAVCO, 2013, <http://facility.unavco.org/software/teqc/>
- [8] Dach R., Hugentobler U., Fridez P., Meindl M.: *User manual of the Bernese GPS Software Version 5.0*, Astronomical Institute, University of Bern, 2007