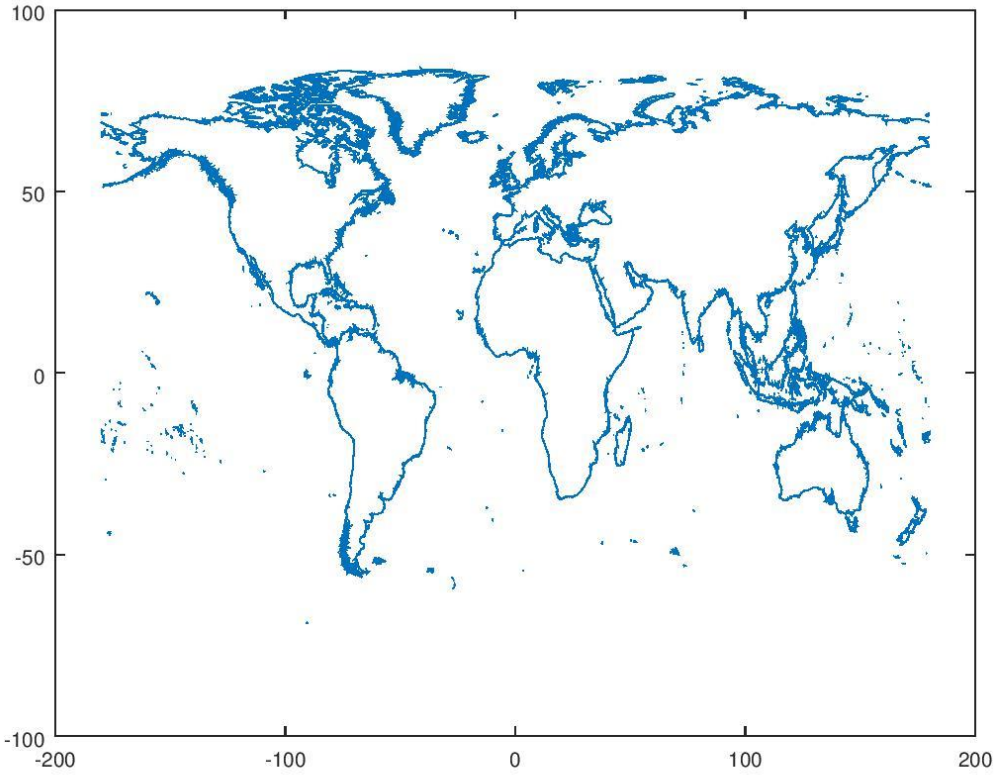
FÖLDGÖMB ÁBRÁZOLÁS 3D-BEN, NMEA GPS ADATOK GOOGLE EARTH (KML) FORMÁBA ALAKÍTÁSA

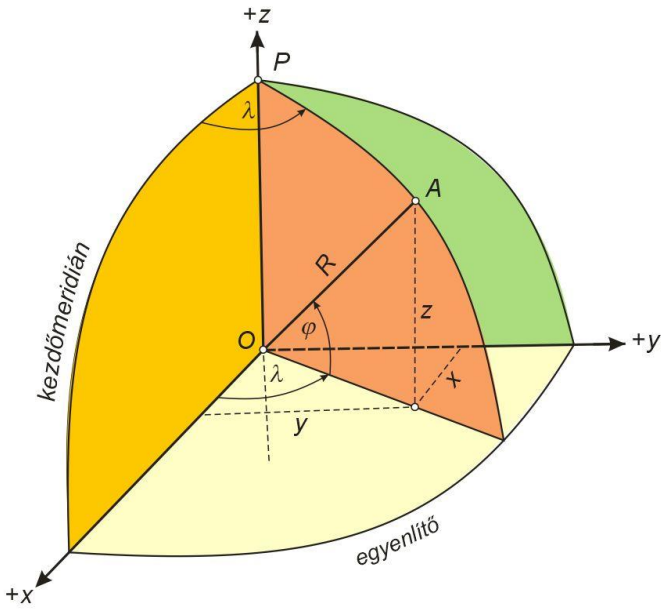
FÖLDGÖMB ÁBRÁZOLÁS 3D-BEN

A coastline.txt fájlban a kontinensek partvonalai találhatóak. ábrázoljuk ezeket 2D-ben és 3D-ben is!

```
> clear all; close all; clc; page_screen_output(0)
>
> % partvonal koordinátáinak betöltése
> % síkbeli ábrázolás (fi, lambda alapján)
> a = load('coastline.txt');
> figure(1)
> plot(a(:,1),a(:,2))
```



A térbeli koordináták számításához gömbi földrajzi koordináta rendszerből térjünk át térbeli derékszögű koordinátákra az alábbi ábra alapján:

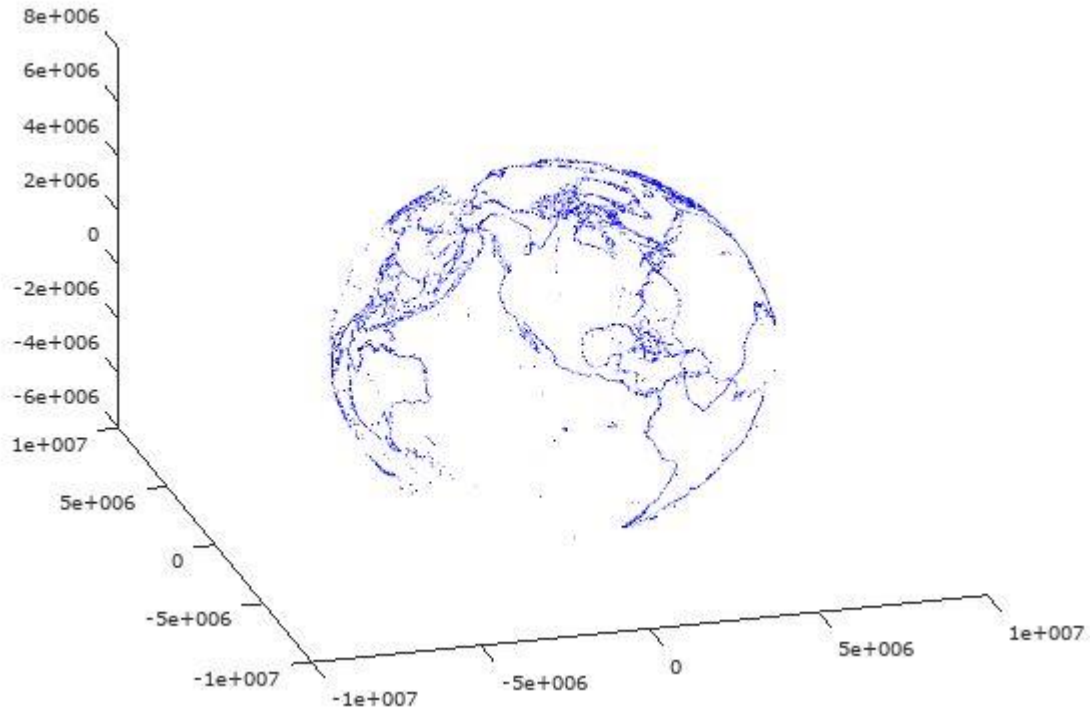


$$x = R \cos\varphi \cos\lambda,$$

$$y = R \cos\varphi \sin\lambda$$

$$z = R \sin\varphi.$$

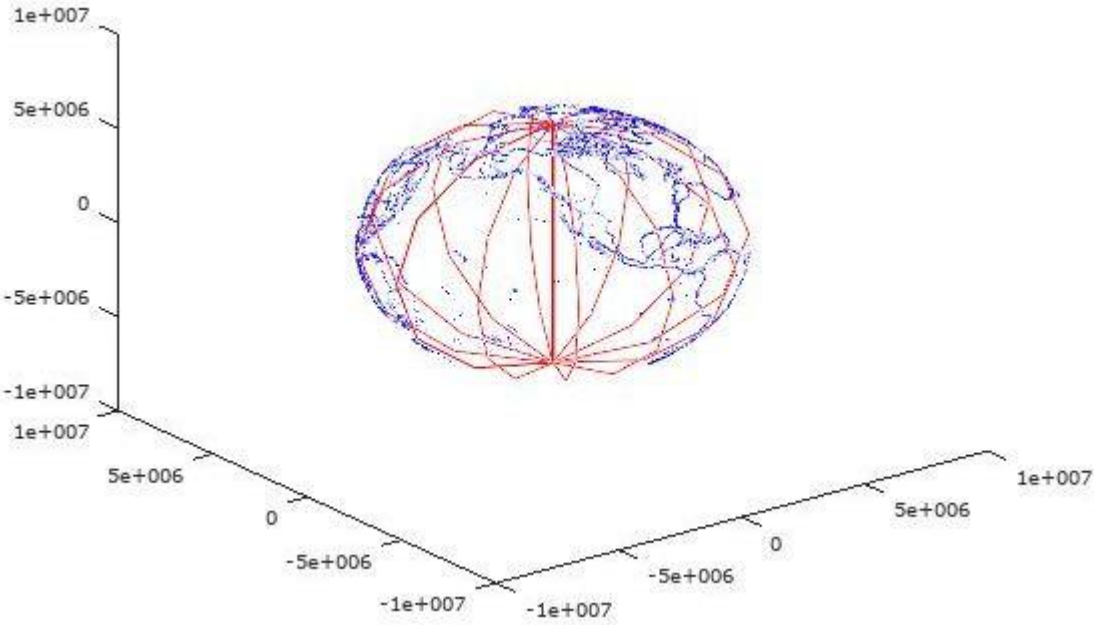
```
> % térbeli koordináták kiszámítása
> R = 6378000;
> z = R*sind(a(:,2));
> p = R*cosd(a(:,2)); % paralelkör sugara
> x = p.*cosd(a(:,1));
> y = p.*sind(a(:,1));
>
> % térbeli ábrázolás
> figure(2);
> plot3(x, y, z, 'b');
```



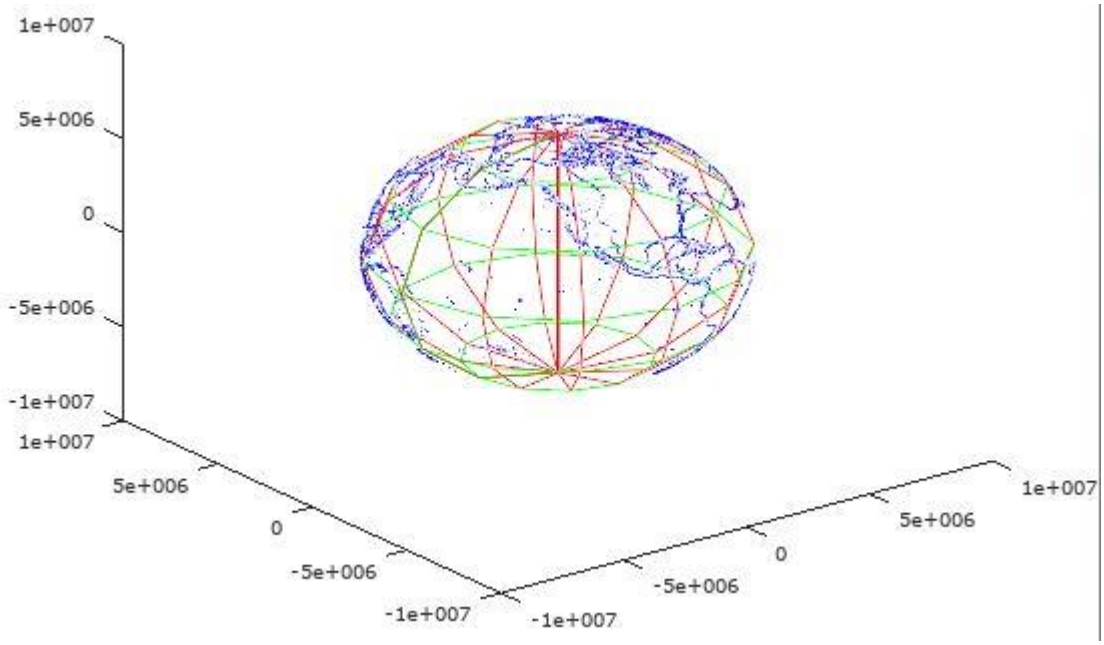
Ábrázoljuk a fokhálózatot is 30 fokonként! Először vegyük fel a földrajzi hosszúság (lambda) és földrajzi szélesség (fi) értékeit 30 fokonként. Utána állítsuk elő a meridián görbék pontjait egy vektorba. Ezeket számítsuk át az előző módon xyz térbeli koordinátákká és ábrázoljuk őket. ismételjük meg a műveletet a paralelkörökkel is! Figyeljünk, hogy egy adott

meridiánnál a lambda állandó (ez kerül a külső ciklusba), míg egy adott paralelkörnél a fi lesz állandó!

```
> % fokhálózati vonalak felvétele 30 fokonként
> lambda = -180:30:180;
> fi = -90:30:90;
>
> % meridiánok görbéi (pontok 30 fokonként)
> meridian = [];
> for i=1:numel(lambda)
>   for j=1:numel(fi)
>     meridian = [meridian; lambda(i) fi(j)];
>   end
> end
> % meridiánok pontjai xyz koord. rendszerben
> zm = R*sind(meridian(:,2));
> pm = R*cosd(meridian(:,2));
> xm = pm.*cosd(meridian(:,1));
> ym = pm.*sind(meridian(:,1));
> % meridiánok ábrázolása 3D-ben
> figure(2); hold on;
> plot3(xm, ym, zm, 'r');
```




```
> % paralelkörök görbéi (pontok 30 fokonként)
> paralel = [];
> for i=1:numel(fi)
>   for j=1:numel(lambda)
>     paralel = [paralel; lambda(j) fi(i)];
>   end
> end
>
> % paralelkörök pontjai xyz koord. rendszerben
> zp = R*sind(paralel(:,2));
> pp = R*cosd(paralel(:,2));
> xp = pp.*cosd(paralel(:,1));
> yp = pp.*sind(paralel(:,1));
> % paralelkörök ábrázolása 3D-ben
> plot3(xp, yp, zp, 'g');
```



NMEA GPS ADATOK GOOGLE EARTH (KML) FORMÁBA ALAKÍTÁSA

Nyaralás alatt hajókázni mentünk és az útvonalat szeretnénk megmutatni az ismerőseinknek is. Az útvonalat GPS-szel rögzítettük, és az adatokat a navigációban gyakran használt NMEA 0183 formátumban kaptuk meg. Ezt szeretnénk megmutatni az ismerőseinknek. Először olvassuk be az adatokat, majd jelenítsük meg a saját korábbi ábránkban. Ezután alakítsuk át a Google Earth KML formátumába, hogy látványosabban tudjuk megjeleníteni az útvonalat.

NMEA ADATOK - RÖVID ISMERTETŐ

Most csak röviden összefoglaljuk az NMEA forméátumról, ami a mi feladatunkhoz kell, akit részletesebben érdekel a téma megnézheti a http://www.nmea.org oldalt, vagy magyar nyelven pl. Ferencz Viktória 2006-os TDK dolgozatát: http://vit.bme.hu/tdk/2006/dolgozatok/ferenczv.pdf. Az alábbi összefoglaló Primusz Pétertől (http://primuszpeter.blogspot.hu/2009/03/modern-navigacio.html) és Ferencz Viktóriától származik.

Az NMEA 0183 egy nemzetközi hajózási elektronikai szabvány, melyet az NMEA (National Marine Electronics Association) nevű nonprofit szervezet ad ki, és gondoz. Eredetileg különféle hajónavigációs eszközök (LORAN, radarok, OMEGA stb.) közötti kommunikációra kialakított szabvány, amelyet kibővítettek GPS-specifikus formátummal is. Napjainkra a GPS-vevőkből kommunikációs porton keresztül nyert információk leggyakrabban használt formája. Kedvelt, hiszen könnyen értelmezhető, rövid, szabadon sorrendezhető és összeállítható ASCII üzenetcsomagokról van szó.
Minden küldött mondat $ karakterrel kezdődik, amelyet a küldő azonosítója követ, pl.: a GPS-t a GP karakterpár fog azonosítani, a hajónavigációs rendszert pedig az II azonosító jelöl.
Az NMEA 0183 szabvány szerint nagyon sok ún. "GPS adatmondat", parancs, lekérdezés létezik, a legáltalánosabb, a Globalsat összes GPS eszköze által használt parancsok jelentése a következő:

- GGA - Pozíció adat hibaértékkel, magasságadatokkal
- GSA - Aktív műholdak
- GSV - Látható műholdak
- RMC - Pozíció és UTC adatok
- GLL - Csak Pozíció adat (hosszúsági és szélességi)
- VTG - útirány és sebesség adatok

Az azonosító ismeretében tudható, hogy a következő (vesszőkkel elválasztott, azaz CSV formátumú) adatok éppen koordinátát, sebességet vagy valami mást jelentenek. A mondatok végén egy ellenőrzőösszeg és egy soremelés vagy sortörés karakter található.

Néhány példa 'GPS adatmondatokra' (minden ilyen mondat $GP-vel kezdődik):

Az egyik legfontosabb üzenet a 'GGA ' karakterhármast tartalmazó mondat, amely tartalmazza a 3D helymeghatározó adatokat és a pontosságra vonatkozó információt. Egy ilyen mondat a következőképpen értelmezhető:
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

GGA	GPS észlelési adatok
123519	észlelési időpont (12:35:19 UTC)
4807.038,N	földrajzi szélesség (48° 07.038',É)
01131.000,E	földrajzi hosszúság (11° 31.000', K)
1	Észlelés típusa (jóság): 0 = érvénytelen, 1 = GPS észlelés (SPS) , 2 = DGPS észlelés, 3 = PPS észlelés, 4 = RTK észlelés, 5 = Float (lebegő) RTK, 6 = értékelt (számítás nélkül), 7 = manuális bevitel, 8 = szimuláció
08	műholdak száma
0.9	HDOP
545.4,M	tengerszint feletti magasság [m]
46.9,M	geoid-ellipszoid (WGS-84) távolság
(üres karakter)	az utolsó DGPS frissítés óta eltelt idő [s]
(üres karakter)	DGPS állomás ID (azonosító)
*47	checksum adat (minden esetben * az első karakter)

A 'GLL' kódot tartalmazó mondat a szélességi és hosszúsági adatokra vonatkozó értékeket adja meg.
$GPGLL,4916.45,N,12311.12,W,225444,A,*31

GLL	geográfiai helyzet, földrajzi szélesség és hosszúság
4916.46,N	földrajzi szélesség (49° 16.45', É)
12311.12,W	földrajzi hosszúság (123° 11.12', NY)
225444	észlelés ideje (22:54:44 UTC)
A	adatok státusza (aktív vagy érvénytelen)
*31	checksum

A SAJÁT NMEA ADATAINK BEOLVASÁSA

Első feladatként olvassuk be az NMEA adatokat Octave-ba (vagy Matlabba)! Töltsük le a hb_nmea.txt fájlt az oktatas.epito.bme.hu oldalról.

Ebben a fájlban csak $GPGLL kezdetű sorok vannak. Pl.

$GPGLL,5156.9051,N,00117.1178,E*69

Egy sor maximum 80 karakter lehet. Meghatározott hosszúságú mezők vannak vesszővel elválasztva, elég könnyen beolvasható, feldolgozható fájl. 5156.9051,N = 51° 56.9051', É-i szélesség, 00117.1178,E = 1° 17.1178', K-i hosszúság. A földrajzi szélességnél az első két karakter a fok érték, utána perc, a hosszúságnál az első 3 karakter a fok érték, utána perc (mivel előbbi -90-+90, utóbbi -180-+180-ig terjed). Szélességnél N (Észak) pozitív, S (Dél) negatív, Hosszúságnál E (Kelet) pozitív, W (Nyugat) negatív.

Olvassuk be a meghatározott formátum szerint az fscanf paranccsal:

```
> clear all; clc; close all;
> page_screen_output(0);
>
> fid = fopen('hb_nmea.txt');
> mat = fscanf(fid,'$GPGLL,%2d%f,%1s,%3d%f,%1s*%*2s\r\n', [6 inf]);
> fclose(fid);
> mat = mat'
```

A beírt szöveget ($GPGLL,) átugorja, utána 2 karakteres egész szám (%2d), majd egy valós szám (%f), utána egy betű (%1s), majd 3 karakter egész szám (%3d), majd egy valós szám (%f), egy karakter (%1s), majd * és utána kihagy 2 karaktert (%*2s) a sorvége jel előtt. A % utáni értékeket elmenti a mat nevű tömbbe, 6 sorba és előre meg nem határozott számú oszlopba (amennyi adat van). Transzponáljuk a mátrixot, hogy az egy ponthoz tartozó adatok egy sorba kerüljenek egy oszlop helyett. Az fscanf csak számokat képes tárolni egy tömbben, a textscan-nel ellentétben, ami cellatömbben többféle adatot is tárolhat, így az N/S, E/W (Észak/Dél, kelet/Nyugat) betűk is csak az ASCII kódjukkal lesznek eltárolva. Az ASCII kódokat lekérdezhetjük, pl. double('N') paranccsal. N = 78, S = 83, E = 69, W = 87. Fordítva pedig a char paranccsal kérdezhetjük le, hogy egy adott kódhoz milyen betű tartozik pl. char(78).

Válogassuk le a szélesség, hosszúság adatokat, alakítsuk tizedfokká, és ha déli szélesség (S) vagy nyugati (W) hosszúság szerepel benne, akkor szorozzuk meg (-1)-gyel az adott értéket. Ezt megoldhatjuk a szokásos módon 'for' ciklust és feltételt használva, de kihasználhatjuk hogy a Matlabot elsősorban mátrix/vektor műveletekre optimalizálták. Ez a 'vektorizálás' általában rövidebb, könnyebben áttekinthető és gyorsabb kódot eredményez. Nézzük meg a szélességeknél a hagyományos módon ciklussal, a hosszúságnál pedig vektorizációval.

```
> lat = mat(:,1) + mat(:,2)/60.0;
> for i=1:size(mat,1)
>   if mat(i,3)==double('S');
>     lat(i) = lat(i) * (-1);
>   end
> end
>
> long = mat(:,4) + mat(:,5)/60.0;
> % Ciklus nélkül megoldva ugyanez a hosszúságra vektorizációval
```

```
> % Megkeressük azokat az indexeket, amikre igaz a feltétel
> ii = mat(:,6) == double('W');
> % és az adott indexűeket megszorozzuk (-1)-gyel.
> long(ii) = long(ii) * (-1.0);
```

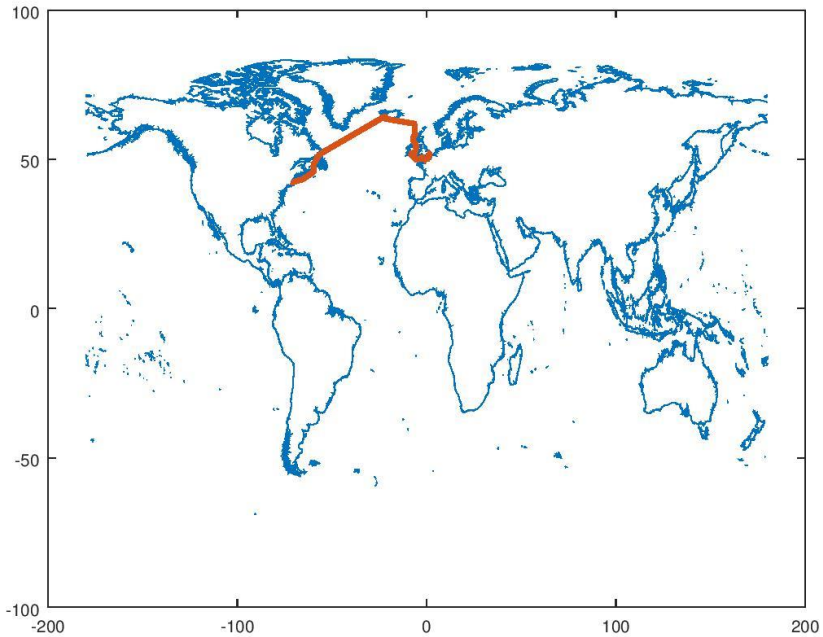
A formázott szövegként történő beolvasás helyett választhatjuk a soronkénti beolvasást is. Így a következőképp tudjuk előállítani a szélesség, hosszúság értékeket.

```
> %Adatok beolvasása méslépp, soronként, utólagos feldolgozással,
> % ha ismnerjük, hogy mi hányadik karakter
> fid = fopen('hb_nmea.txt');
>
> %Olvassuk be ciklussal az összes elemet,
> % és tároljuk a lat1, long1 vektorban a szélesség, hosszúság értékeket
> lat1 = []; long1 = [];
> while feof(fid)==0
>   line = fgetl(fid);
>   fi_fok = line(8:9);   fi_perc = line(10:16);
>   fi = str2num(fi_fok)+str2num(fi_perc)/60;
>   NS = line(18);   if NS=='S'; fi=fi*-1; end;
>   lambda_fok = line(20:22);  lambda_perc = line(23:29);
>   lambda = str2num(lambda_fok)+str2num(lambda_perc)/60;
>   EW = line(31);   if EW=='W'; lambda=lambda*-1; end;
>   lat1 = [lat1; fi]; long1 = [long1; lambda];
> end
> fclose(fid)
```

AZ NMEA ADATAINK ÁBRÁZOLÁSA

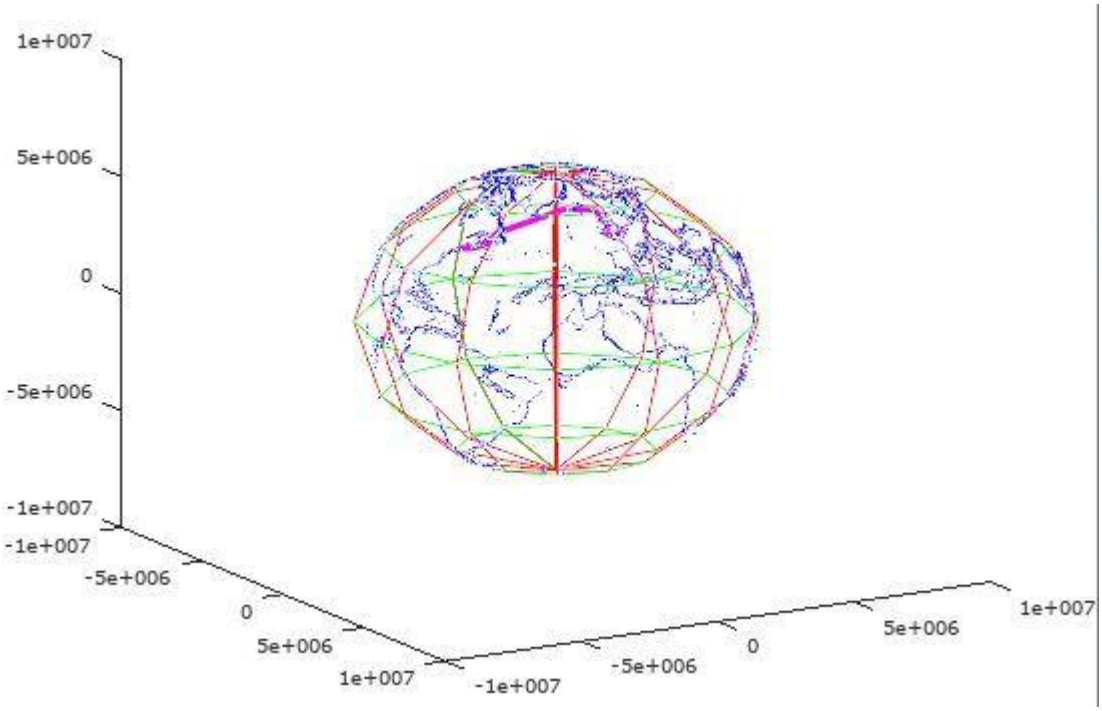
Ábrázoljuk az adatokat a korábbi 2D ábránkba és mentsük el az eredményt!

```
> figure(1); hold on;
> plot(long, lat,'LineWidth',3);
> save longlat.mat long lat
```



Rajzoljuk be a 3D ábránkba is!

```
> %útvonal pontjai xyz koord. rendszerben
> zu = R*sind(lat);
> pu = R*cosd(lat);
> xu = pu.*cosd(long);
> yu = pu.*sind(long);
> % paralelkörök ábrázolása 3D-ben
> figure(2);
> plot3(xu, yu, zu, 'm','LineWidth',3);
```



Lássuk be a fenti ábra nem igazán látványos, nehéz lenne ezzel dicsekedni a transzatlanti nyaralásunkkal. Jobb lenne, ha sokak által ismert és használt Google Earth-be tudnánk felrakni az adatainkat. A Google Earth formátuma a KML. Ismerkedjünk meg egy kicsit vele.

KML FORMÁTUM

A KML vagy Keyhole Markup Language egy földrajzi jellemzők, például pontok, vonalak, képek, sokszögek és megjelenítési modellek tárolására és modellezésére szolgáló XML fájlformátum a Google Föld, a Google Térkép és egyéb alkalmazásokban. A KML segítségével helyeket és információkat oszthat meg ezeknek az alkalmazásoknak a többi felhasználójával.

Egy KML fájlt a Google Föld hasonlóan dolgoz fel ahhoz, ahogy a webböngészők feldolgozzák a HTML és XML fájlokat. HTML-hez hasonlóan a KML is névvel és attribútumokkal rendelkező jelölőket alkalmaz a meghatározott megjelenítési céljaira. Ily módon a Google Föld a KML fájlok böngészőjeként viselkedik. (lásd: https://support.google.com/earth/answer/148118?hl=hu)

Egy egyszerű mintapélda (Bodroginé Dr.Zichar Marianna: KML.pdf, https://w1.inf.unideb.hu/web/noir/gis, Josie Wernecke: The KML handbook alapján).

Megjegyzés sor: <!-- tetszőleges szöveg - ->

HelloFold.kml

```
<?xml version="1.0" encoding="utf-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Helló Világ</name>
    <description>
    <![CDATA[
      <p><b>Itt fejlesztik a Google Earth-t!</b> </p>
    ]]>
    </description>
    <Point>
      <coordinates>
        -122.084583,37.42227,0
      </coordinates>
    </Point>
  </Placemark>
</kml>
```

SimpleLineString.kml

```
<?xml version="1.0" encoding="utf-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name>Simple LineString</name>
    <Placemark>
      <name>Juan de Fuca Plate</name>
      <LineString>
        <coordinates>
          -130.597293,50.678292,0
          -129.733457,50.190606,0
          -130.509877,49.387208,0
          -128.801553,48.669761,0
          -129.156745,47.858658,0
          -128.717835,47.739997,0
        </coordinates>
      </LineString>
    </Placemark>
  </Document>
</kml>
```

ADATOK KML FORMÁTUMBA MENTÉSE

Látjuk, hogy a legegyszerűbb KML fájlok (pontok, egyszerű vonalláncok) nem túl bonyolultak, viszonylag könnyen elő lehet állítani akár Matlabot/Octave-ot használva is, ha ismerjük a fájl specifikációt. Ha már vonalstílust, pontstílust, egyebeket akarunk állítani, akkor már kicsit nehezebb, utána kell nézni alaposabban.

Viszont koordináta listából KML fájlt készíteni egy meglehetősen gyakran felmerülő feladat. Könnyen lehet, hogy vannak erre már kész matlab megoldások. Ilyenkor lehet érdemes esetleg körülnézni a neten (pl. a Matlab Central-on, ami egy Matlab felhasználói közösség: http://www.mathworks.com/matlabcentral/), ahelyett, hogy rögtön nekiállnánk megírni a saját matlab programunkat.

A Matlab Central oldalon válasszuk ki a File exchange menüt, majd keressünk rá a 'KML write' kifejezésre! Jó pár találatot kapunk, nézzük meg őket. Nekünk földrajzi koordinátákból kell egy vonalláncot készíteni. Használhatnánk akár a 'KML Toolbox'-ot Rafael Oliveirától, ami egy sokoldalú KML eszköztár, de a mi feladatunkhoz pont megfelelő a 'kml line plot' program Cameron Sparr-tól, ami lényegesen egyszerűbben kezelhető és pont azt a feladatot végzi, amire szükségünk van (Draw nan-separated lines (or a single line) onto Google Earth.). A Matlab Central oldaláról regisztráció után lehet letölteni fájlokat (érdemes esetleg a 'geodetic' kulcsszóra is rákeresni hasznos függvényekért!). Hogy most ne kelljen mindenkinek regisztrálni, az oktatas.epito.bme.hu oldalra feltöltöttem a kml_line.zip fájlt. Töltsük le és tömörítsük ki, majd nyissuk meg a kml_line.m fájlt.

Nézzük meg a specifikációt:

```
% KML_LINE    Draw nan-separated lines (or single line) onto Google Earth.
%
% Syntax:
%     KML_LINE(LON, LAT) writes nan-separated lines specified
%     in LON and LAT to an output file, doc.kml
%     KML_LINE(LON, LAT, NAME) writes nan-separated lines specified
%     in LON and LAT to an output file, NAME.kml
%     KML_LINE(LON, LAT, NAME, COLOR) writes lines same as above but with
%     MATLAB color value (default is 'w').
%     KML_LINE(LON, LAT, NAME, WIDTH) writes lines same as above but with
%     width WIDTH (default is 1).
%     KML_LINE(LON, LAT, NAME, COLOR, WIDTH) writes lines same as above but
%     with width WIDTH and color COLOR.
%
% Input:
%     LON: 1-D array of longitude line values. Separate lines are separated
%     by a NaN. Must correspond to LAT.
%     LAT: 1-D array of latitude line values. Separate lines are separated
%     by a NaN. Must correspond to LON.
%     NAME: String name of output file (without the .kml extension)
%     COLOR: MATLAB color (default is 'w'), supports vector colors.
%     WIDTH: int or float width value (default is 1)
%
```

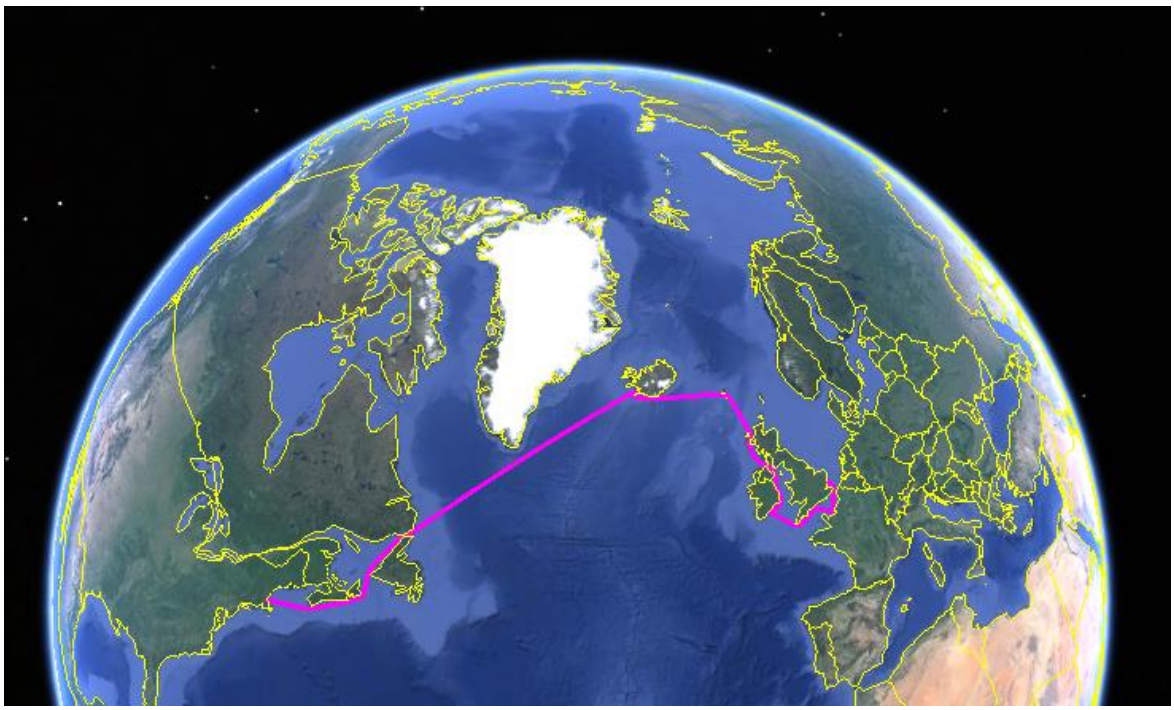
```
% Output:
%     This function creates a kml file called NAME.kml in the current
%     working directory
%
% Examples:
%     % four different ways of calling kml_line:
%     load('palau_coastline.mat');
%     kml_line(lon_coast, lat_coast, 'palau_coastline');
%     kml_line(lon_coast, lat_coast, 'palau_coastline', 'r');
%     kml_line(lon_coast, lat_coast, 'palau_coastline', 1.5);
%     kml_line(lon_coast, lat_coast, 'palau_coastline', 'magenta', 2);
%
```

```
%
% Cameron Sparr - Nov. 31, 2011
% cameronsparr@gmail.com
%
```

Mi most a legutolsó hívási módot fogjuk választani, hogy színt és vonalvastagságot is megadhassunk:

```
> kml_line(long, lat, 'hajokazas', 'magenta', 3);
```

Ezután Google Earth-be töltsük be a hajokazas.kml fájlt, és nézzük meg merre is volt a nyaralás!



Átkelés útvonala az Atlanti Óceánon Harwichtól Bostonig luxushajóval.

MAPPING ÉS GEOMETRY CSOMAG UTASÍTÁSAI

A különböző csomagok listája, telepítése, betöltése:

```
pkg list % csomagok listázása
% mapping csomag telepítése internetről közvetlenül, ha nincs telepítve
% pkg install -forge mapping;
pkg describe -verbose mapping % a mapping csomag parancsainak listája
pkg load mapping; % mapping csomag betöltése
```

MAPPING PACKAGE

Ha már a neten elérhető megoldásokról beszélünk, akkor érdemes lehet kicsit belenézni az Octave Mapping és Geometry Package utasításaiba is, hogy mi az, amit mi is használni tudunk belőle. Néhányat már láttunk, hogyan lehet fok-perc-másodpercből tizedfokba és vissza váltani, fokból radiánba és vissza stb. (deg2rad;degtorad;rad2deg;radtodeg; degrees2dm; degrees2dms; dm2degrees; dms2degrees; függvények).

Milyen egyéb számunkra érdekes függvényei vannak még a mapping csomagnak?

Lehet a **gömbön** szögeket, ívhosszakat számítani, ami vetülettanból lehet ismerős.

azimuth: Azimut számítása a legnagyobb gömbi körön P_1 és P_2 között.

distance: távolság a legnagyobb gömbi kör mentén P_1 és P_2 között és opcionálisan az azimut is.

reckon: Első geodéziai alapfeladat a gömbön: adott pontból azimut és gömbi távolság (az ívhez tartozó középponti szög) megadásával a végpont koordinátáinak kiszámítása.

km2deg, km2rad, deg2km, rad2km: Gömbi távolságok középponti szögekké alakítása és vissza. Az alapértelmezett sugár 6371 km, de meg lehet adni tetszőleges sugarat, vagy a holdat, Marsot, és egyéb bolygókat is.

Mértékegységek közti átváltások:

km2sm, km2nm, nm2km, nm2sm, sm2km, sm2nm : Átváltás kilométerből mérföldbe, tengeri mélföldbe és vissza.

unitsratio: átváltási arány különböző mértékegysége között, pl. km, cm, láb, hüvelyk

wrapTo180, wrapTo360, wrapToPi, wrapto2Pi: szögek átalakítása a -180-+180 fok tartományra, vagy a 0-360 fok tartományra, vagy ugyanez radiánban.

Ezenkívül shape fájlok, rasztre fájlok olvasása, írása.

GEOMETRY PACKAGE

Ebben rengeteg utasítás van, most csak néhány fontosabb parancsot nézzünk meg:

centroid – súlypont számítás ponthalmazra

distancePoints – pontok közötti távolság minden kombinációban

findClosestPoint – legközelebbi pont indexének és távolságának megkeresése egy pontfelhőhöz képest

isPointInCircle – adott körön belül van-e a pont?

midPoint – középpont egy szakaszon

minDistancePoints – Legkisebb távolság sok pont között

polarPoint – Poláris pont számítás (első geodéziai alapfeladat) (mért távolság + irányszög)

angle2Points – 2 ponttal adott egyenes x tengellyel bezárt szöge

angle3Points – Térbeli szög kiszámítása P1, P2 és P3 között.

distancePointLine – Minimális távolság pont és egyenes között

isParallel – Két vektor párhuzamosság vizsgálata

isPerpendicular – Két vektor merőlegesség vizsgálata

pointOnLine – Pont létrehozása egyenesen adott távolságra

transformPoint – Pont transzformáció affin transzformációval

fillPolygon – Pontlista által megadott polygon kitöltése

A fenti csak néhány példa ebből a csomagból, ezeken kívül sok más is van a csomagban, például különböző tarnszformációk, körökhöz, ellipszisekhez, vonalakhoz, poligonokhoz tartozó parancsok 2D-ben, 3D-ben. Pl metszések, legrövidebb távolságok.

ELSŐ ÉS MÁSODIK GEODÉZIAI ALAPFELADAT A SÍKON

```
%% Második geodéziai alapfeladat
pkg load geometry;
pkg load mapping;
A = flip([657705.45 247565.56]);
B = flip([658310.44 248489.88]);
C = flip([658604.69 247832.58]);
D = flip([658077.70 247431.38]);
deltaAC = geodszog(rad2deg(angle2Points(A,C)))
dAC = distancePoints(A,C)
deltaBD = geodszog(rad2deg(angle2Points(B,D)))
dBD = distancePoints(B,D)

%% Első geodéziai alapfeladat
A = flip([657705.45 247565.56]);
tavAC = 938.05
deltaAC = deg2rad(dms2degrees([73 27 42]))
C_polar = flip(polarPoint(A, tavAC, deltaAC))
```

Eredmények:

```
deltaAC =  73-27-42.3313
dAC =  938.047044662461
deltaBD = 192-24-2.5431
dBD =  1083.78510674395

tavAC =  938.050000000000
deltaAC =  1.28214795733590
C_polar =

   658604.692404206   247832.582285545
```

Mint látjuk megoldható az első és a második geodéziai alapfeladat beépített függvényekkel, de be kell hozzá olvasni két csomagot is (mapping a rad2deg függvényhez és geometry a az angle2Points függvényhez) és figyelembe kell venni, hogy a matematikai és a geodéziai koordináta rendszerek eltérnek, ezért a flip paranccsal meg kell cserélni Y,X sorrendjét. Ebben az esetben lehet, hogy egyszerűbb, ha saját magunknak írunk egy távolság és irányszög számító függvényt, mint, ahogy ezt már egyszer meg is tettük a 2. gyakorlaton:

tav_irany.m fájl:

```
function [t delta] = tav_irany(y1,x1,y2,x2)
      dy = y2 - y1;
      dx = x2 - x1;
      t = sqrt(dy^2+dx^2);
      delta = atan2d(dy,dx); % Octave esetén
      %delta = atan2(dy,dx); % Matlab esetén
      %delta = radtodeg(delta); % Matlab esetén
      if delta<0 delta=delta+360; end
end;
```

Használata:

```
%% Második geodéziai alapfeladat saját függvénnyel
A = [657705.45 247565.56];
C = [658604.69 247832.58];
[dAC deltaAC] = tav_irany(A(1),A(2),C(1),C(2))
deltaAC = geodszog(deltaAC)
```

VETÜLETTAN - ELSŐ ÉS MÁSODIK GEODÉZIAI FŐFELADAT A GÖMBÖN

```
page_screen_output(0);clc; clear all; close all;
pkg load mapping % mapping csomag betöltése
%% Vetülettan - első és második geodéziai főfeladat a gömbön
% Kiinduló adatok
fiA = dms2degrees([47 13 58.4976]);
lA = dms2degrees([-1 3 14.6857]);
fiC = dms2degrees([47 15 41.3749]);
lC = dms2degrees([-2 14 27.8952]);
alfaAB = dms2degrees([223 44 26.3670]);
sAB = 222342.613;
R = 6379743.001;

% Első geodéziai főfeladat
% Adott [fiA, lambdaA], az azimut (alfaAB) és az ívhossz (sAB)
% Keresett [fiB, lambdaB]
tetaAB = km2deg(sAB/1000, R/1000) % Ívhosszból középponti szög
disp(degrees2dms(tetaAB)) % megjelenítés fok-perc-mp
% megjelenítés fok-perc-mp saját függvénnyel geodéziai formában
disp(geodszog(tetaAB))
[fiB, lB] = reckon(fiA,lA,tetaAB,alfaAB) % első geodéziai főfeladat
disp(geodszog(fiB))
disp(geodszog(lB))
alfaBA = azimuth(fiB,lB,fiA,lA) % ellentett irány azimutja
disp(geodszog(alfaBA))

% Második geodéziai főfeladat
% Adott [fiA, lambdaA] és [fiC, lambdaC]
% Keresett az azimut (alfaAC) és az ívhossz (sAC)
[tetaAC alfaAC] = distance(fiA,lA,fiC,lC) % második geodéziai főfeladat
disp(geodszog(alfaAC))
alfaCA = azimuth(fiC,lC,fiA,lA) % ellentett irány azimutja
disp(geodszog(alfaCA))
format long;
sAC = deg2km(tetaAC, R/1000)*1000 % középponti szögből ívhossz
```

Eredmények:

```
tetaAB =  1.99683487701836
    1.00000000000000   59.00000000000000   48.60555726610255
  1-59-48.6056
fiB =  45.7728784819162
lB = -3.03341788401119
 45-46-22.3625
 -4-57-59.6956
alfaBA =  42.3046429670264
 42-18-16.7147

tetaAC =  0.806279583720703
alfaAC =  272.466880671765
272-28-0.7704
alfaCA =  91.5952628357265
 91-35-42.9462
sAC =  89777.2327177581
```