Technische Universität München

Orbit Mechanics

Keplerian Orbits in Space-fixed, Earth-fixed and Topocentric systems

Lucía Placer López

4 Dezember 2017

Índice

1. Given data

Orbits of several satellites are given in an inertial, geocentric reference system (space-fixed) by the Keplerian orbital elements: semi major axis a, eccentricity e, inclination i, right ascension of the ascending node Ω, argument of the perigee ω, and perigee passing time T_0 on Nov. 13, 2017.

Satellite	**a [km]**	**e**	**i** [deg]	Ω [deg]	ω [deg]	T_0 [h]
GOCE	6629	0.004	96.6	210	144.2	02:00
GPS	26560	0.01	55	30	30	11:00
MOLNIYA	26554	0.7	63	200	270	07:30
GEO	geostationary	0	0	0	50	00:00
MICHIBIKI	geosynchronous	0.075	41	200	270	04:10

For the following computations precession, nutation, polar motion and variations in the length of day are neglected. The Earth fixed reference system then rotates with an angular rate of $\omega_{Earth} = 2\pi/86164$ s about the e3-axis of the inertial space-fixed reference system. At the time $t_0 =$ Nov. 13, 2017, $00:00$ the sidereal angle is 03h 29m.

2. Problem 1

Create a MATLAB-function *kep2orb.m* that computes polar coordinates r (radius) and ν (true anomaly) based on input orbital elements. Formulate your program in a way that the time t can be used as input parameter.

```
function [r, v, M, E] = kep2orb(a, e, t_0, t)
```

The purpose of the function is to calculate the radius and the true anomaly:

$$r = a \cdot (1 - e \cdot \cos E) \tag{1}$$

$$\tan\frac{\nu}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2} \longrightarrow \nu = 2\arctan\left(\sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2}\right) \tag{2}$$

We already have the semi mayor axis of each satellite and the eccentricity, but we need to calculate the eccentric anomaly (E).For that we need an iteratively solve of Kepler's equation with an iterative method:

$$E_0 = M \tag{3}$$

where M is the mean anomaly and it is calculated with this equation:

$$M = n(t - T_0) \tag{4}$$

where t and T_0 are given values. But we need to calculate the mean motion (n):

$$\mu = n^2 \cdot a^3 \longrightarrow n = \sqrt{\frac{\mu}{a^3}} \tag{5}$$

where $\mu = GM =$ 389,6005x10^{1}2 m^3/s^2 (for elliptic orbits and Earth satellites).

For large eccentricities Kepler's equation is:

$$\Delta E_i = \frac{M + e \sin E_i - E_i}{1 - e \cos E_i} \tag{6}$$

$$E_{i+1} = E_i + \Delta E_i \tag{7}$$

In the function we compute the position of the satellite for each time step. First we compute the true anomaly for the current time, and after that we assume $E_i = M$ to initialize the eccentric anomaly and calculate ΔE_i. If this value becomes less or equal to 10^{-6}, the loop ends and the value of E_i is the one that is used to calculate the radius and the true anomaly.

3. Problem 2

Plot the orbit for the 5 satellites in the orbital plane for one orbital revolution.

The eccentricity of the given satellites is less than zero, that means that all orbits are elliptical, so the previous function *kep2orb.m* can be used.

First of all we need to calculate the time period:

$$T = 2\pi \cdot \sqrt{\frac{a^3}{\mu}} \tag{8}$$

Where μ as we said before is the standard gravitational parameter of the Earth.

For this plot we need the semi mayor axis a, the eccentricity e, the gravitational parameter u, the time period t and the results of the first exercise.

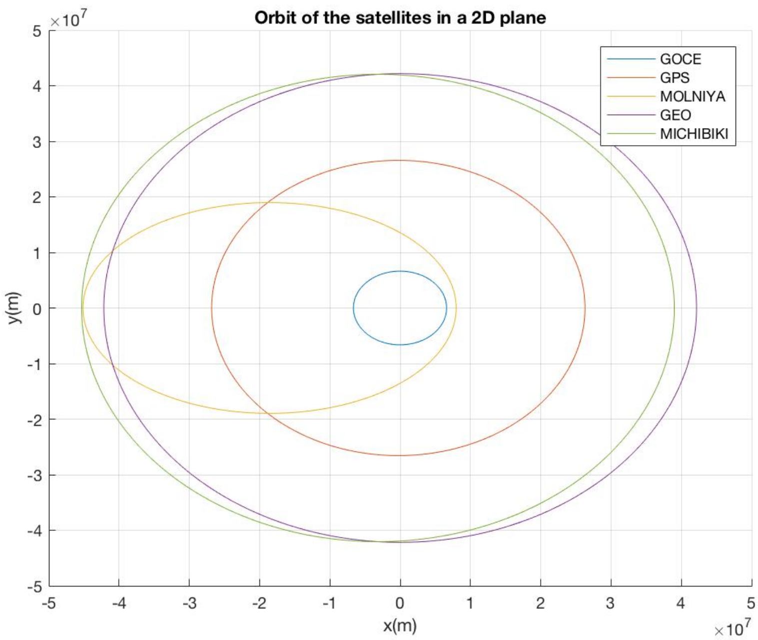
Now we move from polar coordinates to Cartesian coordinates.

The true anomaly is the angular parameter that defines the position of the satellite moving along the orbit. It is the angle between the direction of periapsis and the current position of the body.

So with the radius of the orbit and the true anomaly at any time t we have a projection on x and y axis.

$$x = r\cos(\nu) \qquad\qquad y = r\,\mathrm{sen}(\nu) \tag{9}$$

We calculate the position for each x and y of the satellites in the plane from t=0 to t=T, that is the period of the Earth, 24 hours.



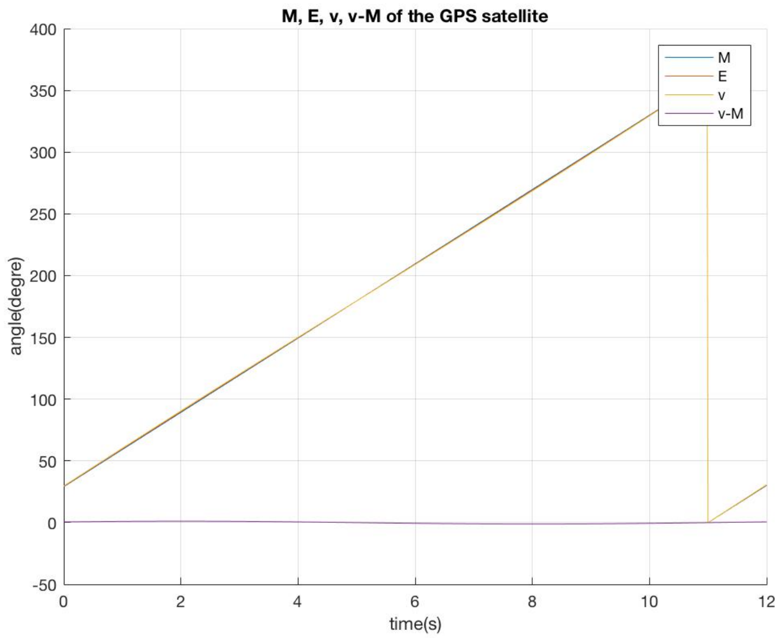
Looking at the plot we can confirm that the given satellites have elliptic orbits, what is seen is the protection of the orbits on Earth's equatorial plane.

4. Problem 3

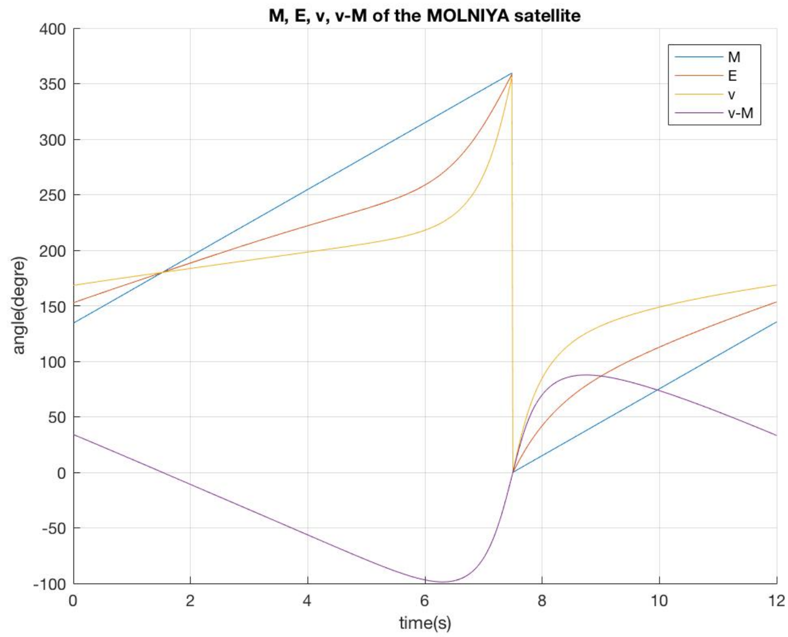
Plot the mean anomaly M, the eccentric anomaly E, and the true anomaly ν as well as the difference $\nu - M$ for one orbital revolution for the GPS satellite and the Molniya satellite.

In these two graphics we can compare the parameters between the two satellites.

In the GPS satellite we can notice that the difference between the values are negligible and that $\mu - M = 0$, due to the fact that the eccentricity of the GPS satellite is almost 0 and the orbit is almost circular.



In the other hand, the Monliya satellite has a eccentricity of $e = 0{,}7$ so it is the most eccentric satellite of our study. In the graphics we cans see the deference between the anomalies that are directly proportional to the eccentricity.



5. Problem 4

Create a MATLAB-function kep2cart.m that uses kep2orb.m, which transforms Keplerian elements to position and velocity in an inertial (space-fixed) system.

```
function [rr, dotrr] = kep2cart(a, e, i, raan, omega, t_0,
    t)
```

Where *rr* is the position in space fixed coordinates and *dotrr* is the velocity in space fixed coordinates.

With *kep2orb.m* we obtained the position and the velocity in a space-fixed inertial system. We calculate the components of position vector and velocity vector using the following equations respectively:

$$\vec{r_{2'}} = r\left\{\cos(\nu), \sin(\nu), 0\right\} \tag{10}$$

$$\dot{\vec{r_{2'}}} = \sqrt{\frac{\mu}{a(1-e^2)}}\left\{-\sin(\nu), \cos(\nu)+e, 0\right\} \tag{11}$$

We have the two vectors so we rotate them using the right ascension of the ascending node (*raan*), inclination (*i*) and argument of the perigee (*omega*) to get position and the velocity in the new system.

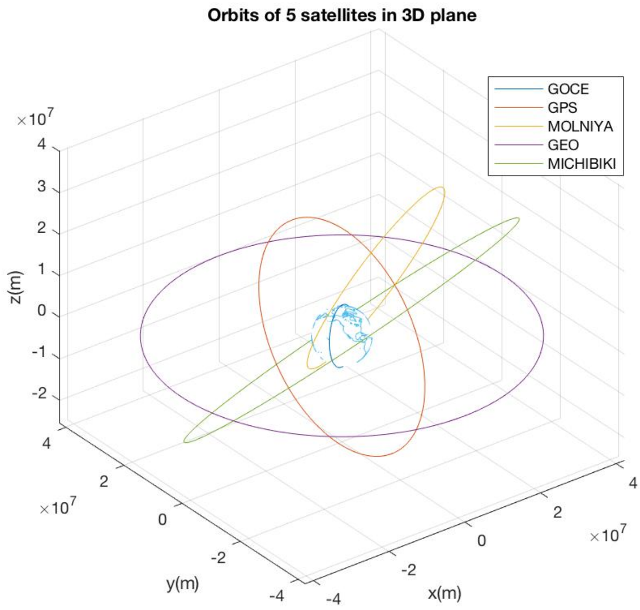
$$\vec{r_2} = R_3(-\Omega)R_1(-i)R_3(-\omega)\vec{r_{2'}} \tag{12}$$

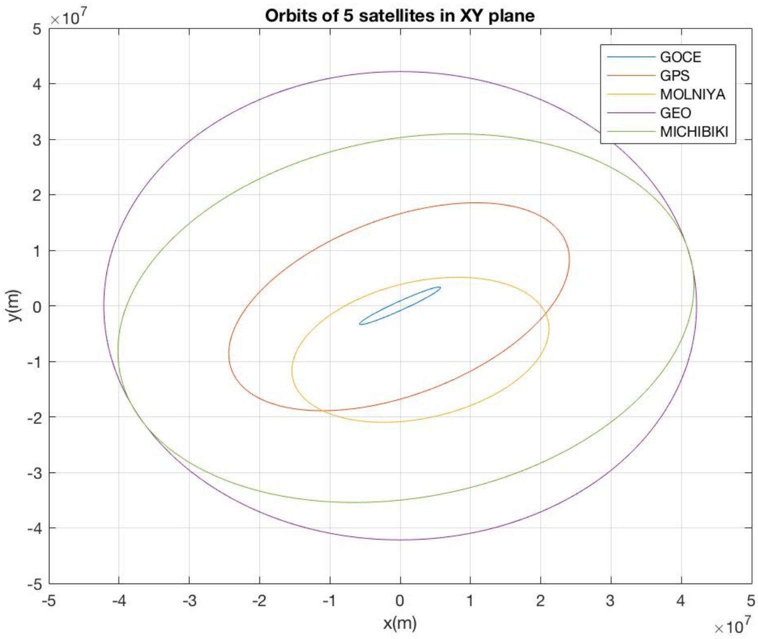
$$\dot{\vec{r_2}} = R_3(-\Omega)R_1(-i)R_3(-\omega)\dot{\vec{r_{2'}}} \tag{13}$$

6. Problem 5

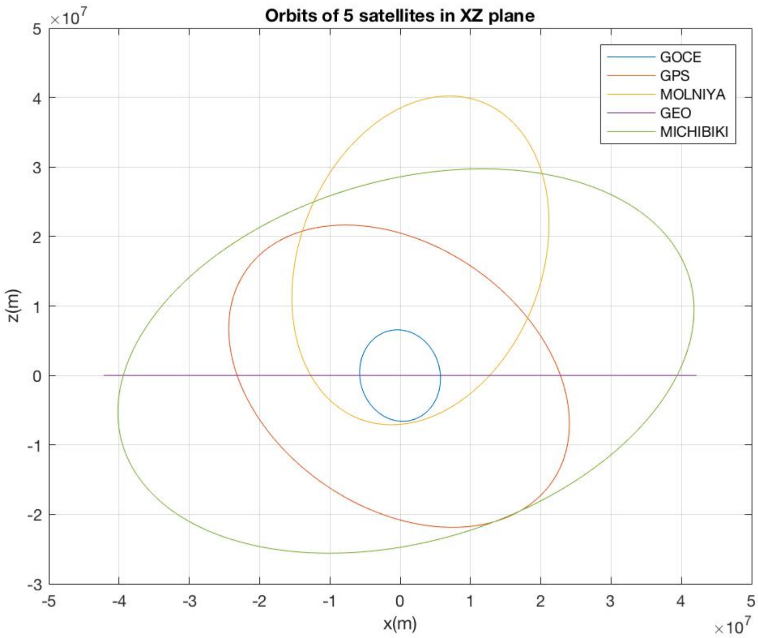
Compute position and velocity vectors of the 5 satellites for a period of one day. Visualize your results. Plot the trajectory in 3D and 2D (projection to $x-y$, $x-z$ and $y-z$ planes) as well as a time series of the magnitude of velocity.

For the plots we are using the previous function to get the arrays of the position and velocity.

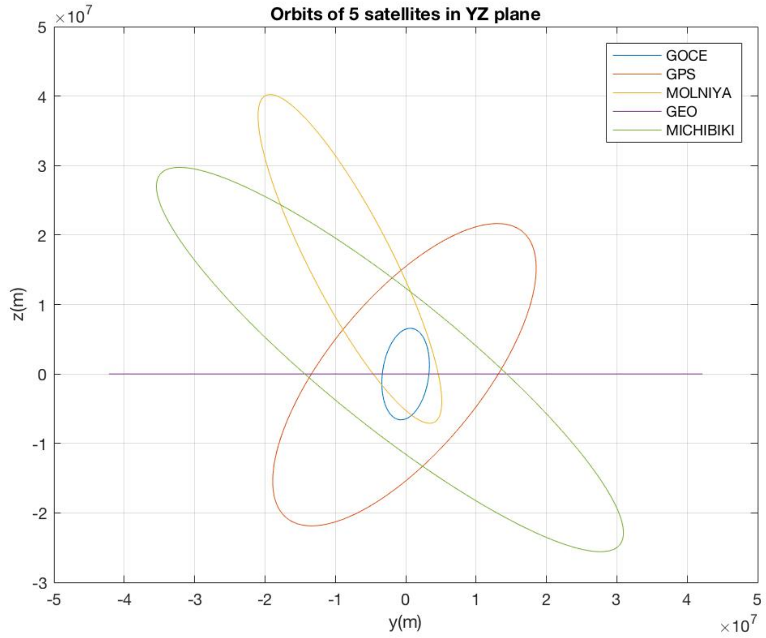




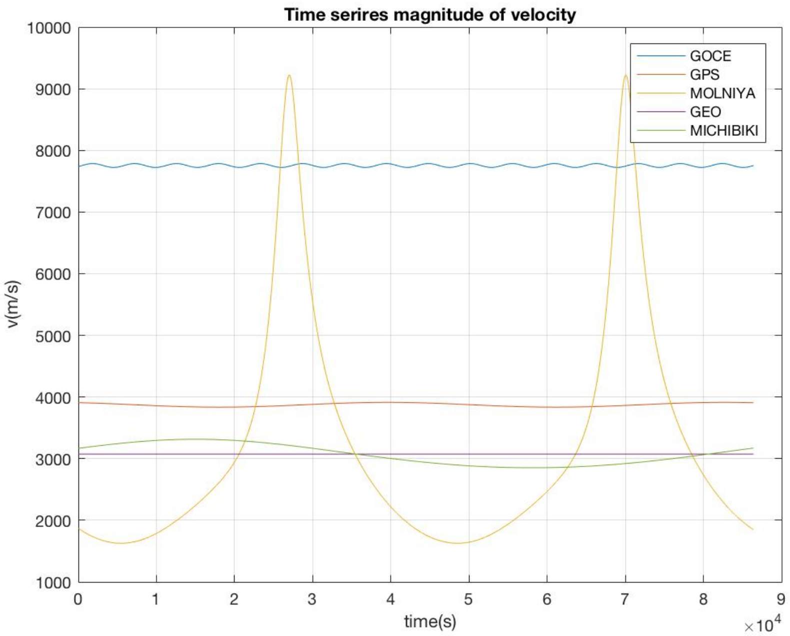
Orbits of 5 satellites in XY plane
GOCE
GPS
MOLNIYA
GEO
MICHIBIKI
y(m)
x(m)
×10^7



Orbits of 5 satellites in XZ plane
GOCE
GPS
MOLNIYA
GEO
MICHIBIKI
z(m)
x(m)
×10^7



The last plot shows the time regarding the speed. Here we can mention that with the exception of the Molniya, the speed remains more or less constant over time because they have low eccentricities. But the orbit of the Molniya is much more eccentric, so when is close to the Earth the velocity increases and when is far away the velocity decreases, this is how Kepler's third law is fulfilled.



7. Problem 6

Create a MATLAB-function cart2efix.m that transforms position and velocity in a spacefixed system into position and velocity in an Earth-fixed system.

```
function [rrr, dotrrr] = cart2efix(rr, dotrr, t)
```

Where *rrr* is the position in Earth fixed coordinates and *dotrrr* is the velocity in Earth fixed coordinates.

For this problem we just need to rotate the space fixed coordinates to earth fixed coordinates. At first the Earth's rotation angle at each epoch is calculated:

$$\dot{\Omega}_e = \frac{2\pi}{86164} \tag{14}$$

The rotation angle is:

$$\theta_0(t) = \dot{\Omega}_e t + (30 + 29/60) \cdot 15 \tag{15}$$

So the position is:

$$\vec{r}_3(t) = R_3(\theta_0(t))\vec{r}_2(t) = \begin{bmatrix} \cos\theta_0(t) & \sin\theta_0(t) & 0 \\ -\sin\theta_0(t) & \cos\theta_0(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{r}_2(t) \tag{16}$$

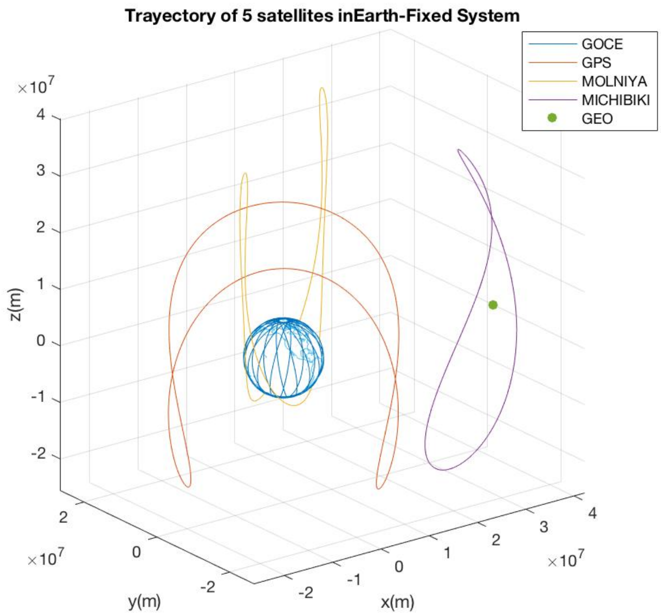
And the velocity:

$$\vec{v}_3(t) = R_3(\theta_0(t))\vec{v}_2(t) = \begin{bmatrix} \cos\theta_0(t) & \sin\theta_0(t) & 0 \\ -\sin\theta_0(t) & \cos\theta_0(t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{v}_2(t) \tag{17}$$

8. Problem 7

Plot the trajectory of the satellites in 3D for the first two orbital revolutions.

For the plot we have to create the arrays for the position and velocity given by *cart2efix.m*. Then we compute for each satellite the position in the 3 axis.



9. Problem 8

Calculate and draw the satellite ground-tracks on the Earth surface.

Now that we have the position and velocity in Earth fixed system, to draw the groud-tracks we need the longitude and the lalitude.

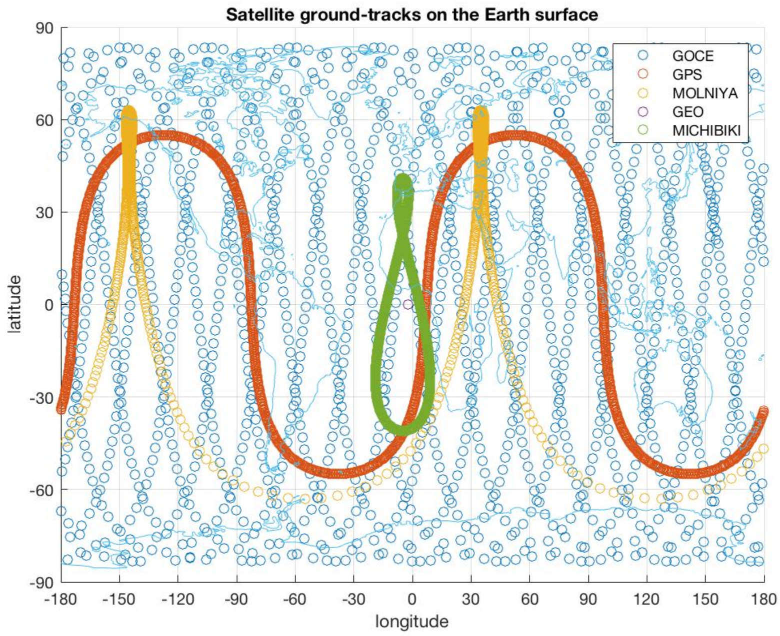
- Longitude: $\lambda \in [-\pi, \pi]$

$$\lambda = \arctan\left(\frac{y_3}{x_3}\right) \tag{18}$$

- Latitude: $\phi \in \left[\frac{-\pi}{2}, \frac{\pi}{2}\right]$

$$\phi = \arctan\left(\frac{z_3}{\sqrt{x_3^2 + y_3^2}}\right) \tag{19}$$

We made a loop to calculate the points of the latitude and longitude for each time t and for the five satellites.



We can appreciate that the GEO satellite is a point at it has the same period as the Earth.

10. Problem 9

Create a MATLAB-function efix2topo.m that transforms position and velocity in an Earthfixed system into position and velocity in a topocentric system centered at the station Wettzell which position vector in an Earth-fixed system is given by: $r_w = (4075{,}53022, 931{,}78130, 4801{,}61819)T$ km.

```
function [rrrr, azim, elev] = efix2topo(rrr, dotrrr, t)
```

Where $rrrr$ is the position in the topocentric system.

The position of the Wettzell station is given by the vector r_w. First, we calculate the latitude and longitude using equations (18) and (19). After that, we calculate the translated vector with the difference between the Wettzell vector and the r_3 vector.

After that we need to transform the topocentric vector form right handed system to left handed system with the following matrix:

$$Q = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{20}$$

So the final rotation to obtain the topocentric vector of the position is:

$$r_4 = Q \cdot R_2(90 - \phi_w) \cdot R_3(\lambda_w) \tag{21}$$

In the e topocentric systems we need the values of the azimuth and the elevation that are given by the next equations:

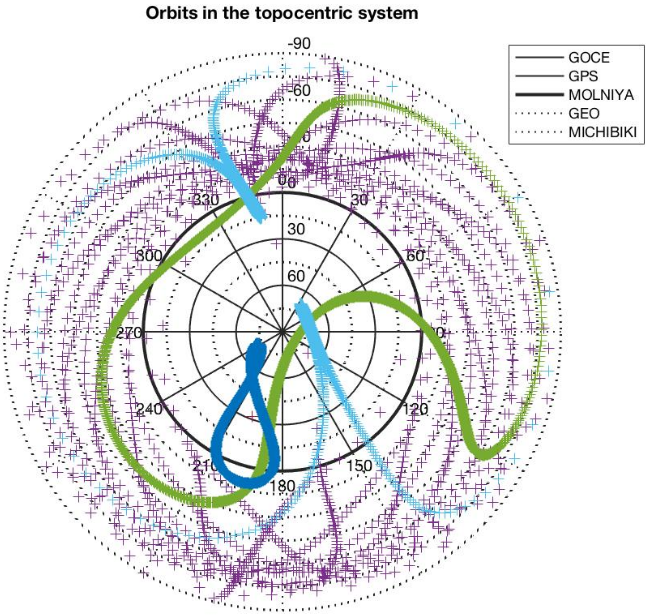
$$A = \arctan\left(\frac{y_4}{x_4}\right) \tag{22}$$

$$h = \arctan\left(\frac{z_4}{\sqrt{x_4^2 + y_4^2}}\right) \tag{23}$$

11. Problem 10

Plot the trajectory of the satellites as observed by Wettzell using the MATLAB-function skyplot.m.

Using the previous function we can plot the trajectories of the satellites, seen at Wetzell.

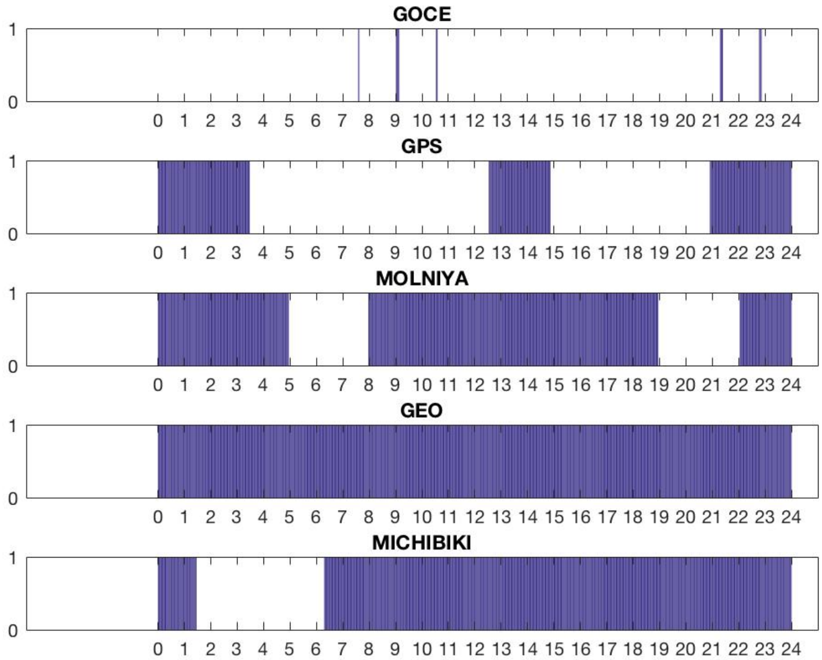


12. Problem 11

Calculate visibility (time intervals) for the satellites at the station Wettzell and visualize them graphically.

With this plot we can visualize the visibility of the satellites at any time, knowing the position. The location here is given by the Wetzell position vector. When the elevation calculated in the function is greater than zero the satellite is visible in the location, the visibility depends on the elevation.

In the plot we can also appreciate that the GEO satellite is completely visible all day because is a geostationary satellite, and opposite to that the GOCE is the least visible because is the closest to the Earth and cannot be seen at low elevation angles.


GOCE
GPS
MOLNIYA
GEO
MICHIBIKI
0
1
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

13. Matlab Code

kep2orb.m

```
function [r, v, M, E] = kep2orb(a, e, t_0, t)
% Description: compute polar coordinates r (radius) and v(true anomaly)
% based on input orbital elements
%
% Inputs: semi mayor axis, eccentricity, perigee passign time and time
%
% Outputs: radius, true anomally, mean motion and eccentric anomaly
%

%% Parameters

% Geocentric gravitational constant: u = GM
u = 3.986004418e+14;

% Mean motion
n = sqrt(u/(a^3));   % Kepler's 3rd law: u = (n^2*a^3)

% Tolerance for eccentric anomaly difference: diff = (E_1 - E_0)
tol = 1e-6;

nsteps = length(t);

% Allocate arrays for radius and true anomally.
% One for each time step
r = zeros(nsteps, 1);
v = zeros(nsteps, 1);
M = zeros(nsteps, 1);
E = zeros(nsteps, 1);


%% Time iteration

% For each time step compute the position of the satellite
for i=1:nsteps

    time = t(i);

    if (time - t_0) < 0
        % Compute true anomaly for current time [rad]
         M_1 = n * (time - t_0) + 2*pi;
    else
        M_1 = n * (time - t_0);
    end

    M_1 = mod(M_1, 2*pi);

    M(i, 1) = M_1;

    % Initialize eccentric anomaly [rad]
```

```
    E_0 = M_1;

    %Initialize difference [rad]
    diff = 1e10;

    %Newton iterarion
    while abs(diff) >= tol
        %Eccentric anomaly [rad]
        del_E = ((M_1 + e * sin(E_0)) - E_0) / (1 - (e * cos(E_0)));
        E_1 = E_0 + del_E;
        diff = E_1 - E_0;
        E_0 = E_1;
    end

    E(i, 1) = E_1;

    %True anomaly [rad]
    v(i, 1) = 2 * atan2(sqrt(1 + e) * sin(E_1/2), sqrt(1 - e) * cos(E_1
        /2));

    %Radius of satellite's orbit [m]
    r(i, 1) = a * (1 - e*cos(E_1));
end

%Outputs r and v have being assigned during time fordwarding for loop

end
```

kep2cart.m

```
function [rr, dotrr] = kep2cart(a, e, i, raan, omega, t_0, t)
%Description: this function transforms the keplerian elements to
    position
%and velocity in an inertial (space-fixed) system
%
%Inputs: semi mayor axis, eccentricity, inclination, right ascension
    of
%the asecendent node, argument of the perigee, perigee passing time
    and
%time
%
%Outputs: the position and velocity in and inertial space-fixed system
%

%Change of the keplerian elements into radians
raan = deg2rad(raan);
omega = deg2rad(omega);
i = deg2rad(i);

[r, v, ~, ~] = kep2orb(a, e, t_0, t);

nsteps = length(t);
```

```
rr = zeros(3, nsteps);
dotrr = zeros(3, nsteps);

rr(1, :) = r(:) .* cos(v(:));
rr(2, :) = r(:) .* sin(v(:));

u = 3.986004418e+14;  % Geocentric gravitational constant: u = GM
k = sqrt(u ./ (a .* (1 - e^2)));

dotrr(1, :) = -k .*  sin(v(:));
dotrr(2, :) = k .* (e + cos(v(:)));

%% Rotations

% First rotation of RAAN in z axis
R3_raan = [cos(raan) sin(raan) 0;
    -sin(raan) cos(raan) 0;
    0 0 1];

% Second rotation of the inclination in x axis
R1_i = [1 0 0
    0 cos(i) sin(i)
    0 -sin(i) cos(i)];

% Third rotation of the argument of the perigee in z axis
R3_omega = [cos(omega) sin(omega) 0;
    -sin(omega) cos(omega) 0;
    0 0 1];

%ACHTUNG: R(-omega) = R(omega).T

% Position rotation
rr = R3_raan' * R1_i' * R3_omega' * rr;

% Velocity rotation
dotrr = R3_raan' * R1_i' * R3_omega' * dotrr;

end
```

cart2efix.m

```
function [rrr, dotrrr] = cart2efix(rr, dotrr, t)
% Description: transform position and velocity in a spacefixed system
    into
% position and velocity in an Earth-fixed system

% Inputs: position and velocity in an inertial spacefixed system and
    time

% Outputs: the position and velocity in an Earth-fixed system
```

```
%% Parameters:

% Earth rotation period [rad/s]
omega_E = 2*pi/86164;

% Sideral angle
sideral = (3 + 29/60) * 15;

nsteps = length(t);

rrr = zeros(3, nsteps);
dotrrr = zeros(3, nsteps);

%% Time loop:

for j=1:nsteps

    theta_0 = omega_E .* t(j) + deg2rad(sideral);

    R3_theta_0 = [cos(theta_0) sin(theta_0) 0;
        -sin(theta_0) cos(theta_0) 0;
        0 0 1];

    rrr(:, j) = R3_theta_0 * rr(:, j);
    dotrrr(:, j) = R3_theta_0 * dotrr(:, j);
end

end
```

efix2topo.m

```
function [rrrr, azim, elev] = efix2topo(rrr, t)
% Description: transforms position and velocity into a topocentric
    system
% centered at the Wettzell
%
% Inputs: position and velocity in an Earth-fixed system
%
% Outputs: position and velocity
%

%% Parameters

% Postion vector in an Earth-fixed system given by:
r_w = [4075.53022; 931.78130; 4801.61819]*1e3;   % [m]

%% Translated vector
r_tran = rrr - r_w;

% Topocentric vector
```

```
% Latitude and longitude of Wettzell
lon_w = atan2(r_w(2), r_w(1));
lat_w = atan2(r_w(3), sqrt(r_w(1).^2 + r_w(2).^ 2));

% Right handed to left handed system converter matrix
Q = [-1 0 0;
     0 1 0;
     0 0 1];

R2ang = pi/2 - lat_w;
R3ang = lon_w;

R2_lat = [cos(R2ang) 0 -sin(R2ang);
          0 1 0;
          sin(R2ang) 0 cos(R2ang)];

R3_lon = [cos(R3ang) sin(R3ang) 0;
          -sin(R3ang) cos(R3ang) 0;
          0 0 1];

rrrr = Q * R2_lat * R3_lon * r_tran;


%% Azimuth and elevaton calculation

nsteps = length(t);

azim = zeros(1, nsteps);
elev = zeros(1, nsteps);


azim = atan2(rrrr(2, :), rrrr(1, :));
elev = atan2(rrrr(3, :), ...
       sqrt(rrrr(1, :).^2 + rrrr(2, :).^ 2));

end
```

main.m

```
%% Main

clear
clf
clc

a = xlsread('data.xlsx','A1:A5');   % Semi mayor axis
e = xlsread('data.xlsx','B1:B5');   % Eccentricity
i = xlsread('data.xlsx','C1:C5');   % Inclination
raan = xlsread('data.xlsx','D1:D5');   %RAAN
omega = xlsread('data.xlsx','E1:E5');   % Argument of the perigee
t_0 = xlsread('data.xlsx','F1:F5');   % Perigee passing time
```

```
omega_E = 2*pi/86164;   % Earth rotation period [rad/s]
period = 24*60*60;   % Orbit period [s]
u = 3.986004418e+14;   % Geocentric gravitational constant: u = GM
R = 6.371e+6;  % Earth radius [m]

%% Task 1:
dt = 60 * 1;
t = 0:dt:period;

[r1, v1, M1, E1] = kep2orb(a(1), e(1), t_0(1), t);
[r2, v2, M2, E2] = kep2orb(a(2), e(2), t_0(2), t);
[r3, v3, M3, E3] = kep2orb(a(3), e(3), t_0(3), t);
[r4, v4, M4, E4] = kep2orb(a(4), e(4), t_0(4), t);
[r5, v5, M5, E5] = kep2orb(a(5), e(5), t_0(5), t);

% Plot for the 5 satellites in the orbital plane for one orbital
% revolution


figure(1)
hold on
grid on

x1 = r1 .* cos(v1);
y1 = r1 .* sin(v1);

x2 = r2 .* cos(v2);
y2 = r2 .* sin(v2);

x3 = r3 .* cos(v3);
y3 = r3 .* sin(v3);

x4 = r4 .* cos(v4);
y4 = r4 .* sin(v4);

x5 = r5 .* cos(v5);
y5 = r5 .* sin(v5);

plot(x1, y1)
plot(x2, y2)
plot(x3, y3)
plot(x4, y4)
plot(x5, y5)

legend('GOCE', 'GPS', 'MOLNIYA', 'GEO', 'MICHIBIKI')
xlabel('x(m)')
ylabel('y(m)')
title('Orbit of the satellites in a 2D plane')

% Plot M, E, v, v-M for the GPS satellite
figure(2)
hold on
grid on

t_plot = t;
```

```
t_plot = t_plot ./ 3600;

plot(t_plot, rad2deg(M2))
plot(t_plot, rad2deg(E2))
plot(t_plot, rad2deg(v2))
plot(t_plot, rad2deg(v2-M2))

legend('M','E','v','v-M')
xlabel('time(s)')
ylabel('angle(degre)')
title('M, E, v, v-M of the GPS satellite')

% Plot M, E, v, v-M for the MOLNIYA satellite

figure(3)
hold on
grid on

t_plot = t;
t_plot = t_plot ./ 3600;

plot(t_plot, rad2deg(M3))
plot(t_plot, rad2deg(E3))
plot(t_plot, rad2deg(v3))
plot(t_plot, rad2deg(v3-M3))

legend('M','E','v','v-M')
xlabel('time(s)')
ylabel('angle(degre)')
title('M, E, v, v-M of the MOLNIYA satellite')

%% Task 2:
dt = 60 * 1;
t = 0:dt:period;
nsteps = length(t);

rr = zeros(3, nsteps, 5);
dotrr = zeros(3, nsteps, 5);



for j=1:5
    [rr(:, :, j), dotrr(:, :, j)] = kep2cart(a(j), e(j), i(j), raan(j),
        omega(j), t_0(j), t);
end


figure(4)


for j=1:5
    plot3(rr(1, :, j), rr(2, :, j), rr(3, :, j))
    hold on
end
```

```
legend('GOCE', 'GPS', 'MOLNIYA', 'GEO', 'MICHIBIKI')
xlabel('x(m)')
ylabel('y(m)')
zlabel('z(m)')
title('Orbits of 5 satellites in 3D plane')
grid on


Earth_coast(3)


figure(5)


for j=1:5
    plot(rr(1, :, j), rr(2, :, j))
    hold on
end

legend('GOCE', 'GPS', 'MOLNIYA', 'GEO', 'MICHIBIKI')
xlabel('x(m)')
ylabel('y(m)')
title('Orbits of 5 satellites in XY plane')
grid on



figure(6)

for j=1:5
    plot(rr(1, :, j), rr(3, :, j))
    hold on
end

legend('GOCE', 'GPS', 'MOLNIYA', 'GEO', 'MICHIBIKI')
xlabel('x(m)')
ylabel('z(m)')
title('Orbits of 5 satellites in XZ plane')
grid on


figure(7)

for j=1:5
    plot(rr(2, :, j), rr(3, :, j))
    hold on
end

legend('GOCE', 'GPS', 'MOLNIYA', 'GEO', 'MICHIBIKI')
xlabel('y(m)')
ylabel('z(m)')
title('Orbits of 5 satellites in YZ plane')
grid on
```

```
figure(8)

for j=1:5
    velocity = sqrt(dotrr(1, :, j).^2 + dotrr(2, :, j).^2 + dotrr(3, :,
        j).^2);
    plot(t, velocity)
    hold on
end

legend('GOCE', 'GPS', 'MOLNIYA', 'GEO', 'MICHIBIKI')
xlabel('time(s)')
ylabel('v(m/s)')
title('Time serires magnitude of velocity')
grid on

%% Task 3:
dt = 60 * 1;
t = 0:dt:period;
nsteps = length(t);

rrr = zeros(3, nsteps, 5);
dotrrr = zeros(3, nsteps, 5);


for j=1:5
    [rrr(:, :, j), dotrrr(:, :, j)] = cart2efix(rr(:, :, j), dotrr(:,
       :, j), t);
end

figure(9)

for j=[1 2 3 5]
    plot3(rrr(1, :, j), rrr(2, :, j), rrr(3, :, j))
    hold on
end

scatter3(rrr(1, 1, 4), rrr(2, 1, 4), rrr(3, 1, 4),'filled')
legend('GOCE', 'GPS', 'MOLNIYA', 'MICHIBIKI','GEO')
xlabel('x(m)')
ylabel('y(m)')
zlabel('z(m)')
title('Trayectory of 5 satellites inEarth-Fixed System')
grid on

Earth_coast(3)

latitude = zeros(nsteps, 5);
longitude = zeros(nsteps, 5);

for j=1:5
    longitude(:, j) = atan2(rrr(2, :, j), rrr(1, :, j));
    latitude(:, j) = atan2(rrr(3, :, j), ...
        sqrt(rrr(1, :, j).^2 + rrr(2, :, j).^ 2));
end
```

```
figure(10)

for j=1:5
    scatter(rad2deg(longitude(:, j)), rad2deg(latitude(:, j)))
    hold on
end

legend('GOCE', 'GPS', 'MOLNIYA','GEO','MICHIBIKI')
xlabel('longitude')
ylabel('latitude')
title('Satellite ground-tracks on the Earth surface')
grid on

Earth_coast(2)

%% Task 4:
dt = 60 * 1;
t = 0:dt:period;
nsteps = length(t);

rrrr = zeros(3, nsteps, 5);
azim = zeros(nsteps, 5);
elev = zeros(nsteps, 5);

figure(11)

for j=1:5
    [rrrr(:, :, j), azim(:, j), elev(:, j)] = efix2topo(rrr(:, :, j), t
       );
    skyplot(rad2deg(azim(:, j)), rad2deg(elev(:, j)), '+');
    hold on
end

title('Orbits in the topocentric system')
legend('GOCE', 'GPS', 'MOLNIYA','GEO','MICHIBIKI')
visibility = zeros(nsteps, 5);

for j=1:5
    visibility(:, j) = rad2deg(elev(:, j)) > 0.0;
end



figure(12)


subplot(5,1,1)
bar(t./3600, visibility(:, 1))
xticks(0:1:24)
yticks(0:1)
title('GOCE')

subplot(5,1,2)
```

```
bar(t./3600, visibility(:, 2))
xticks(0:1:24)
yticks(0:1)
title('GPS')

subplot(5,1,3)
bar(t./3600, visibility(:, 3))
xticks(0:1:24)
yticks(0:1)
title('MOLNIYA')

subplot(5,1,4)
bar(t./3600, visibility(:, 4))
xticks(0:1:24)
yticks(0:1)
title('GEO')

subplot(5,1,5)
bar(t./3600, visibility(:, 5))
xticks(0:1:24)
yticks(0:1)
title('MICHIBIKI')
```