Homework 5

Tycho Bogdanowitsch
AA 279A - Space Mechanics

February 26, 2025

Problem 1.

Part (a)

The value for $dUT1$ on February 1, 2023 was $-$**14.23ms** according to the IERS Bulletin.

Part (b)

```
function [a,e,i,RAAN,omega,nu,n,M] = ECI2OE(r_IJK,v_IJK)
    mu_earth = 3.986e5; % km^3/s^2
    r_norm = norm(r_IJK);
    r_i = r_IJK(1);
    r_j = r_IJK(2);
    r_k = r_IJK(3);
    v_norm = norm(v_IJK);
    h = cross(r_IJK,v_IJK);
    h_norm = norm(h);
    W = h/norm(h);
    W_i = W(1);
    W_j = W(2);
    W_k = W(3);
    i = atan2(sqrt(W_i^2 + W_j^2),W_k);
    RAAN = atan2(W_i,-W_j);
    p = h_norm^2/mu_earth;
    a = ((2/r_norm)-(v_norm^2/mu_earth))^(-1);
    n = sqrt(mu_earth/a^3);
    e = sqrt(1-p/a);
    E = atan2(dot(r_IJK,v_IJK)/(a^2*n),(1-r_norm/a));
    M = E - e*sin(E);
    nu = 2*atan2(sqrt(1+e)*tan(E/2),sqrt(1-e));
    u = atan2(r_k/sin(i),r_i*cos(RAAN)+r_j*sin(RAAN));
    omega = u - nu;
end

function E = NewtonRaphson(M,e,epsilon)
    E = M;

    while true
        fE = E - e*sin(E) - M;
        fprimeE = 1 - e*cos(E);
        delta = -fE/fprimeE;
        E_next = E+delta;
        if abs(delta)<epsilon
            break;
        end
        E = E_next;
    end
end

function [x,y] = OE2Perifocal(a,E,e)
    nu = 2*atan2(sqrt(1+e)*tan(E/2),sqrt(1-e));
    r = a*(1-e^2)/(1+e*cos(nu));
    x = r*cos(nu);
    y = r*sin(nu);
end

function r_ECI = OE2ECI(a,E,e,i,RAAN,omega)
    nu = 2*atan2(sqrt(1+e)*tan(E/2),sqrt(1-e));
    r = a*(1-e^2)/(1+e*cos(nu));
    r_PQW = [r * cos(nu); r * sin(nu); 0];

    R_PQW_IJK = [cos(RAAN)*cos(omega)-sin(RAAN)*cos(i)*sin(
       omega),...
        -cos(RAAN)*sin(omega)-sin(RAAN)*cos(i)*cos(omega),...
        sin(RAAN)*sin(i);
        sin(RAAN)*cos(omega)+cos(RAAN)*cos(i)*sin(omega),...
        -sin(RAAN)*sin(omega)+cos(RAAN)*cos(i)*cos(omega),...
        -cos(RAAN)*sin(i);
        sin(i)*sin(omega), sin(i)*cos(omega), cos(i)];

    r_ECI = R_PQW_IJK*r_PQW;
end

function r_ECEF = ECEF(a,E,e,i,RAAN,omega,UT1,t)
    M = UT1(1);
    D = UT1(2);
    Y = UT1(3);
    t_days = t/86400;
    D = D+t_days;
    if M<=2
        y = Y-1;
        m = M+12;
    else
        y = Y;
        m = M;
    end

    if (y<=1582 && m<=8 && D<= 4)
        B = -2 + (y+4716) / 4 - 1179;
    else
```

```
        B = y/400 - y/100 + y/4;
    end

    MJD = 365*y - 679004 + floor(B) + floor(30.6001 * (m+1))+D;

    d = MJD - 51544.5;
    GMST = 280.4606 + 360.9856476*d;

    GMST_rad_wrapped = wrapTo2Pi(GMST*pi/180);

    R_ECI_ECEF = [cos(GMST_rad_wrapped) sin(GMST_rad_wrapped)
       0;
        -sin(GMST_rad_wrapped) cos(GMST_rad_wrapped) 0;
        0 0 1];

    nu = 2*atan2(sqrt(1+e)*tan(E/2),sqrt(1-e));
    r = a*(1-e^2)/(1+e*cos(nu));
    r_PQW = [r * cos(nu); r * sin(nu); 0];

    R_PQR_IJK = [cos(RAAN)*cos(omega)-sin(RAAN)*cos(i)*sin(
       omega),...
        -cos(RAAN)*sin(omega)-sin(RAAN)*cos(i)*cos(omega),...
        sin(RAAN)*sin(i);
        sin(RAAN)*cos(omega)+cos(RAAN)*cos(i)*sin(omega),...
        -sin(RAAN)*sin(omega)+cos(RAAN)*cos(i)*cos(omega),...
        -cos(RAAN)*sin(i);
        sin(i)*sin(omega), sin(i)*cos(omega), cos(i)];

    r_ECI = R_PQR_IJK*r_PQW;

    r_ECEF = R_ECI_ECEF*r_ECI;
end
```

Using the above functions and the following Simulink program, the orbit of Molniya 3-31 can be found in the ECEF frame.

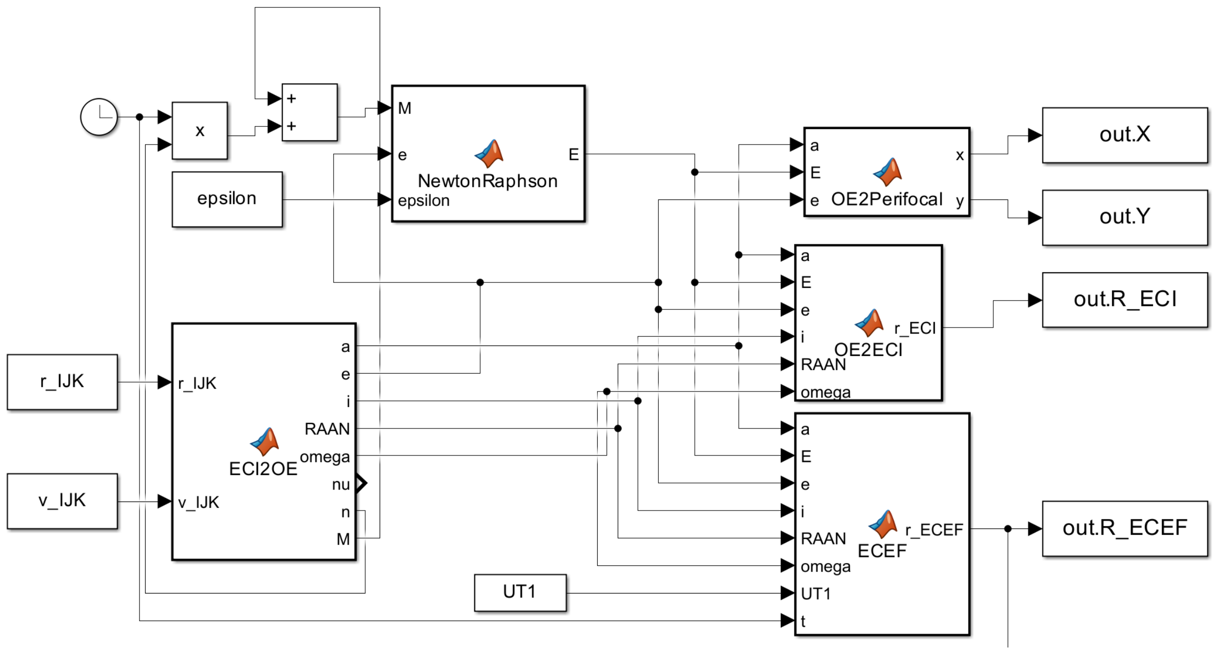

Figure 1: Molniya ECEF Simulink

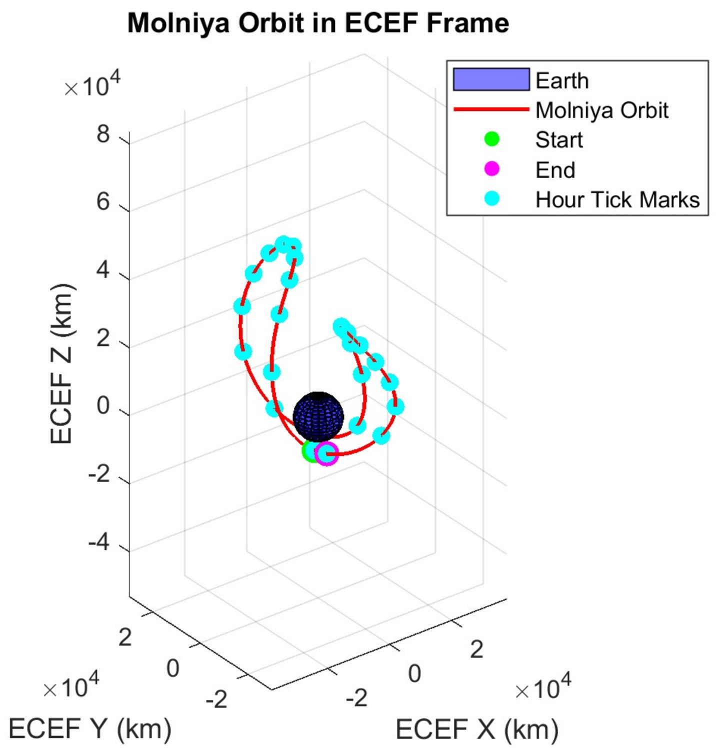

Figure 2: Molniya 3-31 orbit in ECEF Frame

Part (c)

```
function [lambda_prime,psi_prime] = ECEF2Geodetic(ECEF)
    r_x = ECEF(1);
    r_y = ECEF(2);
    r_z = ECEF(3);

    R_earth = 6378.1; % km
    e_earth = 0.0818;
    epsilon = 0.001; % deg

    r_xy = sqrt(r_x^2+r_y^2);

    lambda_prime = atan2d(r_y,r_x);

    psi_prime = asind(r_z/sqrt(r_x^2+r_y^2+r_z^2));

    while true
        N = R_earth/(sqrt(1-(e_earth^2)*(sind(psi_prime))^2));

        psi_prime_next = atan2d(r_z+N*(e_earth^2)*sind(
           psi_prime),r_xy);
        delta = psi_prime_next - psi_prime;

        if abs(delta)<epsilon
            psi_prime = psi_prime_next;
            break;
        end

        psi_prime = psi_prime_next;
    end
end
```

Now expanding the Simulink model using the above function to calculate the Geodetic coordinates, the ground track of the satellite can be plotted.

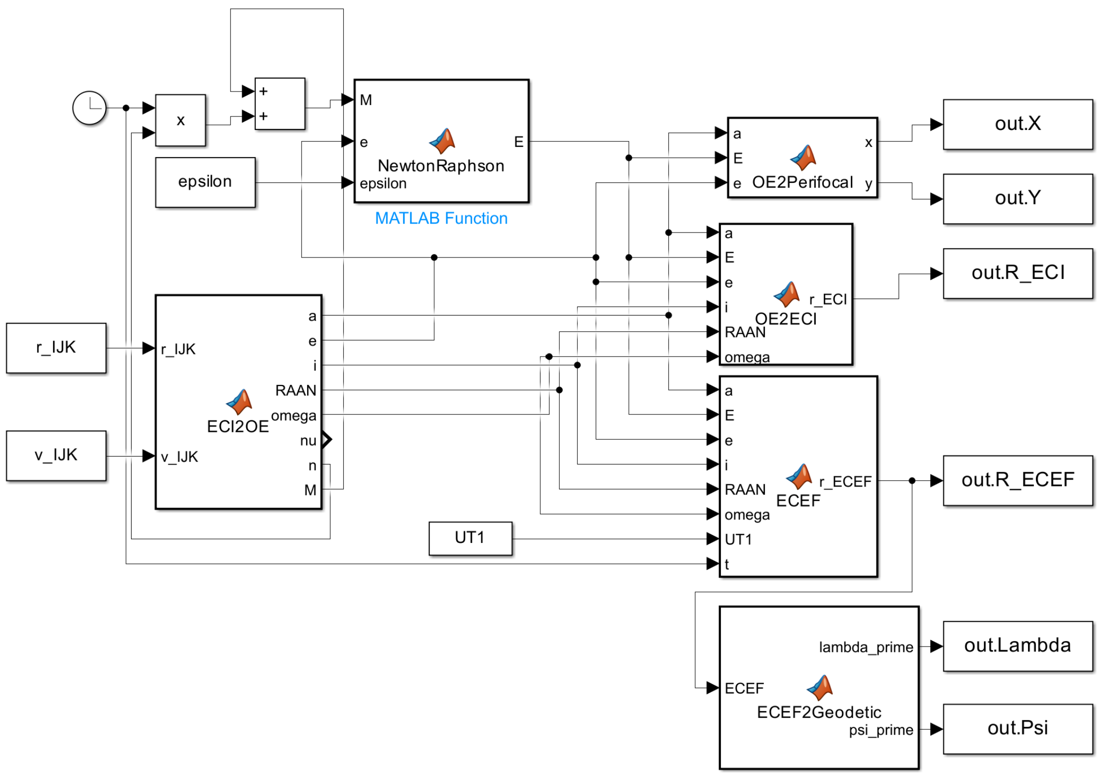

Figure 3: Extended Molniya Simulink

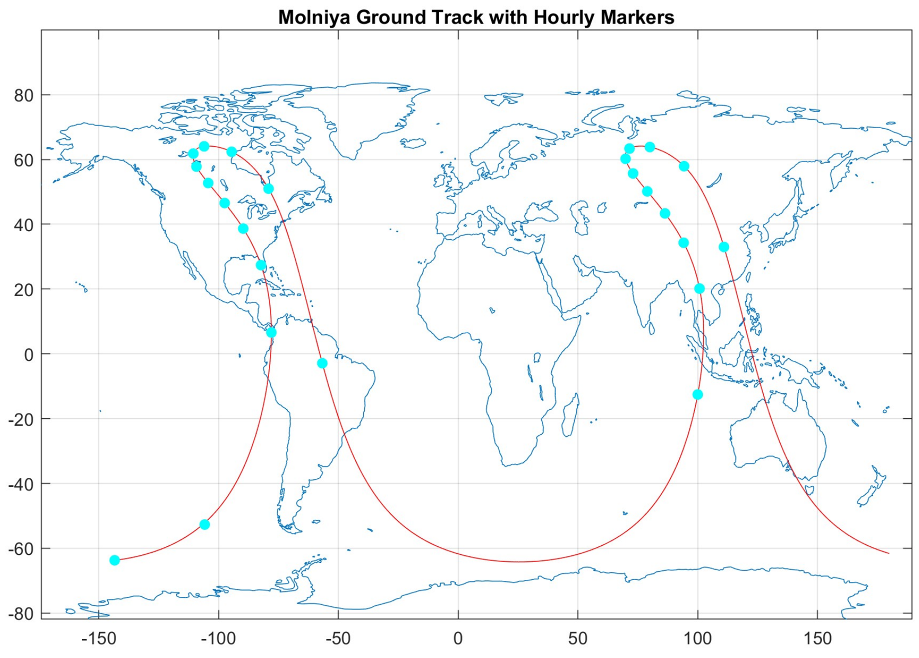

Figure 4: Molniya Ground Track

Part (d)

The Molniya 3-31 spends the most amount of time above North America and Asia. This represents a good use of the satellite time in orbit because the Soviet Union wanted to spy on the U.S. and then downlink their data back to Russia.

Problem 2.

Part (a)

```
function r_ENU = ENU(phi_obs,lambda_obs,r_ECI,MJD,t)
    R_earth = 6371; % km

    t_days = t/86400;
    MJD = MJD+t_days;

    d = MJD - 51544.5;
    GMST = 280.4606 + 360.9856476*d;

    GMST_rad_wrapped = wrapTo2Pi(GMST*pi/180);

    R_ECI_ECEF = [cos(GMST_rad_wrapped) sin(GMST_rad_wrapped)
       0;
        -sin(GMST_rad_wrapped) cos(GMST_rad_wrapped) 0;
        0 0 1];

    r_ECEF = R_ECI_ECEF*r_ECI;

    x_s = r_ECEF(1);
    y_s = r_ECEF(2);
    z_s = r_ECEF(3);

    r_obs_xyz = R_earth*[cosd(phi_obs)*cosd(lambda_obs);
        cosd(phi_obs)*sind(lambda_obs);
        sind(phi_obs)];

    x_obs = R_earth*cosd(phi_obs)*cosd(lambda_obs);
    y_obs = R_earth*cosd(phi_obs)*sind(lambda_obs);
    z_obs = R_earth*(sind(phi_obs));

    dx = x_s - x_obs;
    dy = y_s - y_obs;
    dz = z_s - z_obs;

    R_XYZ_ENU = [-sind(lambda_obs), cosd(lambda_obs), 0;
        -sind(phi_obs)*cosd(lambda_obs),-sind(phi_obs)*sind(
           lambda_obs),cosd(phi_obs);
        cosd(phi_obs)*cosd(lambda_obs),cosd(phi_obs)*sind(
           lambda_obs),sind(phi_obs)];

    r_ENU = R_XYZ_ENU*[dx;dy;dz];
```

```
end
```

Using the above functions in the Simulink program shown below the ENU coordinates of a satellite.

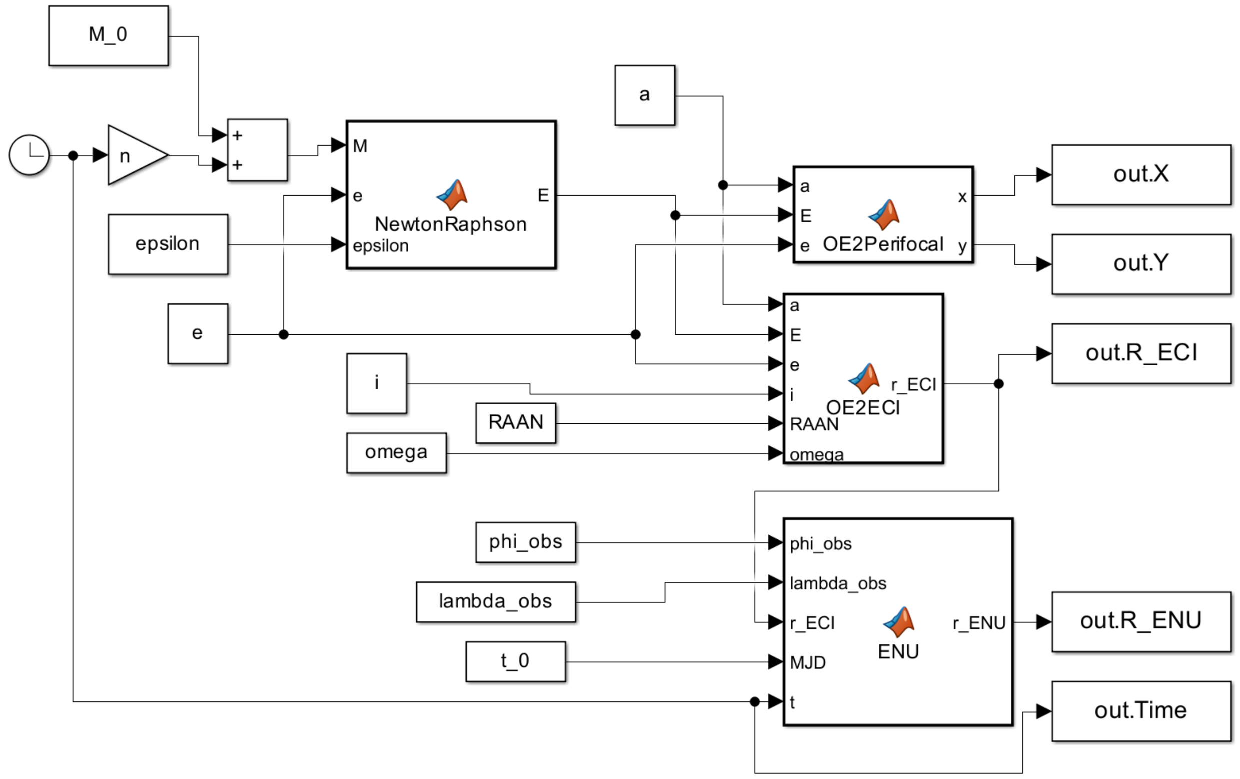

Figure 5: Topocentric Coordinates Simulink

The final ENU coordinates of the satellite at the end of the simulation are:

$$\begin{bmatrix} -1629.4 \\ 4240.4 \\ -843.7 \end{bmatrix} km$$

Part (b)

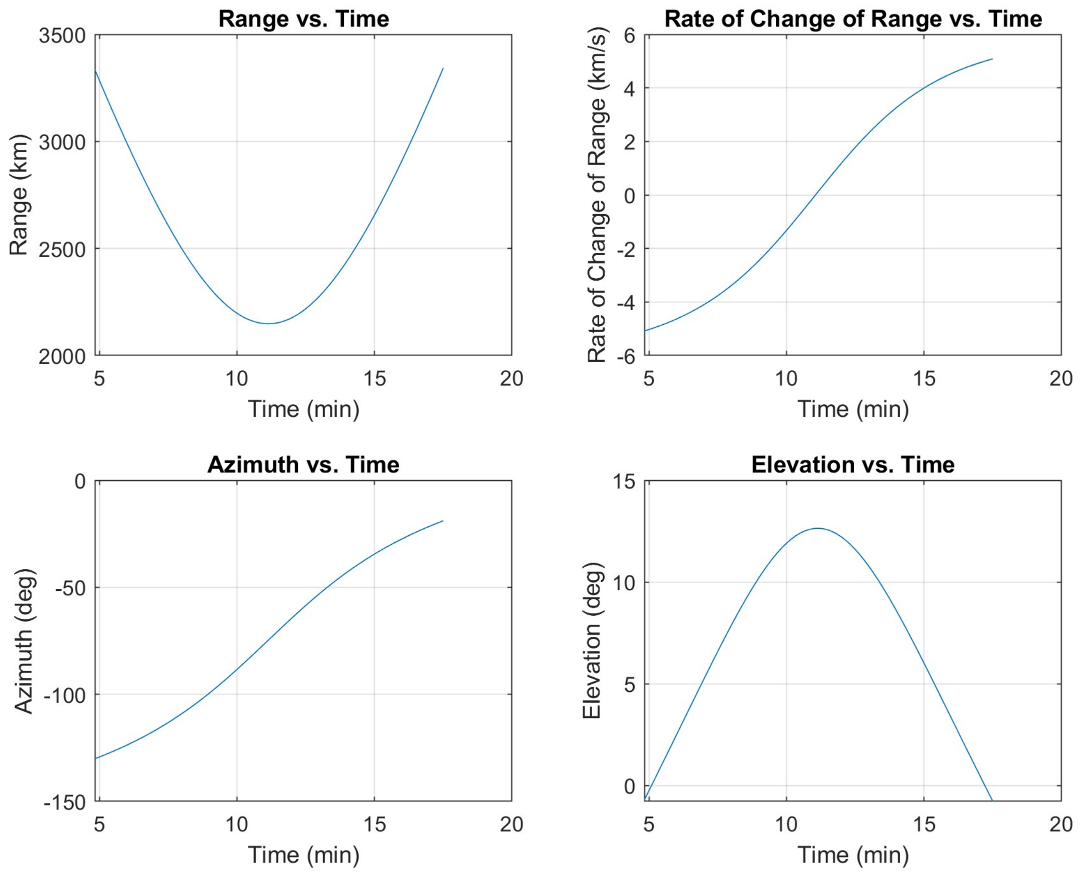

Figure 6: Simulation Plots

From the ENU coordinates, the range, rate of change of range, azimuth, and elevation can all be computed and plotted versus time. Importantly, these plots are only for when the satellite is visible, which is when the elevation is greater than 0 degrees.
The following equations were used to calculate these values:

$$\text{dist} = \|\mathbf{r}_{ENU}\| \tag{1}$$

$$A = atan(r_E/r_N) \tag{2}$$

$$E = atan\left(r_U/\sqrt{r_E^2 + r_N^2}\right) \tag{3}$$

$$\frac{d(\text{range})}{dt} = \frac{\text{diff(dist)}}{\text{diff(time)}} \tag{4}$$

Part (c)

From the plots, the time after t_0 when the satellite is closest to the observer is:

$$t_{\text{closest}} = 11.167 \text{ minutes} \tag{5}$$

Part (d)

From the plots, the satellite is visible to the observer for:

$$t_{\text{visible}} = 12.667 \text{ minutes} \tag{6}$$

Part (e)

From the plots, the maximum elevation of the satellite is:

$$E_{\text{max}} = 12.639^\circ \tag{7}$$

Part (f)

From the plots, the maximum distance to the satellite is:

$$d_{\text{max}} = 3342.941 \text{ km} \tag{8}$$

And the minimum distance to the satellite is:

$$d_{\text{min}} = 2148.196 \text{ km} \tag{9}$$

Part (g)

Using the following equations:

$$\frac{dA}{dt} = \frac{\text{diff}(A_{\text{interval}})}{\text{diff}(\text{time}_{\text{interval}})} \tag{10}$$

$$\frac{dE}{dt} = \frac{\text{diff}(E_{\text{interval}})}{\text{diff}(\text{time}_{\text{interval}})} \tag{11}$$

The maximum rate of change of azimuth is:

$$\dot{A}_{\text{max}} = 0.20 \text{ deg/sec} \tag{12}$$

The maximum rate of change of elevation is:

$$\dot{E}_{\text{max}} = 0.05 \text{ deg/sec} \tag{13}$$

Problem 3.

Part (a)

Since we want to maintain a sun-synchronous orbit, the following equation - which is from the Gauss-Variational Equations - must hold.

$$\left(\frac{d\Omega}{dt}\right)_{\text{avg}} = \frac{360°}{365.25 \text{ days}} = -\frac{3}{2}nJ_2\left(\frac{R_E}{a(1-e^2)}\right)^2 \cos i \tag{14}$$

$$i = \cos^{-1}\left(\left(\frac{d\Omega}{dt}\right)_{\text{avg}}\left(\frac{-2}{3}\right)\left(\frac{1}{nJ_2}\right)\left(\frac{a(1-e^2)}{R_E}\right)^2\right) \tag{15}$$

Thus, the inclination of the orbit should be:

$$i = 96.839° \tag{16}$$

Part (b)

Using the GVE for a near-circular orbit:

$$\Delta V_T = \frac{n\Delta a}{2} \tag{17}$$

The required ΔV is:

$$\Delta V_T = 0.115 \text{ m/s in the positive T direction} \tag{18}$$

A maneuver at periapsis has the largest effect on energy change. This can be seen in both the vis-viva equation and the general GVE for a:

$$v^2 = \mu\left(\frac{2}{r} - \frac{1}{a}\right) \tag{19}$$

$$\frac{da}{dt} = \frac{2e\sin f f_r}{n\sqrt{1-e^2}} + \frac{2aJ_2\sqrt{1-e^2}f_r}{nr} \tag{20}$$

Thus, the maneuver should occur at periapsis to be the most efficient. However, for a near-circular orbit, this efficiency gain may be negligible, in which case the maneuver can occur anywhere.
Periapsis for this orbit is at:

$$u = \omega = 270° \tag{21}$$

This maneuver will change other orbital elements, specifically the eccentricity of the orbit as shown by the GVEs for e_x and e_y:

$$\Delta e_x = \frac{1}{na}(\sin u_m \Delta V_r + 2\cos u_m \Delta V_T) \tag{22}$$

$$\Delta e_y = \frac{1}{na}(-\cos u_m \Delta V_r + 2\sin u_m \Delta V_T) \tag{23}$$

Part (c)

Considering the GVE for Δe_x and Δe_y for two maneuvers separated by 180°:

$$\Delta e_{x,\text{Total}} = \Delta e_{x1} + \Delta e_{x2} = \frac{1}{na}(2\cos u_{m1}\Delta V_{T1}) + \frac{1}{na}(2\cos u_{m2}\Delta V_{T2}) = 0 \tag{24}$$

$$\Delta e_{y,\text{Total}} = \Delta e_{y1} + \Delta e_{y2} = \frac{1}{na}(2\sin u_{m1}\Delta V_{T1}) + \frac{1}{na}(2\sin u_{m2}\Delta V_{T2}) = 0 \tag{25}$$

$$\Delta V_{T1} + \Delta V_{T2} = \frac{n}{2}\Delta a \tag{26}$$

$$-\cos u_{m1}\Delta V_{T1} = \cos u_{m2}\Delta V_{T2} \tag{27}$$

$$-\sin u_{m1}\Delta V_{T1} = \sin u_{m2}\Delta V_{T2} \tag{28}$$

$$\tan u_{m1} = \tan u_{m2} \tag{29}$$

This relationship holds for all

$$u_{m2} = u_{m1} + 180^\circ \tag{30}$$

This means that it is possible for any u_{m1} and u_{m2} that are 180° apart to result in no net eccentricity change with a semi-major axis change.
Thus these maneuvers can occur anywhere if disregarding efficiency. For the most efficient set of maneuvers, they should occur at the periapsis and apoapsis for the same reason as mentioned above.
For the magnitude of these maneuvers, the total magnitude is the same as before

$$\Delta V_T = \frac{n\Delta a}{2} \tag{31}$$

This means that each individual magnitude should be

$$\Delta V_T = \frac{n\Delta a}{4} \tag{32}$$

The required ΔV is:

$$\Delta V_T = 0.057 \text{ m/s in the positive T direction} \tag{33}$$

Part (d)

$$\Delta i = \frac{\cos u_m \Delta V_N}{na} \tag{34}$$

For maximum efficiency, place at:

$$u_m = 0^\circ, 180^\circ \tag{35}$$

$$a\Delta i = \frac{\cos u_m \Delta V_N}{n} \quad \Rightarrow \quad \Delta V_N = n(a\Delta i) \tag{36}$$

The required ΔV is:

$$\Delta V_N = 0.229 \text{ m/s in the positive N direction} \tag{37}$$

No, this maneuver will not affect any other orbital elements. Even though $\Delta\Omega$ and Δu are dependent on ΔV_N, there is also a $\sin u_m$ term, so this effect is canceled out at the line of nodes.

Appendix 1

```
%% Space Mechanics Homework 4
% Tycho Bogdanowitsch
clc; clear;
% Constants
R_earth = 6371; % km
DU_earth = R_earth; % km
AU = 1.496e8; % km
mu_earth = 3.986e5; % km^3/s^2
mu_mars = 4.283e4; % km^3/s^2
mu_sun = 1.327e11; % km^3/s^2
R_sun = 6.963e5; % km

%% Problem 1
close all;
fprintf(['\nProblem 1\n']);

% Part b
r_IJK = [3727.9;-826.3;-7693.2]; % km
v_IJK = [4.0256;7.8002;-0.8610]; % km/s

UTC = [2 1 2023];
dUTC = -0.01423/86400; % seconds --> days
UT1 = UTC + [0 dUTC 0];

epsilon = 1e-10;

stop_time = 1*24*3600; % days --> seconds

function [a,e,i,RAAN,omega,nu,n,M] = ECI2OE(r_IJK,v_IJK)
    mu_earth = 3.986e5; % km^3/s^2
    r_norm = norm(r_IJK);
    r_i = r_IJK(1);
    r_j = r_IJK(2);
    r_k = r_IJK(3);
    v_norm = norm(v_IJK);
    h = cross(r_IJK,v_IJK);
    h_norm = norm(h);
    W = h/norm(h);
    W_i = W(1);
    W_j = W(2);
    W_k = W(3);
    i = atan2(sqrt(W_i^2 + W_j^2),W_k);
    RAAN = atan2(W_i,-W_j);
    p = h_norm^2/mu_earth;
    a = ((2/r_norm)-(v_norm^2/mu_earth))^(-1);
    n = sqrt(mu_earth/a^3);
    e = sqrt(1-p/a);
    E = atan2(dot(r_IJK,v_IJK)/(a^2*n),(1-r_norm/a));
    M = E - e*sin(E);
    nu = 2*atan2(sqrt(1+e)*tan(E/2),sqrt(1-e));
    u = atan2(r_k/sin(i),r_i*cos(RAAN)+r_j*sin(RAAN));
    omega = u - nu;
end

function E = NewtonRaphson(M,e,epsilon)
    E = M;

    while true
        fE = E - e*sin(E) - M;
        fprimeE = 1 - e*cos(E);
        delta = -fE/fprimeE;
        E_next = E+delta;
        if abs(delta)<epsilon
            break;
        end

        E = E_next;
    end
end

function [x,y] = OE2Perifocal(a,E,e)
    nu = 2*atan2(sqrt(1+e)*tan(E/2),sqrt(1-e));
    r = a*(1-e^2)/(1+e*cos(nu));
    x = r*cos(nu);
    y = r*sin(nu);
end

%E_test = NewtonRaphson(M,e,epsilon);

x = out.X;
y = out.Y;

x_start = x(1);
y_start = y(1);

x_end = x(end);
y_end = y(end);
```

```
figure(1);
hold on;
plot(x, y,'DisplayName','Molniya Orbit');
axis equal;
xlabel('PQW X (km)');
ylabel('PQW Y (km)');
title('Molniya Orbit in PQW Frame');
scatter(x_start,y_start,'g','filled','DisplayName', 'Start');
scatter(x_end,y_end,'m','filled','DisplayName', 'End');
grid on;
legend show;
hold off;

function r_ECI = OE2ECI(a,E,e,i,RAAN,omega)
    nu = 2*atan2(sqrt(1+e)*tan(E/2),sqrt(1-e));
    r = a*(1-e^2)/(1+e*cos(nu));
    r_PQW = [r * cos(nu); r * sin(nu); 0];

    R_PQW_IJK = [cos(RAAN)*cos(omega)-sin(RAAN)*cos(i)*sin(
       omega),...
         -cos(RAAN)*sin(omega)-sin(RAAN)*cos(i)*cos(omega),...
         sin(RAAN)*sin(i);
         sin(RAAN)*cos(omega)+cos(RAAN)*cos(i)*sin(omega),...
         -sin(RAAN)*sin(omega)+cos(RAAN)*cos(i)*cos(omega),...
         -cos(RAAN)*sin(i);
         sin(i)*sin(omega), sin(i)*cos(omega), cos(i)];

    r_ECI = R_PQW_IJK*r_PQW;
end

%r_ECI_test = OE2ECI(a,E_test,e,i,RAAN,omega);

r_ECI = out.R_ECI;

[xE,yE,zE] = ellipsoid(0,0,0,R_earth,R_earth,R_earth,20);

figure(2)
hold on;

surface(xE, yE, zE, 'FaceColor', 'blue', 'EdgeColor', 'black',
   'FaceAlpha', 0.5,'DisplayName', 'Earth');

plot3(r_ECI(1,:), r_ECI(2,:), r_ECI(3,:),'r', 'LineWidth', 1.5,
   'DisplayName', 'Molniya Orbit');
```

```
scatter3(r_ECI(1,1), r_ECI(2,1), r_ECI(3,1), 100, 'g', 'filled'
   ,'DisplayName', 'Start');
scatter3(r_ECI(1,end), r_ECI(2,end), r_ECI(3,end), 100, 'm', '
   filled','DisplayName', 'End');

axis equal;
grid on;
legend show;
xlabel('ECI X (km)');
ylabel('ECI Y (km)');
zlabel('ECI Z (km)');
title('Molniya Orbit in ECI Frame');
view(3);
hold off;

function r_ECEF = ECEF(a,E,e,i,RAAN,omega,UT1,t)
    M = UT1(1);
    D = UT1(2);
    Y = UT1(3);
    t_days = t/86400;
    D = D+t_days;
    if M<=2
        y = Y-1;
        m = M+12;
    else
        y = Y;
        m = M;
    end

    if (y<=1582 && m<=8 && D<= 4)
        B = -2 + (y+4716) / 4 - 1179;
    else
        B = y/400 - y/100 + y/4;
    end

    MJD = 365*y - 679004 + floor(B) + floor(30.6001 * (m+1))+D;

    d = MJD - 51544.5;
    GMST = 280.4606 + 360.9856476*d;

    GMST_rad_wrapped = wrapTo2Pi(GMST*pi/180);

    R_ECI_ECEF = [cos(GMST_rad_wrapped) sin(GMST_rad_wrapped)
       0;
```

```
        -sin(GMST_rad_wrapped) cos(GMST_rad_wrapped) 0;
        0 0 1];

    nu = 2*atan2(sqrt(1+e)*tan(E/2),sqrt(1-e));
    r = a*(1-e^2)/(1+e*cos(nu));
    r_PQW = [r * cos(nu); r * sin(nu); 0];

    R_PQR_IJK = [cos(RAAN)*cos(omega)-sin(RAAN)*cos(i)*sin(
       omega),...
        -cos(RAAN)*sin(omega)-sin(RAAN)*cos(i)*cos(omega),...
        sin(RAAN)*sin(i);
        sin(RAAN)*cos(omega)+cos(RAAN)*cos(i)*sin(omega),...
        -sin(RAAN)*sin(omega)+cos(RAAN)*cos(i)*cos(omega),...
        -cos(RAAN)*sin(i);
        sin(i)*sin(omega), sin(i)*cos(omega), cos(i)];

    r_ECI = R_PQR_IJK*r_PQW;

    r_ECEF = R_ECI_ECEF*r_ECI;
end

r_ECEF = out.R_ECEF;

n = length(r_ECEF);
num_points = 24;

points = round(linspace(1,n,num_points));
hour_points = r_ECEF(:,points);

figure(3)
hold on;

surface(xE, yE, zE, 'FaceColor', 'blue', 'EdgeColor', 'black',
   'FaceAlpha', 0.5,'DisplayName', 'Earth');

plot3(r_ECEF(1,:), r_ECEF(2,:), r_ECEF(3,:),'r', 'LineWidth',
   1.5,'DisplayName', 'Molniya Orbit');

scatter3(r_ECEF(1,1), r_ECEF(2,1), r_ECEF(3,1), 100, 'g', '
   filled','DisplayName', 'Start');
scatter3(r_ECEF(1,end), r_ECEF(2,end), r_ECEF(3,end), 100, 'm',
    'filled','DisplayName', 'End');
scatter3(hour_points(1,:), hour_points(2,:), hour_points(3,:),
   50, 'c', 'filled','DisplayName', 'Hour Tick Marks');
```

```
axis equal;
grid on;
legend show;
xlabel('ECEF X (km)');
ylabel('ECEF Y (km)');
zlabel('ECEF Z (km)');
title('Molniya Orbit in ECEF Frame');
view(3);
hold off;

% Part c
function [lambda_prime,psi_prime] = ECEF2Geodetic(ECEF)
    r_x = ECEF(1);
    r_y = ECEF(2);
    r_z = ECEF(3);

    R_earth = 6378.1; % km
    e_earth = 0.0818;
    epsilon = 0.001; % deg

    r_xy = sqrt(r_x^2+r_y^2);

    lambda_prime = atan2d(r_y,r_x);

    psi_prime = asind(r_z/sqrt(r_x^2+r_y^2+r_z^2));

    while true
        N = R_earth/(sqrt(1-(e_earth^2)*(sind(psi_prime))^2));

        psi_prime_next = atan2d(r_z+N*(e_earth^2)*sind(
           psi_prime),r_xy);
        delta = psi_prime_next - psi_prime;

        if abs(delta)<epsilon
            psi_prime = psi_prime_next;
            break;
        end

        psi_prime = psi_prime_next;
    end
end

long = out.Lambda;
lat = out.Psi;
```

```
n = length(long);
num_points = 24;

points = round(linspace(1,n,num_points));
long_points = long(points);
lat_points = lat(points);

figure(4)
load coastlines;
plot(coastlon, coastlat);
hold on;

discontinuities = find(abs(diff(long)) > 180);
start = 1;
for i = 1:length(discontinuities)
    stop = discontinuities(i);
    plot(long(start:stop), lat(start:stop), 'r');
    start = stop + 1;
end
%plot(long,lat,'r');
scatter(long_points,lat_points,'c','filled','DisplayName', '
   Hour Tick Marks');
title('Molniya Ground Track with Hourly Markers');
grid on;
hold off;

%% Problem 2
close all;
fprintf(['\nProblem 2\n']);

% Part a
t_0 = 59987.6458; % MJD
a = 7155.81; % km
RAAN = deg2rad(230.875); % rad
e = 0.0001971; %
omega = deg2rad(93.069); % rad
i = deg2rad(86.403); % rad
M_0 = deg2rad(267.074); % rad

mu_earth = 3.986e5; % km^3/s^2

T = 2*pi*sqrt(a^3/mu_earth); % seconds
n = 2*pi/T; % rad/s

phi_obs = 37.426622; % deg
```

```
lambda_obs = -122.173355; % deg

function r_ENU = ENU(phi_obs,lambda_obs,r_ECI,MJD,t)
    R_earth = 6371; % km

    t_days = t/86400;
    MJD = MJD+t_days;

    d = MJD - 51544.5;
    GMST = 280.4606 + 360.9856476*d;

    GMST_rad_wrapped = wrapTo2Pi(GMST*pi/180);

    R_ECI_ECEF = [cos(GMST_rad_wrapped) sin(GMST_rad_wrapped)
       0;
        -sin(GMST_rad_wrapped) cos(GMST_rad_wrapped) 0;
        0 0 1];

    r_ECEF = R_ECI_ECEF*r_ECI;

    x_s = r_ECEF(1);
    y_s = r_ECEF(2);
    z_s = r_ECEF(3);

    r_obs_xyz = R_earth*[cosd(phi_obs)*cosd(lambda_obs);
        cosd(phi_obs)*sind(lambda_obs);
        sind(phi_obs)];

    x_obs = R_earth*cosd(phi_obs)*cosd(lambda_obs);
    y_obs = R_earth*cosd(phi_obs)*sind(lambda_obs);
    z_obs = R_earth*(sind(phi_obs));

    dx = x_s - x_obs;
    dy = y_s - y_obs;
    dz = z_s - z_obs;

    R_XYZ_ENU = [-sind(lambda_obs), cosd(lambda_obs), 0;
        -sind(phi_obs)*cosd(lambda_obs),-sind(phi_obs)*sind(
           lambda_obs),cosd(phi_obs);
        cosd(phi_obs)*cosd(lambda_obs),cosd(phi_obs)*sind(
           lambda_obs),sind(phi_obs)];

    r_ENU = R_XYZ_ENU*[dx;dy;dz];
end
```

```
r_ENU = squeeze(out.R_ENU);

fprintf('Final ENU Coordinates (km): \n');
disp(r_ENU(:,end))

% Part b
time = out.Time;
time_min = time/60;

r_E = r_ENU(1,:);
r_N = r_ENU(2,:);
r_U = r_ENU(3,:);
dist = vecnorm(r_ENU);
A = atan2d(r_E,r_N);
E = atan2d(r_U,sqrt(r_E.^2+r_N.^2));
drange_dt = diff(dist)'./diff(time);


E_zero_indices = find(abs(E)<1);
start = E_zero_indices(1);
stop = E_zero_indices(end);

time_interval = time_min(start:stop);
time_short = time_interval(1:end-1);
dist_interval = dist(start:stop);
drange_dt_interval = drange_dt(start:stop);
A_interval = A(start:stop);
E_interval = E(start:stop);

figure(1)
plot(time_min,E)
xlabel('Time (min)');
ylabel('Elevation (deg)');
title('Elevation vs. Time')
grid on;

figure(2);

subplot(2,2,1);
plot(time_interval, dist_interval);
xlabel('Time (min)');
ylabel('Range (km)');
title('Range vs. Time');
grid on;
```

```
subplot(2,2,2);
plot(time_interval, drange_dt_interval);
xlabel('Time (min)');
ylabel('Rate of Change of Range (km/s)');
title('Rate of Change of Range vs. Time');
grid on;

subplot(2,2,3);
plot(time_interval, A_interval);
xlabel('Time (min)');
ylabel('Azimuth (deg)');
title('Azimuth vs. Time');
grid on;

subplot(2,2,4);
plot(time_interval, E_interval);
xlabel('Time (min)');
ylabel('Elevation (deg)');
title('Elevation vs. Time');
grid on;

% Part c
[min_dist, min_index] = min(dist_interval);
t_min_dist = time_interval(min_index);
fprintf('The time after t_0 when the satellite is closest to
   the observer is: %.3f minutes\n', t_min_dist);

% Part d
t_interval_length = time_interval(end) - time_interval(1);
fprintf('The satellite is visible to the observer for: %.3f
   minutes\n', t_interval_length);

% Part e
[max_elevation, max_elevation_index] = max(E_interval);
fprintf('The maximum elevation of the satellite is: %.3f
   degrees\n', max_elevation);

% Part f
max_dist = max(dist_interval);
min_dist = min(dist_interval);
fprintf('The maximum distance to the satellite is: %.3f km\n',
   max_dist);
fprintf('The minimum distance to the satellite is: %.3f km\n',
   min_dist);
```

```
% Part g
dA_dt = (diff(A_interval)' ./ diff(time_interval)) ./ 60;
dE_dt = (diff(E_interval)' ./ diff(time_interval))./ 60;
[max_dA_dt, max_dA_index] = max(abs(dA_dt));
[max_dE_dt, max_dE_index] = max(abs(dE_dt));

fprintf('Maximum rate of change of Azimuth: %.2f deg/sec \n',
   max_dA_dt);
fprintf('Maximum rate of change of Elevation: %.2f deg/sec \n',
    max_dE_dt);

%% Problem 3
clear; clc; close all;
mu_earth = 3.986e5; % km^3/s^2
J_2 = 1082e-6;
R_earth = 6371;

a = 6720; % km
e = 0.00021;
omega = deg2rad(270); % rad
RAAN = deg2rad(90); % rad

T = 2*pi*sqrt(a^3/mu_earth); % seconds
n = 2*pi/T; % rad/s

% Part a
year_sec = 365.25*24*3600; % days --> seconds
dRAANdt = deg2rad(360)/year_sec;

i = acosd(dRAANdt*(-2/3)*(1/(n*J_2))*((a*(1-e^2)/R_earth))^2);
fprintf('The inclination of the orbit should be: %.3f degrees\n
   ', i);

% Part b
delta_a = 200; % m

delta_v_T = (n/2)*delta_a;
fprintf('The required deltaV has a magnitude of: %.3f m/s in
   the positive T direction\n', delta_v_T);

% Part c
delta_v_T = (n/4)*delta_a;
fprintf('The required deltaV has a magnitude of: %.3f m/s in
   the positive T direction\n', delta_v_T);
```

```
% Part d
aDeltaI = 200; % m

delta_v_N = n*aDeltaI;
fprintf('The required deltaV has a magnitude of: %.3f m/s in
   the positive N direction\n', delta_v_N);
```