Task 1:

```
function nu = rootFinder(f,t0,nu0,t,p,e,mu,N)
% ------------------------------------------------------------------%
% ------------------------ ROOTFINDER.M ----------------------------%
% ------------------------------------------------------------------%
% Find the root of a function of the form
% G(nu) = F(nu) - (t-t0) = 0
% where F(nu) is given as
%
%                    /nu
%                   /
%                   |
%           F(nu) = | f(q)dq
%                   |
%                   /
%                  /nu0
%
% The inputs and outputs of this function are as follows:
% Inputs:
% f: integrand of function
% nu0: true anomaly at current time
% t0: current time
% t: terminal time
% p: semi-latus rectum
% e: eccentricity
% mu: gravitational parameter
% N: number of Legendre-Gauss points
% Output:
% nu: true anomaly at time t
% ------------------------------------------------------------------%

deltat = (t-t0);
deltaNu = deg2rad(5);
guessNu = nu0+deltaNu;

for i=1:20
    [theta,w] = GaussPointsWeights(nu0,guessNu,N);
    theta=theta';
    F = f(theta,p,e,mu);
    F = F';
    FofNu = w.'*F;

    GofNu = FofNu - deltat;
    Gprime = f(guessNu,p,e,mu);

    guessNu = guessNu -(GofNu/Gprime);
end

nu = guessNu;

end
```

```
function nuValues = computeNu(tValues,rv0,vv0,mu)
% ------------------------------------------------------------%
% ------------------------ COMPUTENU.M ------------------------%
% ------------------------------------------------------------%
% Compute the values of the true anomaly at the times given in
% the column matrix TVALUES using the initial position given in
% RV0 and the initial inertial velocity given in VV0.
% Inputs:
%   tValues: column matrix of values of time
%       rv0: initial position expressed in
%            planet-centered inertial Cartesian coordinates
%       vv0: initial inertial velocity expressed in
%            planet-centered inertial Cartesian coordinates
%       mu: gravitational parameter of planet
% Output:
%    nuValues: column matrix of values of true anomaly that is
%              of the same length as the column matrix TVALUES
% ------------------------------------------------------------%

oe = rv2oe_Hackbardt_Chris(rv0,vv0,mu);
N = 20;
nuValues = zeros(0,length(tValues));
e = oe(2);
p = oe(1)*(1-e^2);
nuValues(1)=oe(6);
nu0=nuValues(1);

for i=1:length(tValues)-1
    nu0 = nuValues(i);
    t0 = tValues(i);
    t = tValues(i+1);
    nuValues(i+1) = rootFinder(@timeChangeIntegrand,t0,nu0,t,p,e,mu,N);
end

nuValues = nuValues';

end
```

Task 2:

```
function [rvValues,vvValues] = computeRandV(nuValues,a,e,Omega,inc,omega,mu)
% ------------------------------------------------------------%
% ---------------------- COMPUTERANDV.M ----------------------%
% ------------------------------------------------------------%
% Compute the values of the R and V for the calculated values of
% true anomaly.
% Inputs:
% nuValues: column matrix of values of true anomaly
% a: semi-major axis
% e: eccentricity
% Omega: longitude of the ascending node
% inc: orbital inclination
% omega: argument of the periapsis
% mu: gravitational parameter of planet
% Outputs:
% rvValues: column matrix of values of position,
% where each row in the matrix RVVALUES is
% expressed in planet-centered inertial Cartesian
% coordinates
% vvValues: column matrix of values of inertial velocity,
% where each row in the matrix VVVALUES is
% expressed in planet-centered inertial Cartesian
% coordinates
% ------------------------------------------------------------
% NOTE: the number of ROWS in RVVALUES and VVVALUES should be
% the same as the number of ROWS in NUVALUES
% ------------------------------------------------------------
rvValues = zeros(length(nuValues),3);
vvValues = zeros(length(nuValues),3);

for i=1:length(nuValues)
    oe = [a,e,Omega,inc,omega,nuValues(i)];
    [rvec,vvec]  = oe2rv_Hackbardt_Chris(oe,mu);
    for j=1:3
    rvValues(i,j)=rvec(j);
    vvValues(i,j)=vvec(j);
    end

end

end
```

Task 3:

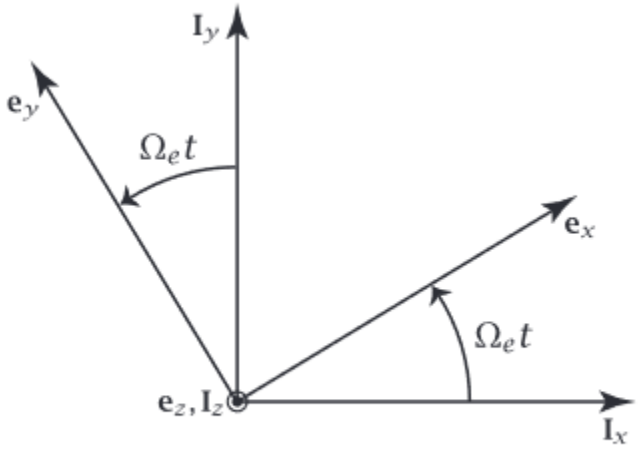


Figure 1: Schematic of the ECI basis $\mathcal{I} = \{\mathbf{I}_x, \mathbf{I}_y, \mathbf{I}_z\}$ and the ECEF basis $\mathcal{E} = \{\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z\}$.

Using Figure 1 to find the transformation from ECI to ECEF coordinates:

We know that the angle, $\Omega_e t$, is the rotation rate of earth multiplied by time.

Using vector decomposition to find the ECI basis vectors in terms of ECEF basis vectors:

$$\boldsymbol{I}_x = \cos\Omega_e t\, \boldsymbol{e}_x - \sin\Omega_e t\, \boldsymbol{e}_y$$

$$\boldsymbol{I}_y = \sin\Omega_e t\, \boldsymbol{e}_x + \cos\Omega_e t\, \boldsymbol{e}_y$$

$$\boldsymbol{I}_z = \boldsymbol{e}_z$$

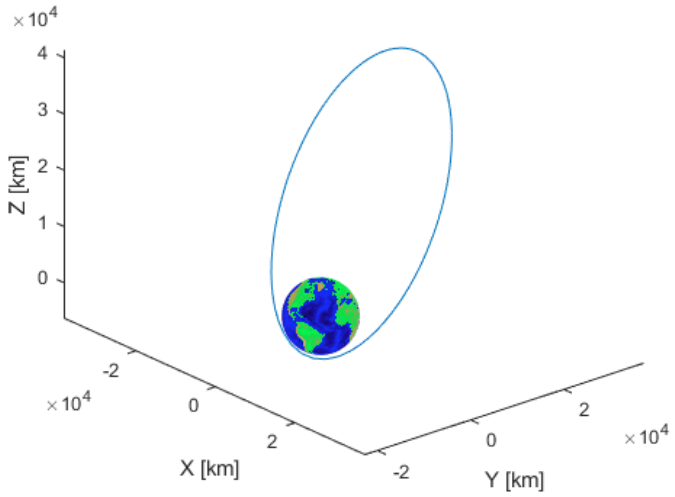
This gives us the transformation matrix:

$$\boldsymbol{T}_I^E = \begin{Bmatrix} \cos\Omega_e t & -\sin\Omega_e t & 0 \\ \sin\Omega_e t & \cos\Omega_e t & 0 \\ 0 & 0 & 1 \end{Bmatrix}$$

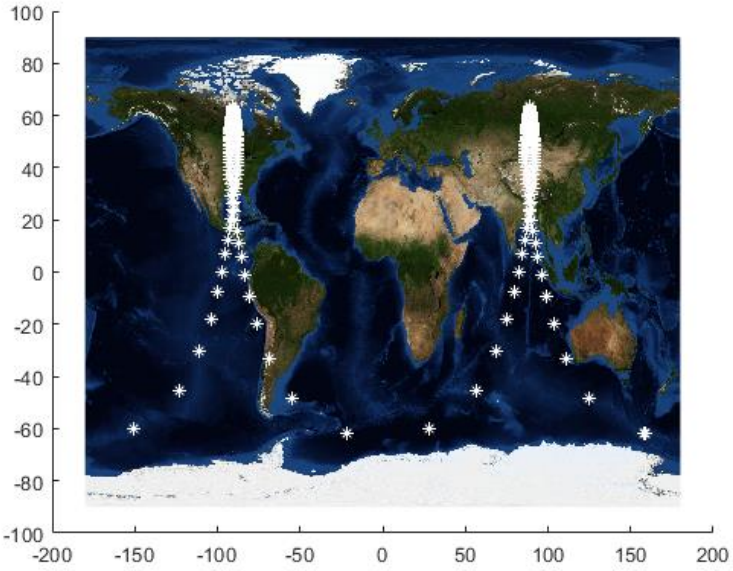
```
function TI2E = dcmI2E(t,OmegaE)
% ------------------------------------------------------------%
% ------------------------- DCMI2E.M -------------------------%
% ------------------------------------------------------------%
% Compute the transformation matrix from Earth-centered inertial%
% Cartesian coordinates to Earth-centered Earth-fixed Cartesian %
% coordinates. The transformation matrix is of size 3 by 3 and %
% represents the transformation matrix at the time t %
% Inputs: %
% t: the time at which the transformation is desired %
% OmegaE: the rotation rate of the Earth %
% Outputs: %
% TI2E: 3 by 3 matrix that transforms components of a %
% vector expressed in Earth-centered inertial %
% Cartesian coordinates to Earth-centered %
% Earth-fixed Cartesian coordinates %
% ------------------------------------------------------------%
% NOTE: the units of the inputs T and OMEGAE must be consistent %
% and the angular rate must be in RADIANS PER TIME UNIT %
% ------------------------------------------------------------%

theta = OmegaE * t;

TI2E = [cos(theta),-sin(theta),0;sin(theta),cos(theta),0;0,0,1];

end


function rvValuesECEF = eci2ecef(tValues,rvValuesECI,OmegaE)
% ------------------------------------------------------------%
% ------------------------ ECI2ECEF.M ------------------------%
% ------------------------------------------------------------%
% Transform the values of the position along a spacecraft orbit %
% from Earth-centered inertial Cartesian coordinates to %
% Earth-centered Earth-fixed Cartesian coordinates. %
% represents the transformation matrix at the time t %
% Inputs: %
% tValues: column matrix of values of time at which the %
% transformation is to be performed %
% rvValuesECI: matrix of values of the position expressed %
% in Earth-centered inertial Cartesian %
% coordinates and stored ROW-WISE %
% OmegaE: the rotation rate of the Earth %
% Outputs: %
% rvValuesECEF: matrix of values of the position expressed %
% in Earth-centered Earth-fixed Cartesian %
% and stored ROW-WISE %
% ------------------------------------------------------------%

rvValuesECEF=zeros(length(tValues),3);
for i=1:length(tValues)
    t = tValues(i);
    TI2E = dcmI2E(t,OmegaE);
    rECI = rvValuesECI(i,:);
    rvValuesECEF(i,:) = rECI*TI2E;
end

end
```

```
function [lonE,lat] = ecef2LonLat(rvValuesECEF)
% ------------------------------------------------------------%
% ---------------------- ECEF2LONLAT.M ----------------------%
% ------------------------------------------------------------%
% Compute the Earth-relative longitude and geocentric latitude %
% from the Earth-centered Earth-fixed position. %
% Input: %
% rvValuesECEF: matrix of values of the position expressed %
% in Earth-centered Earth-fixed Cartesian %
% and stored ROW-WISE %
% Outputs: %
% lonE: a column matrix containing the values of the %
% Earth-relative longitude (radians) %
% lat: a column matrix containing the values of the %
% geocentric latitude (radians) %
% ------------------------------------------------------------%

lonE = zeros(length(rvValuesECEF),1);
lat = zeros(length(rvValuesECEF),1);

for i=1:length(rvValuesECEF)
     lonE(i)=atan2(rvValuesECEF(i,2),rvValuesECEF(i,1));
     lat(i)=atan2(rvValuesECEF(i,3),sqrt(rvValuesECEF(i,1)^2+rvValuesECEF(i,2)^2));
end

end
```

Task 4:




```
function mercatorDisplay(lonE,lat)
earth = imread('earth.jpg');
% Open a new figure, then run the command "clf"
% Example:
figure(2);
clf
% Then run the code below.
image('CData',earth,'XData',[-180 180],'YData',[90 -90])
hold on
plot(lonE*180/pi,lat*180/pi,'w*');
end
```



Main Function:

```
clc;clear;close all;

%Defines position and velocity, calculates oribital elements and names them
rv0 = [-1217.39430415697,-3091.41210822807,-6173.40732877317];
vv0 = [9.88635815507896,-0.446121737099303,-0.890884522967222];
mu = 398600;
oe = rv2oe_Hackbardt_Chris(rv0,vv0,mu);
a = oe(1);
e = oe(2);
Omega = oe(3);
i = oe(4);
omega = oe(5);
p = oe(1)*(1-e^2);

%Calculates the period in seconds
period = 2*pi*sqrt(a^3/mu);

%Sets deltat to five minutes (code will calculate earth coordinates at a
%point every five minutes on the orbit)
%Calculates t0 to be time from apoapsis
deltat = 5*60;
initialt=timeChangeIntegral(@timeChangeIntegrand,pi,oe(6),p,e,mu,20);
finalt = (2*period)+initialt;
times = initialt:deltat:finalt;
times = times';

%Calculates nu, position, and velocity values for points every five minutes
on orbit
nuValues = computeNu(times,rv0,vv0,mu);
[rvValuesECI,vvValues] = computeRandV(nuValues,a,e,Omega,i,omega,mu);

%Rotation rate of earth
OmegaE = (2*pi)/((23*3600)+(56*60)+4);

%Converts ECI postion to ECEF and then to longitude and latitude
rvValuesECEF = eci2ecef(times,rvValuesECI,OmegaE);
[lonE,lat] = ecef2LonLat(rvValuesECEF);

%Creates earth and plots orbit
earthSphere
hold on
plot3(rvValuesECI(:,1),rvValuesECI(:,2),rvValuesECI(:,3))
view(49.5,22.8)

%Plots the longitudes and latitudes
mercatorDisplay(lonE,lat)

%Converts longitudes and latitudes to degrees
lonE = rad2deg(lonE);
lat = rad2deg(lat);

%Finds the two coordinates where latitude is a local maximum
westLon=[];
```

```
westLat=[];
eastLon=[];
eastLat=[];
for i=1:length(lonE)
    if lonE(i)<0
        westLon=[westLon;lonE(i)];
        westLat=[westLat;lat(i)];
    else
        eastLon=[eastLon;lonE(i)];
        eastLat=[eastLat;lat(i)];
    end
end
[maxWestLat,i] = max(westLat);
[maxEastLat,j] = max(eastLat);
maxWestLon = westLon(i);
maxEastLon = eastLon(i);

%Prints the answers
fprintf('Part a:\n');
fprintf('The segments at the highest latitudes on earth are at apoapsis
on\n');
fprintf('the orbit\n\n');
fprintf('Part b:\n');
fprintf('The coordinates where greatest latitude occurs are:\n');
fprintf('%g degrees east, %g degrees north\n',maxEastLon,maxEastLat);
fprintf('%g degrees west, %g degrees north\n\n',maxWestLon,maxWestLat);
fprintf('Part c:\n');
fprintf('The city in the east is: Tutonchany, Russia\n');
fprintf('The location in the west is: Nunavut, Canada, on the coast of Hudson
Bay\n\n');
fprintf('Part d:\n');
fprintf('The spacecraft spends most of its time in the northern regions of
earth since the orbit\n');
fprintf('is inclined and is highly eccentric. The eccentricity vector points
towards the southern\n');
fprintf('hemisphere of earth, meaning periapsis occurs over the southern
hemisphere, and apoapsis\n');
fprintf('occurs above the northern hemisphere. We also know that the speed of
a spacecraft at apoapsis\n');
fprintf('is slower than its speed at periapsis. These results combined
explain why the spacecraft\n');
fprintf('spends most of its time above the locations on earth mentioned.\n');
```

Code Output:

Part a:

The segments at the highest latitudes on earth are at apoapsis on

the orbit

Part b:

The coordinates where greatest latitude occurs are:

89.0239 degrees east, 63.3991 degrees north

-90.4902 degrees west, 63.3991 degrees north

Part c:

The city in the east is: Tutonchany, Russia

The location in the west is: Nunavut, Canada, on the coast of Hudson Bay

Part d:

The spacecraft spends most of its time in the northern regions of earth since the orbit is inclined and is highly eccentric. The eccentricity vector points towards the southern hemisphere of earth, meaning periapsis occurs over the southern hemisphere, and apoapsis occurs above the northern hemisphere. We also know that the speed of a spacecraft at apoapsis is slower than its speed at periapsis. These results combined explain why the spacecraft spends most of its time above the locations on earth mentioned.