Table of Contents

..... 1
Workspace Prep 1
PART 1: Converting Mean Solar to JD 1
PART 2: Calculating Local Sidereal Time in Degrees 1
PART 3: ODE45 Problem 2
Functions: 3


```
%Roshan Jaiswal-Ferri
%Section - 01
%Aero 351 Homework 1 - Datetime Calcs: 9/23/24
```

Workspace Prep

```
format long     %Allows for more accurate decimals
close all;      %Clears all
clear all;      %Clears Workspace
clc;            %Clears Command Window
```

PART 1: Converting Mean Solar to JD

```
day = 22;
month = 9;
year = 2024;
ut = '8:00:00'; %enter time as a string in hh:mm:ss format (24hr time)

JDtime = tojd(day,month,year,ut);
JDtime2 = juliandate(2024,9,22,8,0,0);

disp('Julian Date of September 22, 2024:')
disp(['My function: ', num2str(JDtime)])
disp(['Built in function: ', num2str(JDtime2)])
disp(' ')
```

```
Julian Date of September 22, 2024:
My function: 2460575.8333
Built in function: 2460575.8333
```

PART 2: Calculating Local Sidereal Time in Degrees

```
disp('LST in Melbourne, AU, 12/21/2007, 10am UT')
LST = sidereal(144,58,'e',12,21,2007,'10:00:00');
disp(num2str(LST))

disp('LST in SLO, US, 7/4/2024, 12:30pm UT')
```

```
LST = sidereal(120.653,0,'w',7,4,2024,'12:30:00');
disp(num2str(LST))
disp(' ')
```

LST in Melbourne, AU, 12/21/2007, 10am UT
24.5646
LST in SLO, US, 7/4/2024, 12:30pm UT
349.8577

PART 3: ODE45 Problem

```
muEarth = 398600; %km^3/s^2
rVect = [3207 5459 2714]; %pos
vVect = [-6.532 0.7835 6.142]; %km/s (using km and seconds as units)

timespan = [0 5*60*60]; %5hrs into seconds (units need to match)

state = [rVect vVect]; %creating input variable

%outputs = ode45(@filename w/ eqs of motion,time,state,options,anything else)

%INPUTS MUST BE IN THAT ORDER UNTIL OPTIONS

options = odeset('RelTol',1e-8,'AbsTol',1e-8);%ALWAYS CHANGE THESE

[timeNew,stateNew] = ode45(@twobodymotion,timespan,state,options,muEarth);

endPosVec = [stateNew(end,1), stateNew(end,2), stateNew(end,3)];
origin = [0, 0, 0];

mag = norm(endPosVec);
disp(['Magnitude of the distance (km): ', num2str(mag)]);


finalVelVec = [stateNew(end,4), stateNew(end,5), stateNew(end,6)];

finalVelMag = norm(finalVelVec);
disp(['Magnitude of the final velocity (km/s): ', num2str(finalVelMag)]);


figure('Name','2D Plot')
plot(stateNew(:,1),stateNew(:,2)) %all rows in col 1 then all in col 2
xlabel('x Km')
ylabel('y Km')
%Mark start and end points
hold on
plot(stateNew(1,1), stateNew(1,2), 'ro', 'MarkerSize', 10, 'DisplayName',
'Start Point')
plot(stateNew(end,1), stateNew(end,2), 'go', 'MarkerSize', 10,
'DisplayName', 'End Point')
legend('Orbit Path', 'Start Point', 'End Point', Location='best')
```

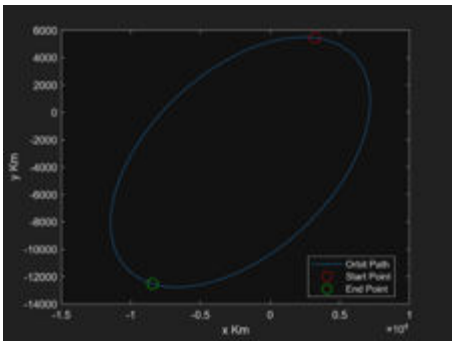
```
figure('Name','3D Plot')
plot3(stateNew(:,1),stateNew(:,2),stateNew(:,3))
xlabel('x Km')
ylabel('y Km')
zlabel('z Km')
%Mark start and end points
hold on
plot3(stateNew(1,1), stateNew(1,2), stateNew(1,3), 'ro', 'MarkerSize', 10,
'DisplayName', 'Start Point')
plot3(stateNew(end,1), stateNew(end,2), stateNew(end,3), 'go', 'MarkerSize',
10, 'DisplayName', 'End Point')
legend('Orbit Path', 'Start Point', 'End Point', Location='best')
```

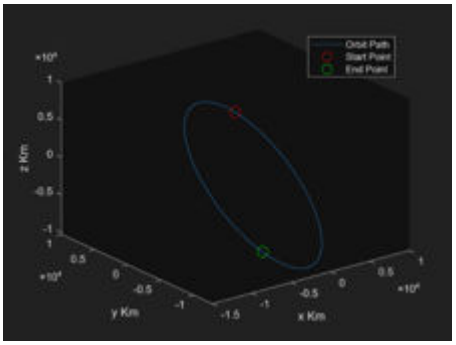
Functions:

```
function dstate = twobodymotion(time,state,muEarth) %dstate is derivitve of
state
    %define vars
    x = state(1);
    y = state(2);
    z = state(3);
    dx = state(4); %vel
    dy = state(5); %vel
    dz = state(6); %vel

    %mag of pos vector
    r = norm([x y z]);

    %accel: !!eqs of motion!!
    ddx = -muEarth*x/r^3;
    ddy = -muEarth*y/r^3;
    ddz = -muEarth*z/r^3;

    dstate = [dx; dy; dz; ddx; ddy; ddz];

end

function [LST] = sidereal(long, min, dir, month, day, year, ut)
    %Constants / Vars
    julianCentury = 36525; %days
    J2000 = 2451545;

    [~, J0, deciTimeUT] = tojd(day, month, year, ut);
    T0 = (J0-J2000)/julianCentury;
    GST = 100.4606184 + 36000.77004*T0 + 0.000387933*T0^2 - 2.58*10^-8*T0^3;
%Greenwhich Siderial Time in degs
    GST = GST + 360.98564724*(deciTimeUT/24); %add UT time

    if min
        long = long + (min/60);
    end
```

```
    if dir == 'e'
        LST = GST + long;
    elseif dir == 'w'
        deg = 360 - long;
        LST = GST + deg;
    else
        disp('Longitude Error')
    end

    if LST > 360
        while LST > 360
            LST = LST - 360;
        end
    end

end

function [JD, J0, deciTime] = tojd(day, month, year,ut)

    %Floor is the integer portion of x, Y is the 4-digit year, M is the
2-digit month, and D
    %is the 2-digit day of the month

    timeV = sscanf(ut, '%d:%d:%d');
    hours = timeV(1);
    minutes = timeV(2);
    seconds = timeV(3);

    deciTime = hours + (minutes/60) + (seconds/3600);


    J0 = 367*year - floor((7*(year+floor((month+9)/12)))/4)
+floor((275*month)/9) + day + 1721013.5;
    JD = J0 + (deciTime/24);

end
```

Magnitude of the distance (km): 16029.6629
Magnitude of the final velocity (km/s): 3.8741





Published with MATLAB® R2023b