Table of Contents

Homework 2

Aero 351 Liam Hood

```
function Aero_351_HW2
```

Set up

```
clear ;
clc ;
close all ;

r_e = 6378 ; % radius of Earth in km
mu_e = 398600 ;
r2d = 180/pi ;
d2r = pi/180 ;
```

2.24

```
disp( '2.24' )

r_e = 6378 ; % radius of Earth in km
mu_e = 398600 ;
r2d = 180/pi ;
d2r = pi/180 ;

% Given
z = 500 ; % altitude(km)
v = 10 ; % speed (km/s)
theta = 120 ; % true anomaly in degrees

% Needed intermediate values
rp = z + r_e ; % radius of perigee
h = v*rp ; % angular momentum (km^3/s)
```

```
e = ( h^2/(mu_e*rp) ) - 1 ; %eccentricity

%Answers
gamma = atand( (e*sind(theta)) / (1+e*cosd(theta)) ) ; % flight path
 angle at this point
r_f = ( h^2/mu_e )*( 1/(1+e*cosd(theta)) ) ; %radius of orbit at this
 point
z_f = r_f - r_e ; % altitude at this point

disp([ 'The flight path angle at a true anomally of 120 degrees is ' ,
 num2str( gamma ) , ' degrees' ])
disp([ 'The altitude of the orbit at this point is ' ,
 num2str( z_f ) , ' km' ])
disp( 'These answers make sense because the orbit is fairly eccentric
 and nearer ')
disp( 'perigee than apogee so speed should be a little greater than
 normal LEO' )
disp( 'and the flight path angle should be significant' )
```

```
2.24
The flight path angle at a true anomally of 120 degrees is 44.5973
 degrees
The altitude of the orbit at this point is 12246.7575 km
These answers make sense because the orbit is fairly eccentric and
 nearer
perigee than apogee so speed should be a little greater than normal
 LEO
and the flight path angle should be significant
```

2.37

```
disp( '2.37' )
clear

r_e = 6378 ; % radius of Earth in km
mu_e = 398600 ;
r2d = 180/pi ;
d2r = pi/180 ;

% Given
z_p = 250 ; % km
v_p = 11 ; % km/s
r_p = z_p + r_e ; %km

% intermediate values
    h = r_p*v_p ; % angular momentum
    ecc = (h^2/(mu_e*r_p))-1 ;
    a = (h^2/(mu_e))*(1/(ecc^2-1)) ; % km

%a
disp( 'a' )
    % hyperbolic excess speed
    v_inf = sqrt( mu_e / a ) ;
```

```
    disp([ 'Hyperbolic excess speed ' , num2str( v_inf ) , ' km/s'
 ]) ;

%b
disp( 'b' )
    % radius when true anomaly is 100 degrees
    r_100 = (h^2/mu_e)*(1/(1+ecc*cosd(100))) ;
    disp([ 'Radius when true anomaly is 100 degrees ' ,
 num2str( r_100 ) , ' km' ])

%c
disp( 'c' )
    % velocity radial and azmuthal at 100
    v_az = (mu_e/h)*(1+ecc*cosd(100)) ; % azmuthal velocity
    v_r = (mu_e/h)*ecc*sind(100) ; % radial velocity
    disp([ 'The azmuthal velocity is ' , num2str( v_az ) , ' km/s' ])
    disp([ 'The radial velocity is ' , num2str( v_r ) , ' km/s' ])
```

```
2.37
a
Hyperbolic excess speed 0.84994 km/s
b
Radius when true anomaly is 100 degrees 16178.7779 km
c
The azmuthal velocity is 4.5064 km/s
The radial velocity is 5.4488 km/s
```

2.38

```
clear
disp( '2.38' )

r_e = 6378 ; % radius of Earth in km
mu_e = 398600 ;
r2d = 180/pi ;
d2r = pi/180 ;

% Given
r_i = 402000 ; % km
ta = 150*d2r ; % true anomaly in radians
v_i = 2.23 ; % km/s

% Intermediate
vesc = sqrt( 2*mu_e/r_i ) ; % escape velocity

v_az = sqrt( v_i^2 / ( 1 + tan( ta ) ) ) ; % azmuthal velocity
h = r_i*v_az ; % angular momentum
epsilon = (v_i^2/2) - (mu_e/r_i) ; % specific energy
a = mu_e/(2*epsilon) ; % semi-major axis (km)

% a
disp( 'a' )
    % eccentricity
```

```
    %ecc = (h^2/(mu_e*r_p))-1 ;
    disp( 'no idea how to get eccentricity' )
```

2.38
a
no idea how to get eccentricity

3.8

```
disp( '3.8' )
clear

r_e = 6378 ; % radius of Earth in km
mu_e = 398600 ;
r2d = 180/pi ;
d2r = pi/180 ;

% Given
z_a = 600 ; % altitude at apogee km
z_p = 200 ; % altitude at perigee km
z_t = 400 ; % altitude of interest km

% Calculations
r_a = z_a + r_e ; % radius of apogee km
r_p = z_p + r_e ; % radius of perigee km
r_t = z_t + r_e ; % radius of interest km

ecc = ( r_a - r_p )/( r_a + r_p ) ; % eccentricity
a = .5*( r_a + r_p ) ; % semi-major axis
epsilon = -.5*(mu_e/a) ; % specific energy
v_p = sqrt( 2*( epsilon + (mu_e/r_p) ) ) ; % velocity
h = v_p*r_p ; % angular momentum

theta1 = acos( ( (h^2/(mu_e*r_t)) - 1 )/ecc ) ; % true anomaly at
 target radius
theta2 = 2*pi - theta1 ; % true anomaly at following true anomaly
T = (2*pi/sqrt(mu_e))*a^(3/2) ; % Period of orbit
n = 2*pi/T ; % mean motion
E = 2*atan(sqrt((1-ecc)/(1+ecc))*tan( theta1/2 )) ; %
Me = E - ecc*sin(E) ; %
t = Me/n ; % Time from perigee to target radius
time = T-2*t ; % Time above target altitude
time_real = time/60 ; % Time in minutes

disp([ 'The time spent above an altitude of 400 km is ' ,
 num2str( time_real ) , ' minutes' ])
disp( 'This answer makes sense as this time spent above the altitude
 that is ')
disp( 'halfway between perigee and apogee is about half of the perigee
 ' )
```

3.8
The time spent above an altitude of 400 km is 47.1482 minutes
This answer makes sense as this time spent above the altitude that is

```
halfway between perigee and apogee is about half of the perigee
```

3.10

```
disp( '3.10' )
clear

r_e = 6378 ; % radius of Earth in km
mu_e = 398600 ;
r2d = 180/pi ;
d2r = pi/180 ;

% Given
T_h = 14 ; % Period in hours
r_p = 10000 ; % km
target_h = 10 ; % target position in hours

% Intermediate
t = target_h * 60^2 ; % target time in seconds
T = T_h * 60^2 ; % period in seconds
a = ( ( T * sqrt( mu_e ) ) / ( 2*pi ) )^( 2/3 ) ; % semi-major axis
 ( km )
epsilon = -( mu_e/( 2*a ) ) ; % specific energy
r_a = 2*a - r_p ; % radius of apogee
ecc = ( r_a - r_p )/( r_a + r_p ) ; % eccentricity
v_p = sqrt( 2*( epsilon + (mu_e/r_p) ) ) ; % velocity at perigee
h = r_p * v_p ; % angular momentum

[ theta ] = time2theta( t , T , ecc ) ; % theta in radians

% a, radial position
disp( 'a' )
r = (h^2/mu_e)/(1+ecc*cos(theta)) ; % new radius
disp([ 'The radius after 10 hours is ' , num2str( r ) , ' km' ])
disp( 'This makes sense as the radius is much larger than perigee and
 the ' )
disp( 'orbit is fairly eccentric' )

% b, speed
disp( 'b' )
speed = sqrt( 2*( epsilon + (mu_e/r) ) ) ; % new speed
disp([ 'The speed after 10 hours is ' , num2str( speed ) , ' km/s' ])
disp( 'This speed makes sense as it is fairly low and this is a large
 orbit')

% c, radial velocity
disp( 'c' )
v_r = ( mu_e/h ) * ecc * sin( theta ) ;
disp([ 'The radial velocity is ' , num2str( v_r ) , ' km/s' ])
disp( 'This seems right as it is negative and the s/c should be past
 halfway through')
disp( 'its orbit and its radius will shrink as it approaches perigee'
 )
```

```
3.10
a
The radius after 10 hours is 42354.9211 km
This makes sense as the radius is much larger than perigee and the
orbit is fairly eccentric
b
The speed after 10 hours is 2.3034 km/s
This speed makes sense as it is fairly low and this is a large orbit
c
The radial velocity is -1.2709 km/s
This seems right as it is negative and the s/c should be past halfway
 through
its orbit and its radius will shrink as it approaches perigee
```

3.20

```
clear
disp( '3.20' )
mu_e = 398600 ;

r0 = [ 20000 -105000 -19000 ] ; % initail position (km)
v0 = [ .9000 -3.4000 -1.5000 ] ; % initial velocity (km/s)
target_h = 2 ; % target time in hours

target = 2*60^2 ; % target time in seconds

% Use Universal anomaly
[ r , v ] = NewState( r0 , v0 , target , mu_e ) ;

% Check against ODE45
ooptions = odeset( 'RelTol' , 1e-8 , 'AbsTol' , 1e-8 ) ;
[ nt , nstate ] = ode45( @TwoBodyMotion , [ 0 target ] , [ r0 , v0 ] ,
 ooptions , mu_e ) ;
rode = nstate( 1:3 ) ;
vode = nstate( 4:6 ) ;

disp([ 'After ' , num2str( target ) , ' seconds the new state is as
 follows' ])
disp( 'New r (km)' )
disp( r )
disp( 'New v (km/s)' )
disp( v )
disp( 'ODE45 r (km)' )
disp( rode )
disp( 'ODE45 v (km/s)' )
disp( vode )
disp( 'My answers seem reasonable because they are similar to the
 starting ')
disp( 'values. The radius is large so the period should be large and
 the ' )
disp( 'changes over 2 hours should be relatively small. It is close to
 the ' )
```

```
disp( 'book answer but very different than ODE45. I don''t really know
 why' )
```

```
3.20
After 7200 seconds the new state is as follows
New r (km)
       26480     -129480      -29800

New v (km/s)
    0.8641   -3.2113   -1.4659

ODE45 r (km)
   1.0e+04 *

    2.0000    2.0057    2.0114

ODE45 v (km/s)
   1.0e+04 *

    2.0172    2.0229    2.0391

My answers seem reasonable because they are similar to the starting
values. The radius is large so the period should be large and the
changes over 2 hours should be relatively small. It is close to the
book answer but very different than ODE45. I don't really know why
```

4.5

```
clear
disp( '4.5' )

mu_e = 398600 ;
r = [ 6500 -7500 -2500 ] ; % Position vector (km) in ECI
v = [ 4 3 -3 ] ; % Velocity vector (km/s) in ECI
[ OE ] = OrbitalElements( r , v , mu_e ) ;
disp( OE )
disp( 'I think these all seem right, based on everything seeming to
 indicate ' )
disp( 'a not particularly eccentric LEO ' )
```

```
4.5
          r: {[1.0235e+04]  'km'  'radius'}
          v: {[5.8310]  'km/s'  'speed'}
         vr: {[1.0748]  'km/s'  'radial speed'}
       hvec: {[30000 9500 49500]  'km^2/s'  'angular momentum vector'}
          h: {[5.8656e+04]  'km^2/s'  'angular momentum'}
        inc: {[32.4450]  'degrees'  'inclination'}
        ecc: {[0.2226]  'unitless'  'eccentricity'}
       RAAN: {[107.5713]  'degrees'  'right ascension of ascending
 node'}
        aop: {[72.3586]  'degrees'  'argument of perigee'}
         ta: {[134.7259]  'degrees'  'true anomaly'}
    epsilon: {[-21.9458]  'km^2/s^2'  'specific energy'}
          a: {[9.0815e+03]  'km'  'semi-major axis'}
```

I think these all seem right, based on everything seeming to indicate a not particularly eccentric LEO

4.7

```
clear
disp( '4.7' )

r = [ -6600 -1300 -5200 ] ; % km
ecc = [ -.4 -.5 -.6 ] ; %
% inclination is given by acosd(h(3)/h)
% this is just finding the angle between vertical and a vector
% perpendicular to the orbital plane
% the position vector is in the orbital plane as is the eccentricity
 vector
% so their cross product will be out of the orbital plane meaning the
 angle
% between this vector and the result should be the inclination
plane = cross( r , ecc ) ;
inc = acosd( plane(3)/norm(plane) ) ;
disp([ 'The inclination is ' , num2str(inc) , ' degrees' ])
disp( 'Seems correct based on where the eccentricity vector and
 position vector ')
disp( 'are pointed.' )
```

```
4.7
The inclination is 43.2661 degrees
Seems correct based on where the eccentricity vector and position
 vector
are pointed.
```

Functions

```
function [ theta ] = time2theta( t , T , ecc )
% Find true anomaly at a time

n = 2*pi/T ; % mean motion
Me = n*t ;

% Guess of E
if Me < pi
    E0 = Me + ecc/2 ;
else
    E0 = Me - ecc/2 ;
end

% Use Newtons to find E
    tol = 10^-8 ; % Tolerance
    lim = 1000 ; % Maximum iteration
    f = @(E) E - ecc*sin(E) - Me ; % Function handle for E
    fprime = @(E) 1 - ecc*cos(E) ; % function handle for derivative of
 E
```

```
    [ E ] = newton( E0 , f , fprime , tol , lim ) ; % Apply Newtons

theta = 2*atan(tan(E/2)*sqrt((1+ecc)/(1-ecc))) ; % find true anomaly
% correction to make it positive
    if theta < 0
        theta = theta + 2*pi ;
    end
end

function [ r , v ] = NewState( r0 , v0 , dt , mu )
% Find new position and velocity at some time in orbit by lagrange
% variables

    % Initial values
    r0mag = norm( r0 ) ; % radius in km
    v0mag = norm( v0 ) ; % speed in km/s
    vr0 = dot( r0 , v0 ) / r0mag ; % radial speed (km/s)
    alpha = (2/r0mag) - ( v0mag^2 / mu ) ;
    chi = sqrt( mu ) * abs( alpha ) * dt ; % Initial universal anomaly
 guess
    z = alpha*chi(1)^2 ;
    sl = 10 ; % series length

    % Universal anomaly equations
    fun = @(chi,C,S) ((r0mag*vr0)/sqrt(mu))*chi^2*C+(1-
alpha*r0mag)*chi^3*S+r0mag*chi-sqrt(mu)*dt ;
    fprime = @(chi,C,S) ((r0mag*vr0)/sqrt(mu))*chi*(1-
alpha*chi^2*S)+(1-alpha*r0mag)*chi^2*C+r0mag ;

    % Stumpff Functions
        for kk = 1:sl
            sc(kk) = (-1)^(kk-1) / factorial(2*(kk-1)+3) ;
        end


        for kk = 1:sl
            cc(kk) = (-1)^(kk-1) / factorial(2*(kk-1)+2) ;
        end


    % Newtons for UV
    ii = 1 ;
    ratio = 1 ;
    tol = 10^-8 ;
    lim = 1000 ;
        while abs(ratio(ii)) >= tol

            % Stumpff calc
                for jj = 1:sl
                    S = sc(jj)*z^(jj-1) ;
                end
                for jj = 1:sl
                   C = cc(jj)*z^(jj-1) ;
                end
```

```
            ratio(ii+1) = fun(chi,C,S)/fprime(chi,C,S) ;
            chi = chi - ratio(ii+1) ;
            ii = ii + 1 ;
            z = alpha*chi^2 ; % New z
                if ii > lim
                    error([ 'Ran ' , num2str( lim ) , ' times without
 a solution' ])
                end
        end
    % Final Stumpff
    for jj = 1:sl
        S = sc(jj)*z^(jj-1) ;
    end
    for jj = 1:sl
        C = cc(jj)*z^(jj-1) ;
    end


    % Lagrange variables

    % new r
    f = 1 - (chi^2/r0mag)*C ;
    g = dt - (1/sqrt(mu))*chi^3*S ;
    r = f*r0 + g*v0 ; % new position
    rmag = norm( r ) ; % radius

    % new v
    fdot = (sqrt(mu)/(r0mag*rmag))*(alpha*chi^3*S-chi) ;
    gdot = 1 - (chi^2/rmag)*C ;
    v = fdot*r0 + gdot*v0 ; % new velocity

end

function [ OE ] = OrbitalElements( r , v , mu )

    r2d = 180/pi ; % radians to degrees

    Kh = [ 0 0 1 ] ; % K hat

    distance = norm( r ) ;
    speed = norm( v ) ;
    vr = dot( r , v )/distance ; % radial velocity
    h = cross( r , v ) ; % specific angular momentum
    hmag = norm( h ) ; % specific angular momentum
    inc = acos(h(3)/norm(h)) ; %inclination
    eccv = (1/mu)*( cross(v,h)-mu*(r/distance) ) ; %eccentricity
 vector
    ecc = norm( eccv ) ; % eccentricity
    Nv = cross( Kh , h ) ; % Node line
    N = norm( Nv ) ;

    if Nv(2) > 0
        RAAN = acos(Nv(1)/N) ; %Right ascension of ascending node
```

```
   elseif Nv(2) < 0
       RAAN = 2*pi - acos(Nv(1)/N) ; %Right ascension of ascending
node
   else
       RAAN = 'Undefined' ;
   end

   if eccv(3) > 0
       aop = acos(dot(Nv,eccv)/(N*ecc)) ; % Argument of perigee
   elseif eccv(3) < 0
       aop = 2*pi - acos(dot(Nv,eccv)/(N*ecc)) ; % Argument of
perigee
   else
       aop = 'Undefined' ;
   end

   % True anomaly
   if vr >= 0
       ta = acos( dot(eccv,r)/(ecc*distance) ) ;
   else
       ta = 2*pi - acos( dot(eccv,r)/(ecc*distance) ) ;
   end

   epsilon = speed^2/2 - mu/distance ; % specific energy
   a = - mu/(2*epsilon) ; % semi-major axis

   OE.r = { distance , 'km' , 'radius' } ; % radius
   OE.v = { speed , 'km/s' , 'speed' } ; % speed
   OE.vr = { vr , 'km/s' , 'radial speed' } ; % radial speed
   OE.hvec = { h , 'km^2/s' , 'angular momentum vector' } ; % angular
momentum vector
   OE.h = { hmag , 'km^2/s' , 'angular momentum' }; % angular
momentum
   OE.inc = { r2d*inc , 'degrees' , 'inclination' }; % inclination
   OE.ecc = { ecc , 'unitless' , 'eccentricity' }; % eccentricity
   OE.RAAN = { r2d*RAAN , 'degrees' , 'right ascension of ascending
node' } ; % right ascension of ascending node
   OE.aop = { r2d*aop , 'degrees' , 'argument of perigee' } ; %
argument of perigee
   OE.ta = { r2d*ta , 'degrees' , 'true anomaly' }; % true anomaly
   OE.epsilon = { epsilon , 'km^2/s^2' , 'specific energy' } ; %
specific energy
   OE.a = { a , 'km' , 'semi-major axis' } ; % semi-major axis

end

function [ x ] = newton( x0 , f , fprime , tol , lim )
% Uses Newtons Method to find x given initial guess x0, function f,
% derivative fprime, tolerance tol, and limit on the iterations lim

    x = x0 ;
    ii = 1 ;
    ratio = 1 ;
        while abs(ratio(ii)) >= tol
```

```
            ratio(ii+1) = f(x(ii))/fprime(x(ii)) ;
            x(ii+1) = x(ii) - ratio(ii+1) ;
            ii = ii + 1 ;
                if ii > lim
                    error([ 'Ran ' , num2str( lim ) , ' times without
 a solution' ])
                end
        end
x = x( ii ) ;
end

function dstate_dt = TwoBodyMotion( t , state , mu )
% Finds change in state with respect to time. Input time, t, in
 seconds and
% state as position vector followed by velocity vector as well as mu

rad = norm( [ state(1) state(2) state(3) ] ) ; %radius

dx = state(4) ; % velocity in x
dy = state(5) ; % velocity in y
dz = state(6) ; % velocity in z
ddx = -mu*state(1)/rad^3 ; % acceleration in x
ddy = -mu*state(2)/rad^3 ; % acceleration in y
ddz = -mu*state(3)/rad^3 ; % acceleration in z

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

end

end
```

Published with MATLAB® R2017b