Table of Contents

Homework 1

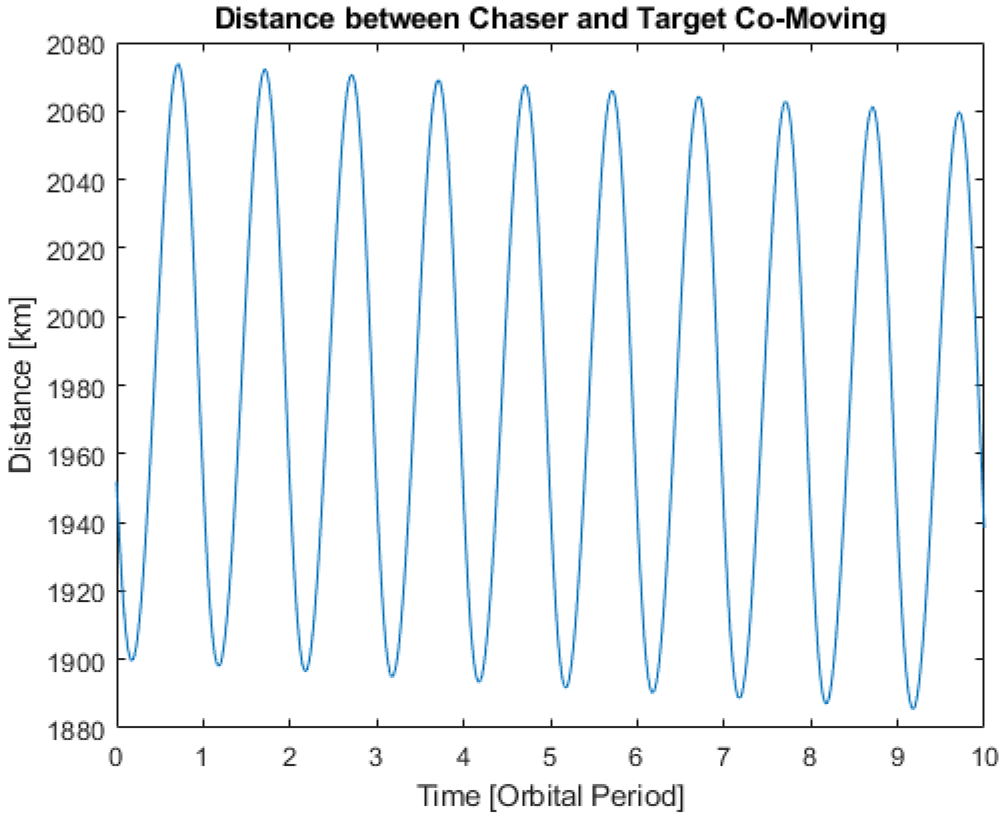
Aero 452 Liam Hood

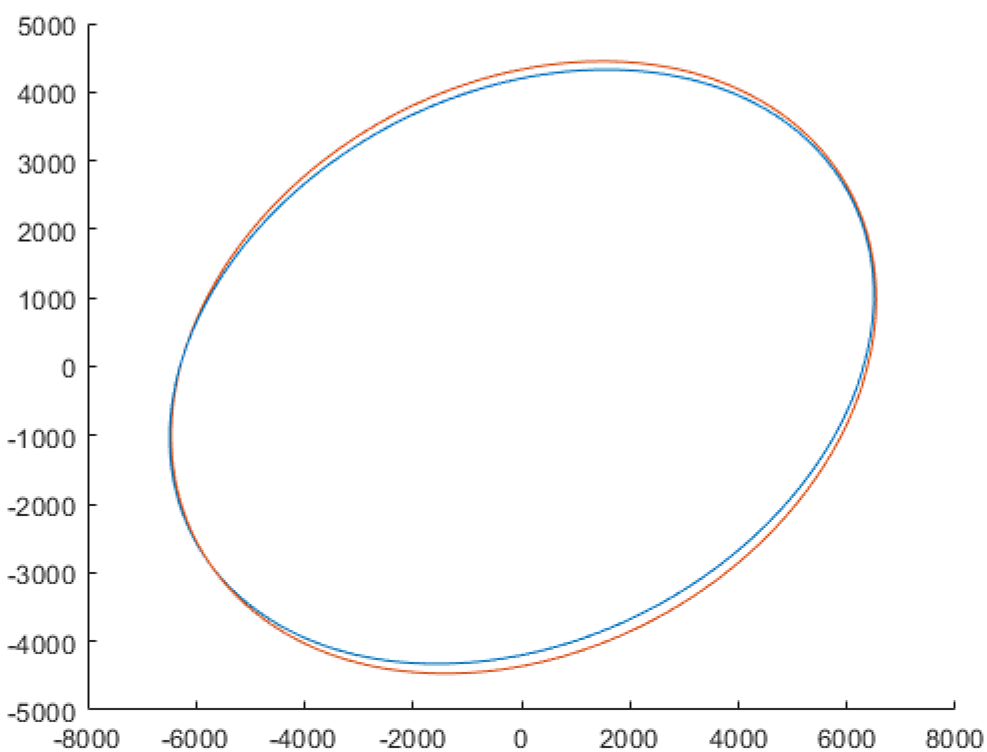
```
function HW1

clear ; close all ; clc ;
mu = 398600 ;
stumpfTerms = 10 ;
[ denomS , denomC ] = StumpffSetUp( stumpfTerms ) ;
pt = 'Problem number %u \n \n' ;
```

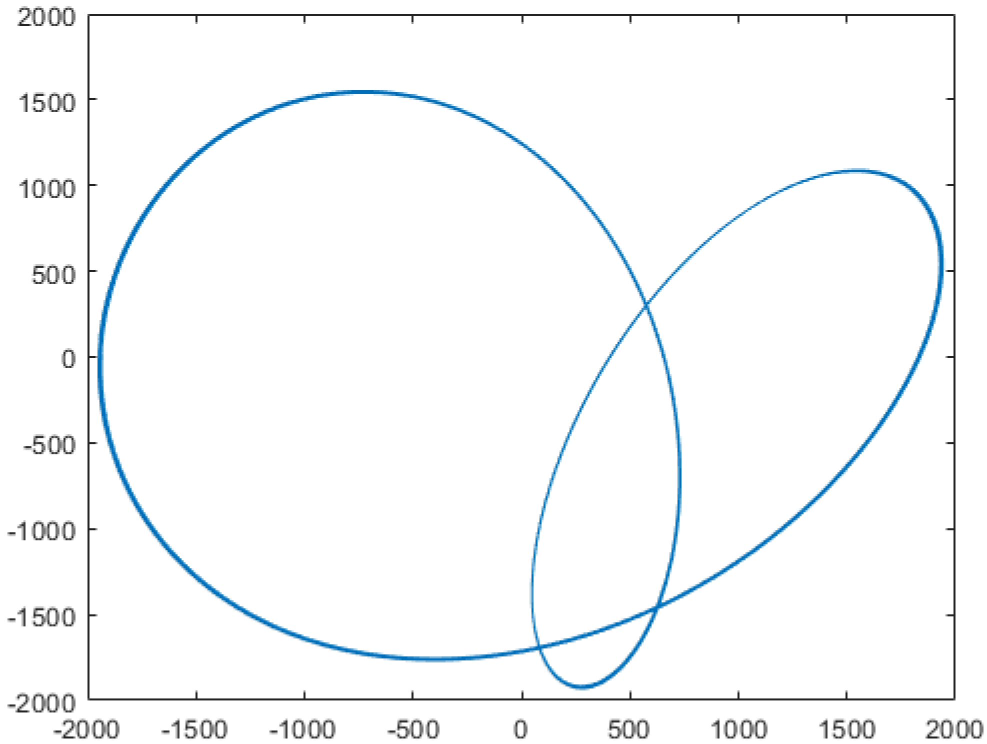
1

```
fprintf( pt , 1 )
% far distance, 10 periods
HW1_P1(mu,denomS,denomC)
fprintf( ' \n' )
```

```
Problem number 1

The closest the satellites get is 1885.589979 km after 13.686111
 hours
I used Vallado's Universal Variable code because mine was
only working if I set the time step absurdly small and let
the code run for a long time. I am going to try to fix that
but I didn't figure it out for this homework.
```


Distance between Chaser and Target Co-Moving
Distance [km]
Time [Orbital Period]
2080
2060
2040
2020
2000
1980
1960
1940
1920
1900
1880
0
1
2
3
4
5
6
7
8
9
10


5000
4000
3000
2000
1000
0
-1000
-2000
-3000
-4000
-5000
-8000
-6000
-4000
-2000
0
2000
4000
6000
8000



2

```
fprintf( pt , 2 )
% circular point in time
HW1_P2(mu,denomS,denomC)
fprintf( ' \n' )
```

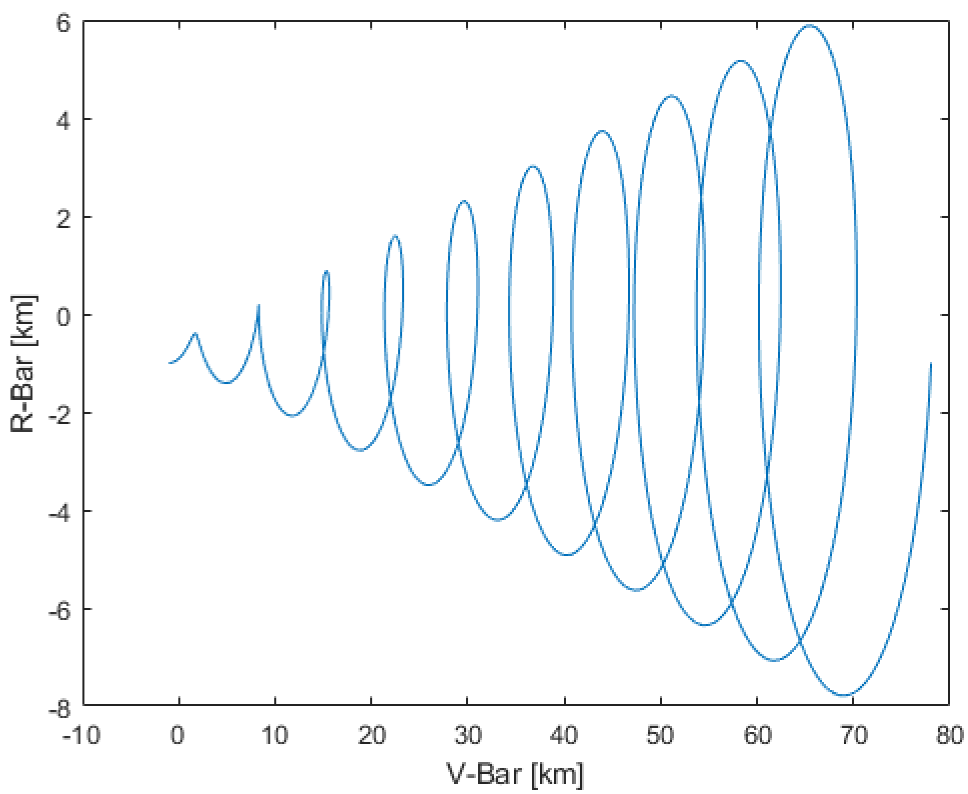
```
Problem number 2

When A is over the equator and B is over the north pole
The position of B relative to A is:
-6678.000000 km
6628.000000 km
0.000000 km
The velocity of B relative to A is:
-0.086932 km/s
0.000000 km/s
0.000000 km/s
The acceleration of B relative to A is:
0.000000e+00 km/s^2
-1.140179e-06 km/s^2
0.000000e+00 km/s^2
```

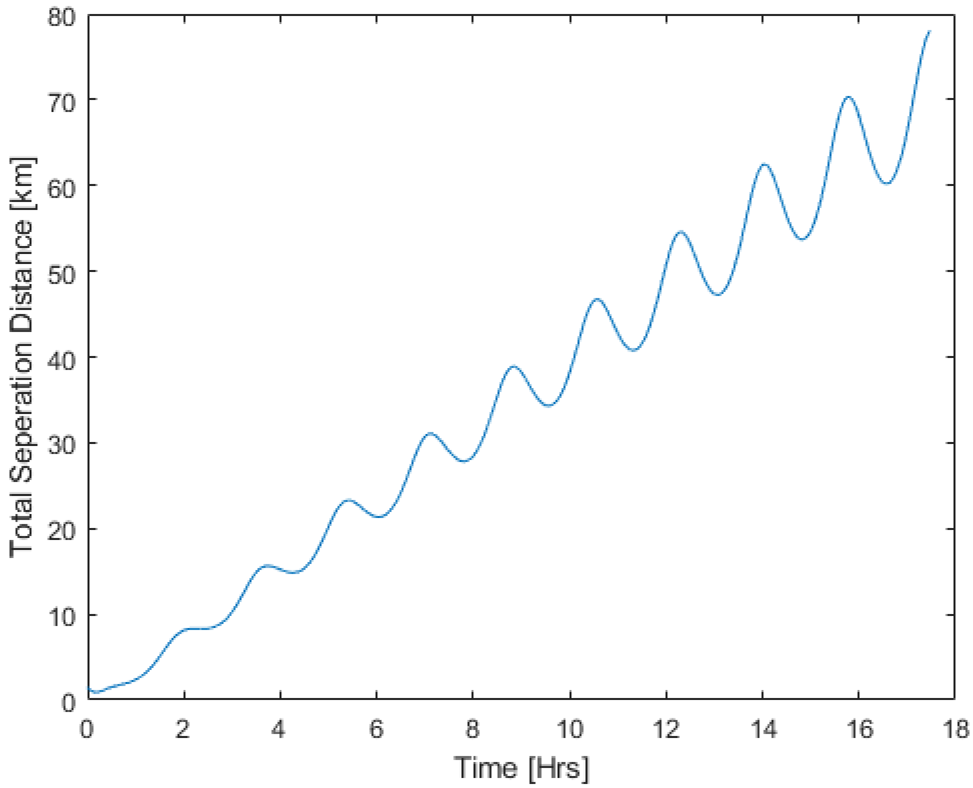
3

```
fprintf( pt , 3 )
% close range, 10 periods
HW1_P3(mu,denomS,denomC)
fprintf( ' \n' )
```

Problem number 3

After 10 periods the distance between the two satellites is 78.040172 km.
The closest that they get is 0.896487 km after 0.179650 hours
These plots look correct as the distance oscillates in the R-bar direction and grows in the V-bar direction. There are also 10 cycles of growing oscillations which makes sense for 10 orbital periods





4

```
fprintf( pt , 4 )
% close range, 15 min
HW1_P4(mu,denomC,denomS)
```

Problem number 4

After 15 minutes the satellite is 11.221554 km from the space station
This seems reasonable seperation after a sixth of a period
in a low orbit

Work

```
function HW1_P1(mu,denomS,denomC)
    t = 0 ;
    tstep = 10 ; % step size in seconds

    % Satellite A orbit definition
    hA = 51400 ; % km2/s
    eccA = 0.0006387 ;
    incA = 51.65 ; %degs
    raanA = 15 ; %degs
    omegaA = 157 ; %degs
    thetaA = 15 ; %degs
```

```
        Period = (2*pi/mu^2)*(hA/(sqrt(1-eccA^2)))^3 ;
        [ rA , vA ] = coes2state(  hA , eccA , thetaA , raanA ,
 omegaA , incA , mu ) ; % state of A in ECI

        % Satellite B orbit definition
        hB = 51398 ; % km2/s
        eccB = 0.0072696 ;
        incB = 50 ; %degs
        raanB = 15 ; %degs
        omegaB = 140 ; %degs
        thetaB = 15 ; %degs
        [ rB , vB ] = coes2state(  hB , eccB , thetaB , raanB ,
 omegaB , incB , mu ) ; % state of A in ECI

        % propogation and calculations
        for ii = 2:(10*Period/tstep)
            Percent = 100*ii/(10*Period/tstep) ;
%             [ rA(:,ii) , vA(:,ii) ] = NewStateUV( rA(:,ii-1) ,
 vA(:,ii-1) , tstep , mu , denomS , denomC ) ; % Step forward for
 target
%             [ rB(:,ii) , vB(:,ii) ] = NewStateUV( rB(:,ii-1) ,
 vB(:,ii-1) , tstep , mu , denomS , denomC ) ; % Step forward for
 chaser
            [ rA(:,ii) , vA(:,ii) ] =  kepler  ( rA(:,ii-1) ,
 vA(:,ii-1) , tstep ) ;
            [ rB(:,ii) , vB(:,ii) ] =  kepler  ( rB(:,ii-1) ,
 vB(:,ii-1) , tstep ) ;

            t(ii) = t(ii-1) + tstep ; % Step forward time
            [ C_xX , C_Xx ] = LVLHfromState( rA(:,ii) , vA(:,ii) ) ; %
 rotation between ECI and LVLH

            % position calculations
            rrel(:,ii) = rB(:,ii) - rA(:,ii) ;
            rrelcm(:,ii) = C_xX*rrel(:,ii) ;
            rbar(ii) = norm( rrel(:,ii) ) ;
            rbarcm(ii) = norm( rrelcm(:,ii) ) ;

            % absolute angular velocity of the moving frame
            omega = cross( rA(:,ii) , vA(:,ii) )/norm( rA(:,ii) )^2 ;
            omegadot = -2*dot( vA(:,ii) , rA(:,ii) )*omega /
 norm( rA(:,ii) ) ;

            % velocity calculations
            vrel(:,ii) = vB(:,ii) - vA(:,ii) - cross( omega ,
 rrel(:,ii) ) ;
            vrelcm(:,ii) = C_xX*vrel(:,ii) ;

        end

        % distance plot
        figure
        plot( t(2:end)/Period , rbarcm(2:end) )
        title( 'Distance between Chaser and Target Co-Moving' )
```

```
        xlabel( 'Time [Orbital Period]' )
        ylabel( 'Distance [km]' )

        % orbits in ECI
        figure
        hold on
        plot3( rA(1,2:end) , rA(2,2:end) , rA(3,2:end) )
        plot3( rB(1,2:end) , rB(2,2:end) , rB(3,2:end) )
        hold off

        % relative position between the two
        figure
        plot( rrelcm(2,2:end) , rrelcm(1,2:end) )

        % closest time
        [ Closest , index ] = min( rbarcm( 2:end ) ) ;
        ClosestTime = t( index )/(3600) ;

        fprintf( 'The closest the satellites get is %f km after %f
hours \n' , Closest , ClosestTime )
        fprintf( 'I used Vallado''s Universal Variable code because
mine was \n' )
        fprintf( 'only working if I set the time step absurdly small
and let \n' )
        fprintf( 'the code run for a long time. I am going to try to
fix that \n' )
        fprintf( 'but I didn''t figure it out for this homework. \n' )
    end

    function HW1_P2(mu,denomS,denomC)
        rearth = 6378 ;
        zA = 300 ;
        zB = 250 ;

        rA = [ zA+rearth ; 0 ; 0 ] ;
        rB = [ 0 ; zB+rearth ; 0 ] ;

        vA = [ 0 ; sqrt( mu / ( zA + rearth ) ) ; 0 ] ;
        vB = [ -sqrt( mu / ( zB + rearth ) ) ; 0 ; 0 ] ;

        rrel = rB - rA ;
            omega = cross( rA , vA ) / norm( rA )^2 ;
        vrel = vB - vA - cross( omega , rrel ) ;
            omegadot = -2*dot( vA , rA )*omega / norm( rA )^2 ;
            aA = -mu*rA / norm( rA )^3 ;
            aB = -mu*rB / norm( rB )^3 ;
        arel = aB - aA - cross( omegadot , rrel ) - cross( omega ,
cross( omega , rrel ) ) - 2*cross( omega , vrel ) ;

        fprintf( 'When A is over the equator and B is over the north
pole \n' )
        fprintf( 'The position of B relative to A is: \n' )
        fprintf( '%f km \n' , rrel )
        fprintf( 'The velocity of B relative to A is: \n' )
```

```
        fprintf( '%f km/s \n' , vrel )
        fprintf( 'The acceleration of B relative to A is: \n' )
        fprintf( '%e km/s^2 \n' , arel )

    end

    function HW1_P3(mu,denomC,denomS)
        % set up satellite A
        zpA = 250 ;
        eccA = 0.1 ;
        hA = sqrt( ( zpA + 6378 )*mu*( 1 + eccA ) ) ;
        incA = 51 ;
        raanA = 0 ;
        omegaA = 0 ;
        thetaA = 0 ;
        [ rA , vA ] = coes2state(  hA , eccA , thetaA , raanA ,
omegaA , incA , mu ) ; % state of A in ECI
        Period = (2*pi/mu^2)*(hA/(sqrt(1-eccA^2)))^3 ;

        % B relative to A
        dr0 = [ -1 ; -1 ; 0 ] ;
        dv0 = [ 0 ; 2 ; 0 ]*1e-3 ;

        % propagate forward using linearization
        tspan = [ 0 Period ]*10 ;
        state0 = [ rA ; vA ; dr0 ; dv0 ] ;
        opts = odeset( 'RelTol' , 1e-8 , 'AbsTol' , 1e-8 ) ;
        [ time , state ] = ode45( @LinearizedRendevous , tspan ,
state0 , opts , mu ) ;

        % 2d plot of relative position (ignores z because it is small)
        figure
        plot( state(:,8) , state(:,7) )
        ylabel( 'R-Bar [km]' )
        xlabel( 'V-Bar [km]' )

        % distance from relative position vector
        tsteps = length( time ) ;
        Distance = zeros( tsteps , 1 ) ;
        for ii = 1:tsteps
            Distance( ii ) = norm( state( ii , 7:9 ) ) ;
        end

        figure
        plot( time/3600 , Distance )
        xlabel( 'Time [Hrs]' )
        ylabel( 'Total Seperation Distance [km]' )

        % find closest distance
        [ Closest , index ] = min( Distance( 1:end ) ) ;
        ClosestTime = time( index )/(3600) ;

        % display answer
```

```
        fprintf( 'After 10 periods the distance between the two
satellites is %f km. \n' , Distance( end ) )
        fprintf( 'The closest that they get is %f km after %f hours
\n' , Closest , ClosestTime )

        fprintf( 'These plots look correct as the distance oscillates
in the \n' )
        fprintf( 'R-bar direction and grows in the V-bar direction.
There are \n' )
        fprintf( 'also 10 cycles of growing oscillations which makes
sense for \n' )
        fprintf( '10 orbital periods \n' )
   end

   function HW1_P4(mu,denomC,denomS)
        period = 90*60 ;
        n = 2*pi / period ;
        t = 15*60 ;
        dr0 = [ 1 ; 0 ; 0 ] ;
        dv0 = [ 0 ; 10 ; 0 ]*1e-3 ;
        [ dr , dv ] = CircularRendevous( t , n , dr0 , dv0 ) ;
        fprintf( 'After 15 minutes the satellite is %f km from the
space station \n' , norm( dr ) )
        fprintf( 'This seems reasonable seperation after a sixth of a
period \n' )
        fprintf( 'in a low orbit \n' )

   end
```

Functions

```
function [ dstate , dt ] = LinearizedRendevous( t , state , mu )

rA = state(1:3) ;
vA = state(4:6) ;
dr = state(7:9) ;
dv = state(10:12) ;

R = norm( rA ) ;
h = cross( rA , vA ) ;
hmag = norm( h ) ;
dstate = zeros( 12 , 1 ) ;

dstate(1:3) = vA ;
dstate(4:6) = -mu*rA / R^3;

dstate(7:9) = dv ;
dstate(10) = ( ( 2*mu/R^3 ) + ( hmag^2/R^4 ) )*dr(1) - ( 2*dot( vA ,
 rA )*hmag/R^4 )*dr(2) + 2*( hmag/R^2 )*dv(2) ;
dstate(11) = ( ( hmag^2/R^4 ) - ( mu/R^3 ) )*dr(2) + ( 2*dot( vA ,
 rA )*hmag/R^4 )*dr(1) - 2*( hmag/R^2 )*dv(1) ;
dstate(12) = -( mu/R^3 )*dr(3) ;
```

```
end

   %
 ------------------------------------------------------------------------------
   %
   %                              function kepler
   %
   %  this function solves keplers problem for orbit determination
 and returns a
   %    future geocentric equatorial (ijk) position and velocity
 vector.  the
   %    solution uses universal variables.
   %
   %  author        : david vallado                  719-573-2600
 22 jun 2002
   %
   %  revisions
   %    vallado     - fix some mistakes
 13 apr 2004
   %
   %  inputs          description                    range / units
   %    ro          - ijk position vector - initial  km
   %    vo          - ijk velocity vector - initial  km / s
   %    dtsec       - length of time to propagate    s
   %
   %  outputs       :
   %    r           - ijk position vector            km
   %    v           - ijk velocity vector            km / s
   %    error       - error flag                     'ok', ...
   %
   %  locals        :
   %    f           - f expression
   %    g           - g expression
   %    fdot        - f dot expression
   %    gdot        - g dot expression
   %    xold        - old universal variable x
   %    xoldsqrd    - xold squared
   %    xnew        - new universal variable x
   %    xnewsqrd    - xnew squared
   %    znew        - new value of z
   %    c2new       - c2(psi) function
   %    c3new       - c3(psi) function
   %    dtsec       - change in time                 s
   %    timenew     - new time                       s
   %    rdotv       - result of ro dot vo
   %    a           - semi or axis                   km
   %    alpha       - reciprocol  1/a
   %    sme         - specific mech energy           km2 / s2
   %    period      - time period for satellite      s
   %    s           - variable for parabolic case
   %    w           - variable for parabolic case
   %    h           - angular momentum vector
   %    temp        - temporary real*8 value
   %    i           - index
```

```
    %
    %  coupling      :
    %    mag         - magnitude of a vector
    %    findc2c3    - find c2 and c3 functions
    %
    %  references    :
    %    vallado       2004, 95-103, alg 8, ex 2-4
    %
    % [r, v] =  kepler  ( ro, vo, dtsec );
    %
 ------------------------------------------------------------------------------

    function [r, v] =  kepler  ( ro, vo, dtseco )
    %function [r,v,errork] =  kepler  ( ro,vo, dtseco, fid );

    % -------------------------  implementation   -----------------
    % set constants and intermediate printouts
mu = 398600.4418 ;
    numiter    =    50;
small = 1e-10 ;
twopi = 2*pi ;


    % --------------------  initialize values   -------------------
    znew  = 0.0;
    dtsec = dtseco;

    if ( abs( dtseco ) > small )
        magro = mag( ro );
        magvo = mag( vo );
        rdotv= dot( ro,vo );

        % -------------  find sme, alpha, and a  ------------------
        sme= ( (magvo^2)*0.5  ) - ( mu /magro );
        alpha= -sme*2.0/mu;

        if ( abs( sme ) > small )
            a= -mu / ( 2.0 *sme );
        else
            a= infinite;
        end
        if ( abs( alpha ) < small )   % parabola
            alpha= 0.0;
        end


        % ------------   setup initial guess for x  ---------------
        % -----------------  circle and ellipse -------------------
        if ( alpha >= small )
            period= twopi * sqrt( abs(a)^3.0/mu  );
            % ------- next if needed for 2body multi-rev ----------
            if ( abs( dtseco ) > abs( period ) )
                % including the truncation will produce vertical lines
 that are parallel
```

```
            % (plotting chi vs time)
            dtsec = rem( dtseco, period );
        end
        xold = sqrt(mu)*dtsec * alpha;
    else
        % --------------------  parabola  ---------------------
        if ( abs( alpha ) < small )
            h = cross( ro,vo );
            magh = mag(h);
            p= magh*magh/mu;
            s= 0.5  * (halfpi - atan( 3.0 *sqrt( mu / (p*p*p) )*
dtsec ) );
            w= atan( tan( s )^(1.0 /3.0 ) );
            xold = sqrt(p) * ( 2.0 *cot(2.0 *w) );
            alpha= 0.0;
        else
            % ------------------  hyperbola  ------------------
            temp= -2.0 * mu * dtsec / ...
                ( a*( rdotv + sign(dtsec)*sqrt(-mu*a)* ...
                (1.0 -magro*alpha) ) );
            xold= sign(dtsec) * sqrt(-a) *log(temp);
        end
    end

    ktr= 1;
    dtnew = -10.0;
    % conv for dtsec to x units
    tmp = 1.0 / sqrt(mu);

    while ((abs(dtnew*tmp - dtsec) >= small) && (ktr < numiter))
        xoldsqrd = xold*xold;
        znew     = xoldsqrd * alpha;

        % ------------- find c2 and c3 functions --------------
        [c2new, c3new] = findc2c3( znew );

        % ------- use a newton iteration for new values -------
        rval = xoldsqrd*c2new + rdotv*tmp *xold*(1.0 -znew*c3new)
+ ...
            magro*( 1.0  - znew*c2new );
        dtnew= xoldsqrd*xold*c3new + rdotv*tmp*xoldsqrd*c2new
+ ...
            magro*xold*( 1.0  - znew*c3new );

        % ------------- calculate new value for x -------------
        temp1 = ( dtsec*sqrt(mu) - dtnew ) / rval;
        xnew = xold + temp1;

        % ----- check if the univ param goes negative. if so, use
bissection
        if xnew < 0.0
            xnew = xold*0.5;
        end
```

```
            ktr = ktr + 1;
            xold = xnew;
        end

        if ( ktr >= numiter )
            errork= 'knotconv';
            fprintf(1,'kep not conv in %2i iter %11.3f \n',numiter,
dtseco );
            for i= 1 : 3
                v(i)= 0.0;
                r(i)= v(i);
            end
        else
            % --- find position and velocity vectors at new time --
            xnewsqrd = xnew*xnew;
            f = 1.0  - ( xnewsqrd*c2new / magro );
            g = dtsec - xnewsqrd*xnew*c3new/sqrt(mu);

            for i= 1 : 3
                r(i)= f*ro(i) + g*vo(i);
            end
            magr = mag( r );
            gdot = 1.0  - ( xnewsqrd*c2new / magr );
            fdot = ( sqrt(mu)*xnew / ( magro*magr ) ) *
( znew*c3new-1.0  );
            for i= 1 : 3
                v(i)= fdot*ro(i) + gdot*vo(i);
            end

            temp= f*gdot - fdot*g;
            if ( abs(temp-1.0 ) > 0.00001  )
                errork= 'fandg';
            end

        end  % if
    else
        % ----------- set vectors to incoming since 0 time --------
        for i=1:3
            r(i)= ro(i);
            v(i)= vo(i);
        end
    end
    end

%
 ------------------------------------------------------------------------------
%
%                           function findc2c3
%
%  this function calculates the c2 and c3 functions for use in the
 universal
%    variable calculation of z.
%
```

```
%  author        : david vallado                  719-573-2600   27
 may 2002
%
%  revisions
%                -
%
%  inputs          description                    range / units
%    znew        - z variable                     rad2
%
%  outputs       :
%    c2new       - c2 function value
%    c3new       - c3 function value
%
%  locals        :
%    sqrtz       - square root of znew
%
%  coupling      :
%    sinh        - hyperbolic sine
%    cosh        - hyperbolic cosine
%
%  references    :
%    vallado       2001, 70-71, alg 1
%
% [c2new,c3new] = findc2c3 ( znew );
%
 ------------------------------------------------------------------------------

function [c2new,c3new] = findc2c3 ( znew )

        small =     0.000001;

        % -------------------------  implementation
 -----------------
        if ( znew > small )
            sqrtz = sqrt( znew );
            c2new = (1.0 -cos( sqrtz )) / znew;
            c3new = (sqrtz-sin( sqrtz )) / ( sqrtz^3 );
        else
            if ( znew < -small )
                sqrtz = sqrt( -znew );
                c2new = (1.0 -cosh( sqrtz )) / znew;
                c3new = (sinh( sqrtz ) - sqrtz) / ( sqrtz^3 );
            else
                c2new = 0.5;
                c3new = 1.0 /6.0;
            end
        end

end

end
```

Published with MATLAB® R2019a