Table of Contents


```
function Aero_557_HW4()
```

557 HW4

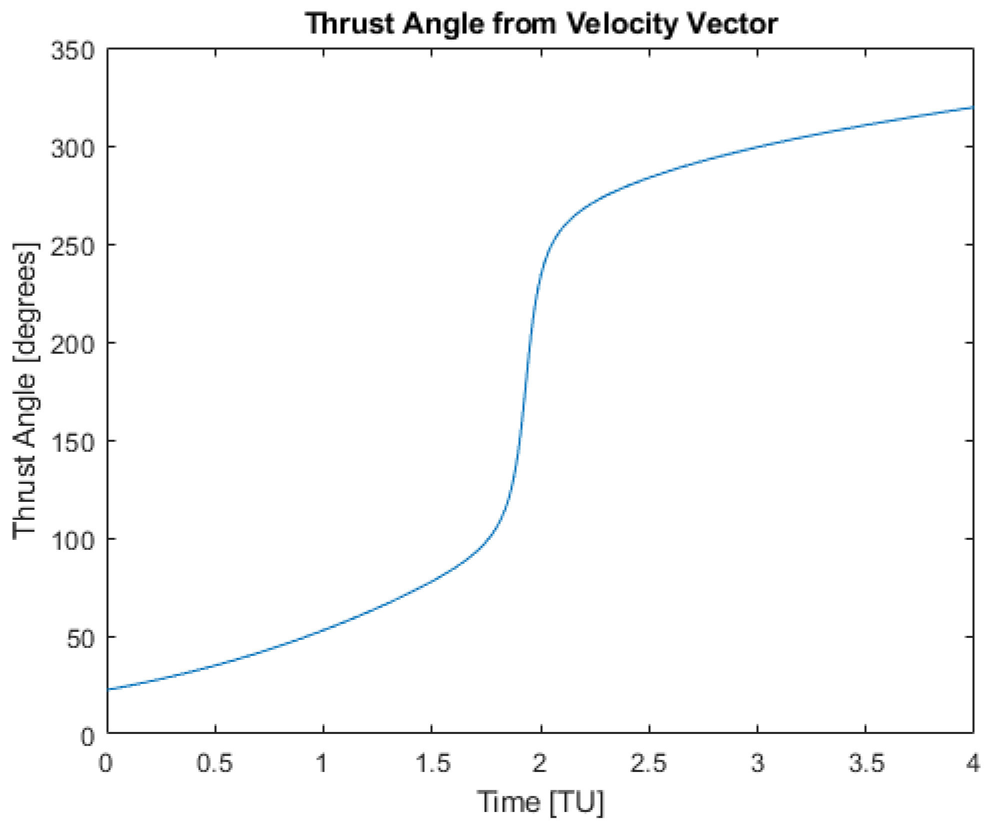
Liam Hood

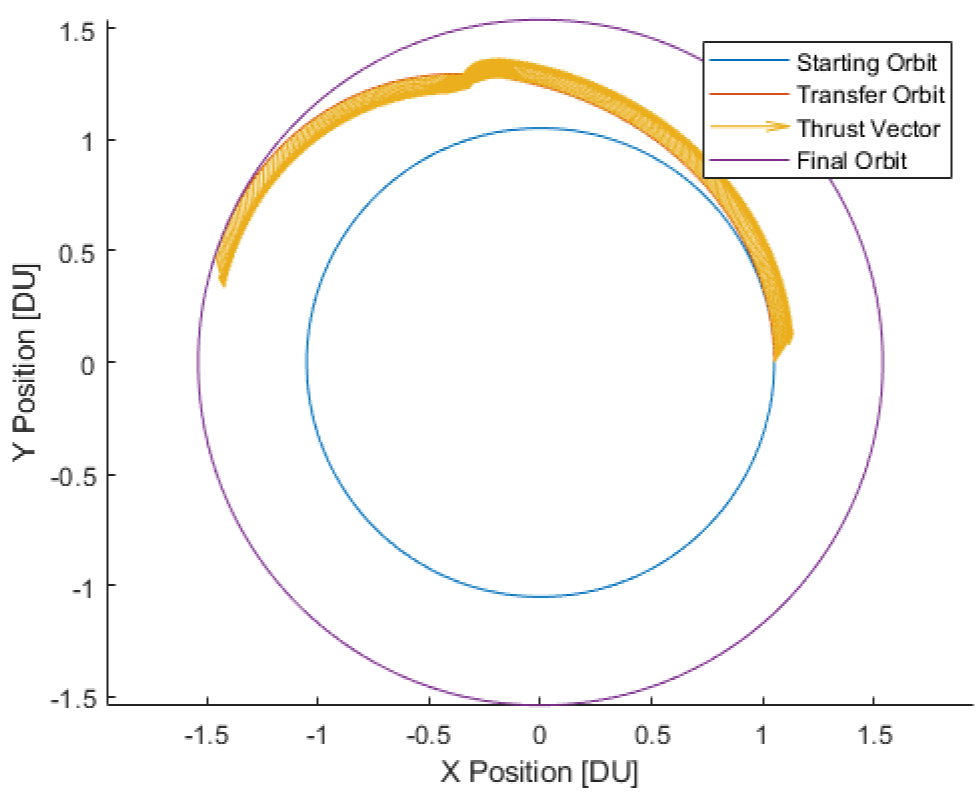
```
clear ; close all ; clc ;
```

1

```
fprintf( 'Problem 1 \n' )
HW4P1()
```

```
Problem 1
The starting r is 1.050000 DU
The starting theta is 0.000000 radians
The starting rdot is 0.000000 DU
The starting thetadot is 0.929429 radians
The starting lambda 1 is -4.186725
The starting lambda 2 is -0.000000
The starting lambda 3 is -0.980374
The starting lambda 4 is -2.499677

The final r is 1.537579 DU
The final theta is 2.823038 radians
The final rdot is -0.000000 DU
The final thetadot is 0.524498 radians
The final lambda 1 is -2.542377
The final lambda 2 is -0.000000
The final lambda 3 is 1.668906
The final lambda 4 is -3.014343
The cost function is -r to maximize altitude and is -1.537579 DU
```

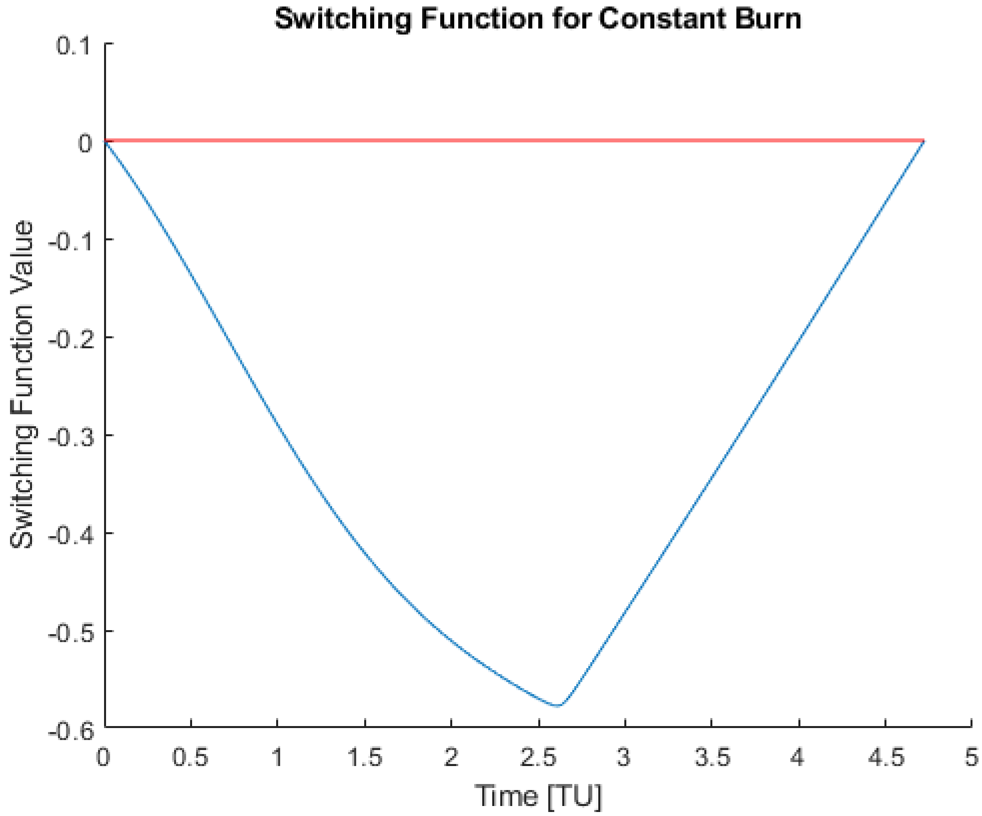

Thrust Angle from Velocity Vector
Thrust Angle [degrees]
Time [TU]
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4

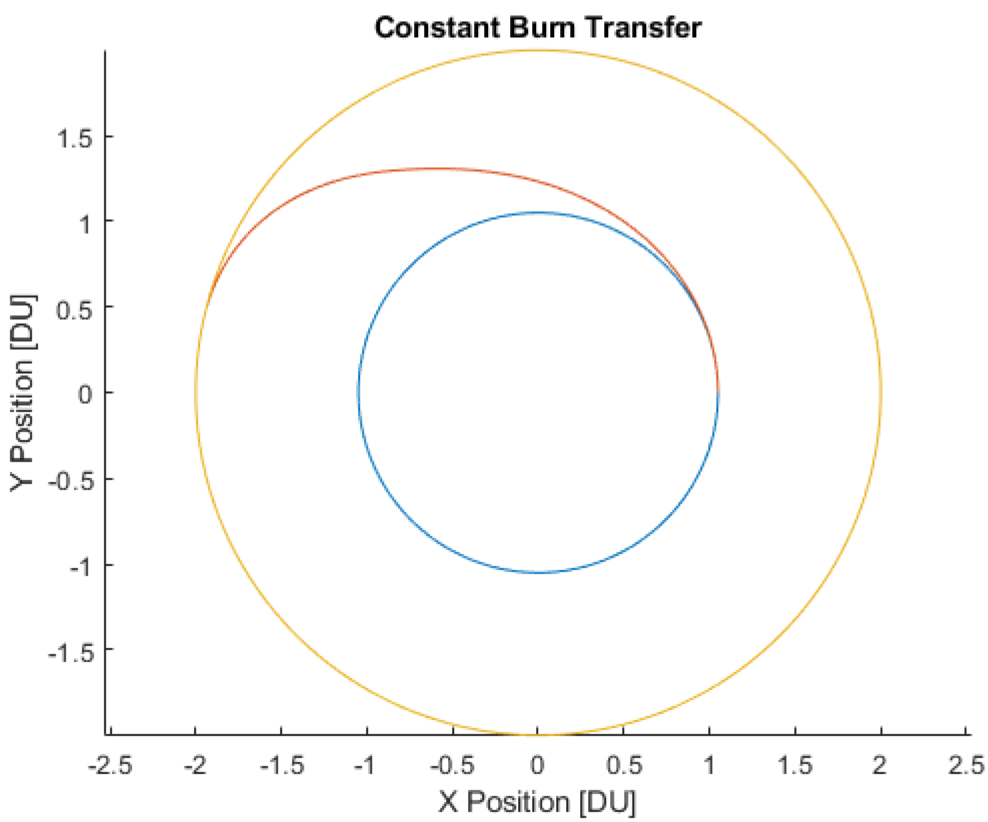

Starting Orbit
Transfer Orbit
Thrust Vector
Final Orbit
Y Position [DU]
X Position [DU]
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5

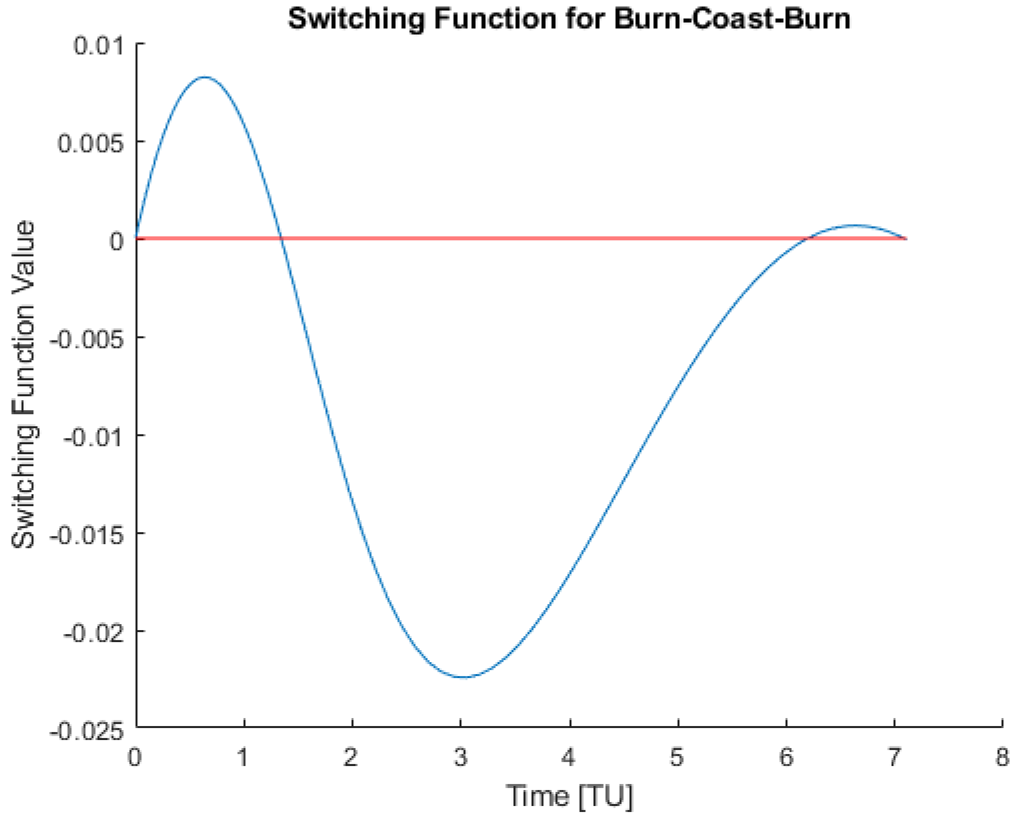
2

```
fprintf( '\n\nProblem 2 \n' )
HW4P2()
```

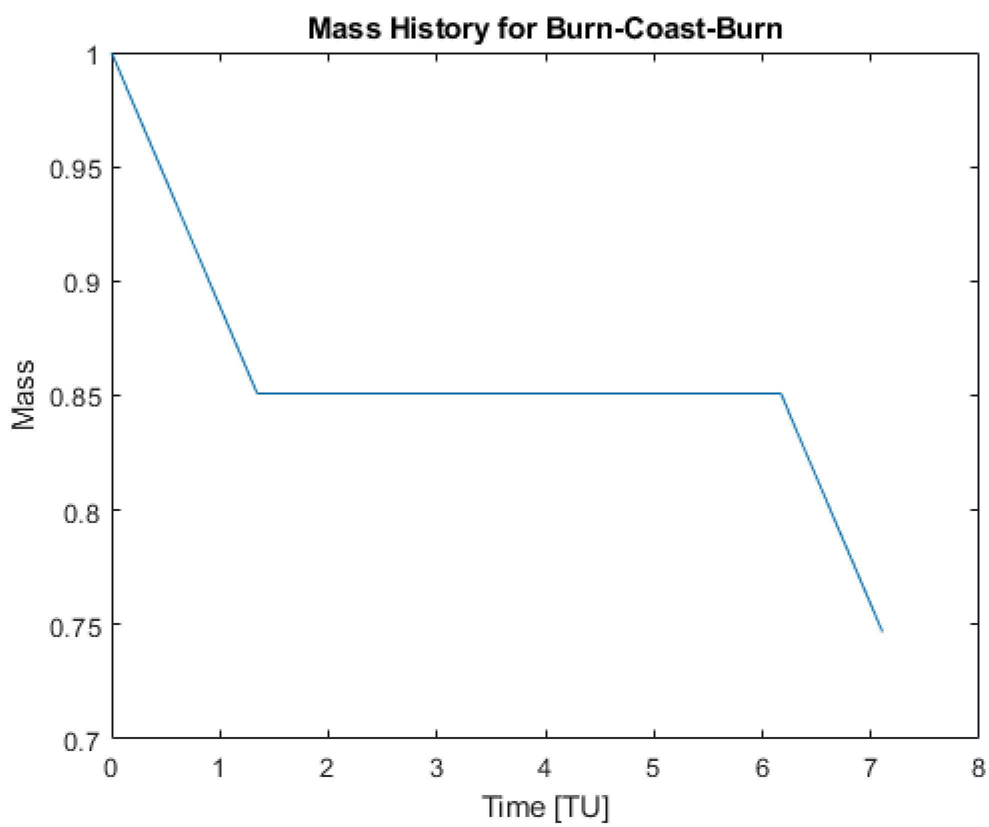
```
Problem 2
The mass used for constant thrust is 0.524896
The mass used for burn-coast-burn is 0.253110
The first burn ends at 1.342483 TU
The second burn starts at 6.169864 TU
The transfer ends at 7.105374 TU
The starting x is 1.050000 DU
The starting y is 0.000000 DU
The starting xdot is 0.000000 DU/TU
The starting ydot is 0.975900 DU/TU
The starting mass is 1.000000 Mass/Original Mass
The starting lambda 1 is -0.750044
The starting lambda 2 is 0.014272
The starting lambda 3 is 0.015355
The starting lambda 4 is -0.828701
The starting lambda 5 is -0.745959

The final x is -1.397187 DU
The final y is -1.431070 DU
The final xdot is 0.505940 DU/TU
The final ydot is -0.493989 DU/TU
The final mass is 0.746890 Mass/Original Mass
The final lambda 1 is 0.196383
The final lambda 2 is 0.203592
The final lambda 3 is -0.590337
The final lambda 4 is 0.583134
The final lambda 5 is -0.999987
The cost function is -massFinal to maximize mass and is -0.746890
```


Switching Function for Constant Burn
Switching Function Value
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time [TU]

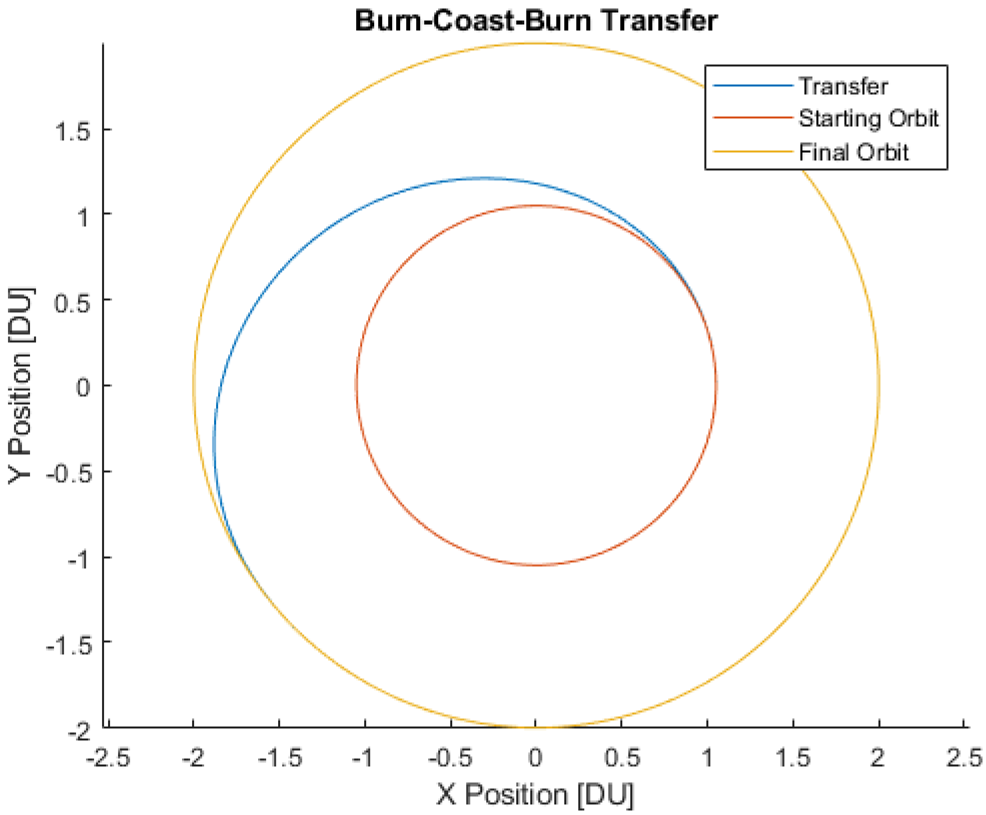

Constant Burn Transfer
Y Position [DU]
1.5
1
0.5
0
-0.5
-1
-1.5
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
X Position [DU]



Switching Function for Burn-Coast-Burn
Switching Function Value
0.01
0.005
0
-0.005
-0.01
-0.015
-0.02
-0.025
0
1
2
3
4
5
6
7
8
Time [TU]



Mass History for Burn-Coast-Burn
Mass
1
0.95
0.9
0.85
0.8
0.75
0.7
0
1
2
3
4
5
6
7
8
Time [TU]



Work

```
function HW4P1()
    theta = 0 ;
    ecc = 0 ;
    a = 1.05 ;

    r = a*( 1 - ecc^2 )/( 1 + ecc*cos( theta ) ) ;
    rdot = ( ecc*sin( theta ) )/( sqrt( a )*sqrt( 1 - ecc^2 ) ) ;
    thetadot = ( 1 + ecc*cos( theta ) )/( a*( 1 - ecc^2 ) )^1.5 ;
    lambda = [ -4 ; 0 ; -1 ; -2.5 ] ;
    s0 = [ r ; theta ; rdot ; thetadot ; lambda ] ;
    opts = optimoptions( 'fsolve' , 'Display' , 'off'
  , 'FunctionTolerance' , 1e-8 , 'StepTolerance' ,
 1e-8 , 'OptimalityTolerance' , 1e-8 , 'Algorithm' , 'levenberg-
marquardt' ) ; %
    [ s0s , F ] = fsolve( @PolarSolveFun , s0 , opts ) ;
    optsode = odeset( 'RelTol' , 1e-8 , 'AbsTol' , 1e-8 ) ;
    [ t , s ] = ode45( @PolarEOM , [ 0 , 4 ] , s0s , optsode , .1 ) ;
    [ ts , ss ] = ode45( @PolarEOM , [ 0 , 10 ] , s0s , optsode ,
 0 ) ;
    [ te , se ] = ode45( @PolarEOM , [ 0 , 20 ] , s(end,:) , optsode ,
 0 ) ;
    for ii = 1:length( t )
```

```
        cosc(ii) = ( -s(ii,8)/( s(ii,1)^2*s(ii,7)^2 +
s(ii,8)^2 )^0.5 ) ;
        sinc(ii) = ( -( s(ii,7)*s(ii,1) )/( s(ii,1)^2*s(ii,7)^2 +
s(ii,8)^2 )^0.5 ) ;
        x(ii) = s(ii,1)*cos(s(ii,2)) ;
        y(ii) = s(ii,1)*sin(s(ii,2)) ;
        c(ii) = atan2d( sinc(ii) , cosc(ii) ) ;
        if c(ii) < 0
            c(ii) = 360 + c(ii) ;
        end
        vxmag = s(ii,3)*cos(s(ii,2)) - s(ii,1)*s(ii,4)*sin(s(ii,2)) ;
        vymag = s(ii,3)*sin(s(ii,2)) + s(ii,1)*s(ii,4)*cos(s(ii,2)) ;
        vmag = sqrt( vxmag^2 + vymag^2 ) ;
        vx(ii) = vxmag/vmag ;
        vy(ii) = vymag/vmag ;
        steer1(ii) = cosc(ii) + vx(ii) ;
        steer2(ii) = sinc(ii) + vy(ii) ;
    end
    for ii = 1:length( ts )
        xs(ii) = ss(ii,1)*cos(ss(ii,2)) ;
        ys(ii) = ss(ii,1)*sin(ss(ii,2)) ;
    end
    for ii = 1:length( te )
        xe(ii) = se(ii,1)*cos(se(ii,2)) ;
        ye(ii) = se(ii,1)*sin(se(ii,2)) ;
    end

    figure
    plot( t , c )
    title( 'Thrust Angle from Velocity Vector' )
    xlabel( 'Time [TU]' )
    ylabel( 'Thrust Angle [degrees]' )

    figure
    axis equal
    hold on
    plot( xs , ys )
    plot( x , y )
    quiver( x , y , steer1 , steer2 , 'AutoScaleFactor' , 1 )
    plot( xe , ye )
    hold off
    xlabel( 'X Position [DU]' )
    ylabel( 'Y Position [DU]' )
    legend( 'Starting Orbit' , 'Transfer Orbit' , 'Thrust Vector'
, 'Final Orbit' )

    stateLabels = [ "r" ; "theta" ; "rdot" ; "thetadot" ] ;
    stateUnits = [ "DU" ; "radians" ; "DU" ; "radians" ] ;
    for ii = 1:4
        fprintf( 'The starting %s is %f %s \n' , stateLabels(ii) ,
s0(ii) , stateUnits(ii) )
    end
    for ii = 1:4
        fprintf( 'The starting lambda %i is %f \n' , ii , s(1,ii+4) )
```

```
    end
    fprintf( '\n' )
    for ii = 1:4
        fprintf( 'The final %s is %f %s \n' , stateLabels(ii) ,
 s(end,ii) , stateUnits(ii) )
    end
    for ii = 1:4
        fprintf( 'The final lambda %i is %f \n' , ii , s(end,ii+4) )
    end
    fprintf( 'The cost function is -r to maximize altitude and is %f
 DU \n' , -s(end,1) )


end

function HW4P2()
    T = .1 ;
    ve = .9 ;
    x = 1.05 ;
    y = 0 ;
    xdot = 0 ;
    ydot = sqrt( 1/x ) ;
    mass = 1 ;
    lambda = [ -2 ; -1 ; -1 ; -2 ; -1 ] ;
    lambda =
 [-0.546630073119638,-0.147919753839651,-0.159151280132354,-0.771166552733118,-0.7
    sto0 = [ 2 ; 0 ; 0 ; 1/sqrt(2) ; 1 ; lambda ] ;
    tg = 6 ;
    tol = 1e-10 ;
    guess = [ lambda ; tg ] ;
    opts = optimoptions( 'fsolve' , 'Display' , 'off'
 , 'FunctionTolerance' , tol ,  ...
        'OptimalityTolerance' , tol , 'MaxIterations' ,
 1e6 , 'MaxFunctionEvaluations' , 1e6 , 'Algorithm' , 'levenberg-
marquardt' ) ; %
    optsode = odeset( 'RelTol' , 1e-8 , 'AbsTol' , 1e-8 ) ;
    [ guess , F ] = fsolve( @BCBfpFun , guess , opts ) ;
    s0fp = [ x ; y ; xdot ; ydot ; mass ; guess(1:5) ] ;
    [ tfp , sfp ] = ode45( @BCBEOM , [ 0 , guess(6) ] , s0fp ,
 optsode , T , ve ) ;
        [ tso , sso ] = ode45( @BCBEOM , [ 0 , 10 ] , s0fp , optsode ,
 0 , ve ) ;
        [ teo , seo ] = ode45( @BCBEOM , [ 0 , 30 ] , sfp(end,1:10) ,
 optsode , 0 , ve ) ;
    lamfp = sfp(:,6:10) ;
    for ii = 1:length( tfp )
        lamv = sqrt( lamfp(ii,3)^2 + lamfp(ii,4)^2 ) ;
        switchingfp(ii) = (lamv)*ve + lamfp(ii,5)*sfp(ii,5) ;
    end
    figure
    hold on
    plot( tfp , switchingfp )
    plot( [ 0 , max(tfp) ] , [ 0 , 0 ] , '-r' )
    hold off
```

```
    title( 'Switching Function for Constant Burn' )
    xlabel( 'Time [TU]' )
    ylabel( 'Switching Function Value' )

    figure
    axis equal
    hold on
    plot( sso(:,1) , sso(:,2) )
    plot( sfp(:,1) , sfp(:,2) )
    plot( seo(:,1) , seo(:,2) )
    title( 'Constant Burn Transfer' )
    xlabel( 'X Position [DU]' )
    ylabel( 'Y Position [DU]' )
    hold off

    lambda = [ -2 ; -3 ; -2 ; -2 ; -1 ] ;
%     sto0 = [ 2 ; 0 ; 0 ; 1/sqrt(2) ; 1 ; lambda ] ;
    t1 = 2.6 ;
    t2 = 4.7 - t1 ;
    tf = 4.7*1.2 - t2 - t1 ;
    guess = [ lambda ; t1 ; t2 ; tf ] ;
    [ guess , F ] = fsolve( @BCBSolveFun , guess , opts ) ;
    s0 = [ x ; y ; xdot ; ydot ; mass ; guess(1:5) ] ;
    t1 = abs( guess(6) ) ;
    t2 = abs( guess(7) ) + t1 ;
    tf = abs( guess(8) ) + t2 ;
    [ tb1 , sb1 ] = ode45( @BCBEOM , [ 0 , t1 ] , s0(1:10) , opts ,
 T , ve ) ;
    [ tc , sc ] = ode45( @BCBEOM , [ t1 , t2 ] , sb1(end,:) , opts ,
 0 , ve ) ;
    [ tb2 , sb2 ] = ode45( @BCBEOM , [ t2 , tf ] , sc(end,:) , opts ,
 T , ve ) ;
    t = [ tb1 ; tc ; tb2 ] ;
    s = [ sb1 ; sc ; sb2 ] ;
    lam = s(:,6:10) ;
        [ tso , sso ] = ode45( @BCBEOM , [ 0 , 10 ] , s0(1:10) ,
 optsode , 0 , ve ) ;
        [ teo , seo ] = ode45( @BCBEOM , [ 0 , 30 ] , s(end,1:10) ,
 optsode , 0 , ve ) ;
    for ii = 1:length( t )
        lamv = sqrt( lam(ii,3)^2 + lam(ii,4)^2 ) ;
        switching(ii) = lamv*ve + lam(ii,5)*s(ii,5) ;
    end
    figure
    hold on
    plot( t , switching )
    plot( [ 0 , max(t) ] , [ 0 , 0 ] , '-r' )
    hold off
    title( 'Switching Function for Burn-Coast-Burn' )
    xlabel( 'Time [TU]' )
    ylabel( 'Switching Function Value' )

    figure
    plot( t , s(:,5) )
```

```
    title( 'Mass History for Burn-Coast-Burn' )
    xlabel( 'Time [TU]' )
    ylabel( 'Mass' )

    figure
    axis equal
    hold on
    plot( s(:,1) , s(:,2) )
    plot( sso(:,1) , sso(:,2) )
    plot( seo(:,1) , seo(:,2) )
    legend( 'Transfer' , 'Starting Orbit' , 'Final Orbit' )
    title( 'Burn-Coast-Burn Transfer' )
    xlabel( 'X Position [DU]' )
    ylabel( 'Y Position [DU]' )

    fprintf( 'The mass used for constant thrust is %f \n' , 1 -
 sfp(end,5) )
    fprintf( 'The mass used for burn-coast-burn is %f \n' , 1 -
 s(end,5) )

    stateLabels = [ "x" ; "y" ; "xdot" ; "ydot" ; "mass"] ;
    stateUnits = [ "DU" ; "DU" ; "DU/TU" ; "DU/TU" ; "Mass/Original
 Mass" ] ;
    fprintf( 'The first burn ends at %f TU \n' , t1 )
    fprintf( 'The second burn starts at %f TU \n' , t2 )
    fprintf( 'The transfer ends at %f TU \n' , tf )
    for ii = 1:5
        fprintf( 'The starting %s is %f %s \n' , stateLabels(ii) ,
 s0(ii) , stateUnits(ii) )
    end
    for ii = 1:5
        fprintf( 'The starting lambda %i is %f \n' , ii , guess(ii) )
    end
    fprintf( '\n' )
    for ii = 1:5
        fprintf( 'The final %s is %f %s \n' , stateLabels(ii) ,
 s(end,ii) , stateUnits(ii) )
    end
    for ii = 1:5
        fprintf( 'The final lambda %i is %f \n' , ii , s(end,ii+5) )
    end
    fprintf( 'The cost function is -massFinal to maximize mass and is
 %f \n' , -s(end,5) )

end
```

Functions

```
    function F = BCBfpFun( guess )
        T = .1 ;
        ve = .9 ;
        lam0 = guess(1:5) ;
        tf = abs( guess(end) ) ;
```

```
            x = 1.05 ;
            y = 0 ;
            xdot = 0 ;
            ydot = sqrt( 1/x ) ;
            mass = 1 ;
        s0 = [ x ; y ; xdot ; ydot ; mass ] ;

        optsode = odeset( 'RelTol' , 1e-8 , 'AbsTol' , 1e-8 ) ;
        [ t , s ] = ode45( @BCBEOM , [ 0 , tf ] , [ s0 ; lam0 ] ,
 optsode , T , ve ) ;

        lamf = s(end,6:10) ;
        sf = s(end,1:5) ;


        % omega
        rf = sqrt( sf(1)^2 + sf(2)^2 ) ;
        F(1,1) = ( sf(1)^2 + sf(2)^2 - 4 ) ;
        F(2,1) = ( sf(3)^2 + sf(4)^2 - 1/2 ) ;
        F(3,1) = ( sf(4)*sf(1) - sf(3)*sf(2) - 2/sqrt(2) ) ;

        % lambda final
        F(4,1) = lamf(5) + 1 ;

        % Switching
            lam0v = sqrt( lam0(3)^2 + lam0(4)^2 ) ;
        F(5,1) = ((lam0v)*ve + lam0(5)*s0(5)) ;
            lamfv = sqrt( lamf(3)^2 + lamf(4)^2 ) ;
        F(6,1) = ( (lamfv)*ve + lamf(5)*sf(5) ) ;

        % dependency
        F(7,1) = -lamf(1)*( sf(2)/sf(1) ) + lamf(2) - lamf(3)*( sf(4)/
sf(3) )*...
            ( ( sf(3) + ( sf(2)/sf(1) )*sf(4) )/( sf(1) + ( sf(4)/
sf(3) )*sf(2) ) )...
            + lamf(4)*( ( sf(3) + ( sf(2)/sf(1) )*sf(4) )/( sf(1) +
 ( sf(4)/sf(3) )*sf(2) ) ) ;

        % Hf
        lamv = sqrt( lamf(3)^2 + lamf(4)^2 ) ;
        c1 = -lamf(3)/lamv ;
        c2 = -lamf(4)/lamv ;
        F(8,1) = lamf(1)*sf(3) + lamf(2)*sf(4) + lamf(3)*( -sf(1)/
rf^3 ) ...
            + lamf(4)*( -sf(2)/rf^3 ) ;
    end

    function F = BCBSolveFun( guess )
        T = .1 ;
        ve = .9 ;
        t = guess(6:8) ;
        t1 = abs( t(1) ) ;
        t2 = abs( t(2) ) + t1 ;
        tf = abs( t(3) ) + t2 ;
```

```
        lam0 = guess( 1:5 ) ;
            x = 1.05 ;
            y = 0 ;
            xdot = 0 ;
            ydot = sqrt( 1/x ) ;
            mass = 1 ;
        s0 = [ x ; y ; xdot ; ydot ; mass ] ;

        opts = odeset( 'RelTol' , 1e-8 , 'AbsTol' , 1e-8 ) ;
        [ tb1 , sb1 ] = ode45( @BCBEOM , [ 0 , t1 ] , [ s0 ; lam0 ] ,
 opts , T , ve ) ;
        [ tc , sc ] = ode45( @BCBEOM , [ t1 , t2 ] , sb1(end,:) ,
 opts , 0 , ve ) ;
        [ tb2 , sb2 ] = ode45( @BCBEOM , [ t2 , tf ] , sc(end,:) ,
 opts , T , ve ) ;
        t = [ tb1 ; tc ; tb2 ] ;
        s = [ sb1 ; sc ; sb2 ] ;
        lamb = sb1(end,6:10) ;
        lamc = sc(end,6:10) ;
        lamf = sb2(end,6:10) ;
        sb = sb1(end,1:5) ;
        sc = sc(end,1:5) ;
        sf = sb2(end,1:5) ;

        % omega
        rf = sqrt( sf(1)^2 + sf(2)^2 ) ;
        vf = sqrt( 1/rf ) ;
        F(1,1) = ( sf(1)^2 + sf(2)^2 - 4 );
        F(2,1) = ( sf(3)^2 + sf(4)^2 - 1/2  ) ;
        F(3,1) = ( sf(4)*sf(1) - sf(3)*sf(2) - 2/sqrt(2) ) ;

        % lambda final
        F(4,1) = lamf(5) + 1;

        % Switching
            lam0v = sqrt( lam0(3)^2 + lam0(4)^2 ) ;
        F(5,1) = (lam0v*ve + lam0(5)*s0(5)) ;
            lambv = sqrt( lamb(3)^2 + lamb(4)^2 ) ;
        F(6,1) = (lambv*ve + lamb(5)*sb(5)) ;
            lamcv = sqrt( lamc(3)^2 + lamc(4)^2 ) ;
        F(7,1) = (lamcv*ve + lamc(5)*sc(5)) ;
            lamfv = sqrt( lamf(3)^2 + lamf(4)^2 ) ;
        F(8,1) = (lamfv*ve + lamf(5)*sf(5)) ;

        % dependency
        F(9,1) = -lamf(1)*( sf(2)/sf(1) ) + lamf(2) - lamf(3)*( sf(4)/
sf(3) )*...
            ( ( sf(3) + ( sf(2)/sf(1) )*sf(4) )/( sf(1) + ( sf(4)/
sf(3) )*sf(2) ) )...
            + lamf(4)*( ( sf(3) + ( sf(2)/sf(1) )*sf(4) )/( sf(1) +
 ( sf(4)/sf(3) )*sf(2) ) ) ;

        % Hf
        rf = sqrt( sf(1)^2 + sf(2)^2 ) ;
```

```
        lamv = sqrt( lamf(3)^2 + lamf(4)^2 ) ;
        c1 = -lamf(3)/lamv ;
        c2 = -lamf(4)/lamv ;
        F(10,1) = lamf(1)*sf(3) + lamf(2)*sf(4) + lamf(3)*( -sf(1)/
rf^3 ) ...
            + lamf(4)*( -sf(2)/rf^3 ) ;
    end

    function ds = BCBEOM( t , s , T , ve )
        lam = s(6:10) ;
        lamv = sqrt( lam(3)^2 + lam(4)^2 ) ;
        c1 = -lam(3)/lamv ;
        c2 = -lam(4)/lamv ;
        r = sqrt( s(1)^2 + s(2)^2 ) ;

        % f
        ds(1,1) = s(3) ;
        ds(2,1) = s(4) ;
        ds(3,1) = -s(1)/r^3 + (T/s(5))*c1 ;
        ds(4,1) = -s(2)/r^3 + (T/s(5))*c2 ;
        ds(5,1) = -T/ve ;

        % lambda dot
        ds(6,1) = lam(3)/r^3 - ( 3*s(1)/r^5 )*( lam(3)*s(1) +
 lam(4)*s(2) ) ;
        ds(7,1) = lam(4)/r^3 - ( 3*s(2)/r^5 )*( lam(4)*s(2) +
 lam(3)*s(1) ) ;
        ds(8,1) = -lam(1) ;
        ds(9,1) = -lam(2) ;
        ds(10,1) = ( T/s(5)^2 )*( lam(3)*c1 + lam(4)*c2 ) ;
    end

    function F = PolarSolveFun( s0 )
        tspan = [ 0 , 4 ] ;
        accel = .1 ;
        opts = odeset( 'RelTol' , 1e-8 , 'AbsTol' , 1e-8 ) ;
        [ t , s ] = ode45( @PolarEOM , tspan , s0 , opts , accel ) ;
        sf = s(end,:) ;

        % force intitial conditions
        F(1,1) = s0(1) - 1.05 ;
        F(2,1) = s0(2) ;
        F(3,1) = s0(3) ;
        F(4,1) = s0(4) - ( 1/( 1.05^1.5 ) ) ;

        % omega
        F(5,1) = sf(3) ;
        F(6,1) = sf(4)^2 * sf(1)^3 - 1 ;

        % lambda final
        F(7,1) = sf(5) - ( -1 + 1.5*sf(8)*sf(4)/sf(1) ) ;
        F(8,1) = sf(6) ;
        F(9,1) = s0(6) ;
```

```
    end

    function ds = PolarEOM( t , s , accel )
        accosc = accel*( -s(8)/( s(1)^2*s(7)^2 + s(8)^2 )^0.5 ) ;
        acsinc = accel*( -( s(7)*s(1) )/( s(1)^2*s(7)^2 +
 s(8)^2 )^0.5 ) ;
        % f
        ds(1,1) = s(3) ;
        ds(2,1) = s(4) ;
        ds(3,1) = s(1)*s(4)^2 - ( 1/s(1)^2 ) + acsinc ;
        ds(4,1) = ( accosc - 2*s(3)*s(4) )/s(1) ;
        % lambda dot
        ds(5,1) = -s(7)*s(4)^2 - 2*s(7)/s(1)^3 + ( s(8)/
s(1)^2 )*( accosc - 2*s(3)*s(4) ) ;
        ds(6,1) = 0 ;
        ds(7,1) = -s(5) + 2*s(8)*s(4)/s(1) ;
        ds(8,1) = -s(6) - 2*s(7)*s(1)*s(4) + 2*s(8)*s(3)/s(1) ;
    end

end
```

Published with MATLAB® R2019a