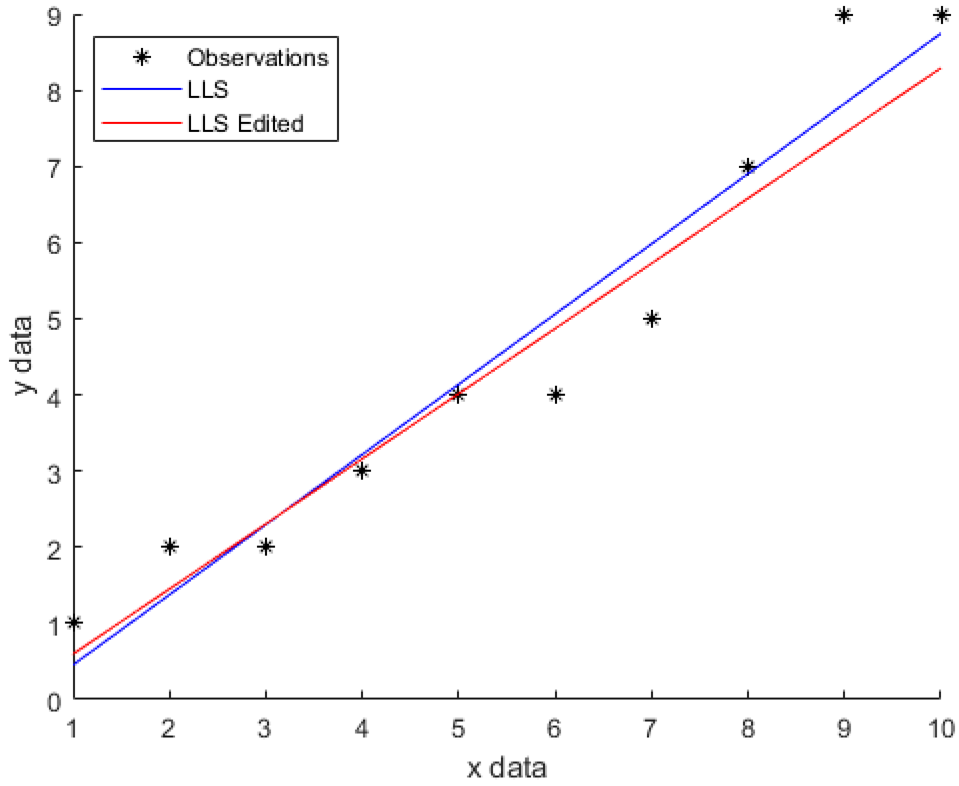
Table of Contents


```
% Homework 2
% Aero 557
% Liam Hood
clear ; close all ; clc ;
load( 'HW2P2.mat' )
load( 'HW2P3.mat' )
Aero_557_HW2fun(HW2P2,HW2P3)
function Aero_557_HW2fun(HW2P2,HW2P3)

pt = 'Problem number %u \n \n' ;
```

1

```
fprintf( pt , 1 )
% Editing Data w/ Linear Least Squares
HW1_P1()
fprintf( ' \n' )
```

```
Problem number 1

The RMS decreased from 0.662411 to 0.532122 by removing point 9
The standard deviation in alpha increases from 0.683130 to 0.700230
The standard deviation in beta increases from 0.110096 to 0.120483
Removing the bad data point decreases the RMS indicating
that the state estimate is more accurate but with so few
data points the this makes the confidence interval larger
```



2

```
fprintf( pt , 2 )
% Weighted Least Squares
[ r , v , AtWA , AtWb , UTC0 ] = HW1_P2(HW2P2) ;
fprintf( ' \n' )
```

```
Problem number 2

The initial position in km is
5748.9002       2679.361      3443.1514
The initial velocity in km/s is
4.3288     -1.9207      -5.726
The covariance matrix is
   0.073369   0.0055401  -0.0081731 -0.00086788   -3.584e-05
 4.2719e-05
  0.0055401    0.039499   -0.061079 -1.2568e-05 -0.00044205
 0.00065879
 -0.0081731   -0.061079     0.10614  4.7526e-05  0.00069277
 -0.0012168
-0.00086788 -1.2568e-05  4.7526e-05  1.4044e-05   -4.361e-07
 -9.3435e-07
 -3.584e-05 -0.00044205  0.00069277  -4.361e-07    6.009e-06
 -9.5517e-06
 4.2719e-05  0.00065879  -0.0012168 -9.3435e-07 -9.5517e-06
 1.9987e-05
```

3

```
fprintf( pt , 3 )
% Sequential Batch Filter
HW1_P3(HW2P3, r , v , AtWA , AtWb , UTC0)
fprintf( ' \n' )
```

```
Problem number 3

The initial position in km is
5748.7502      2679.5074        3442.97
The initial velocity in km/s is
4.332     -1.9225     -5.7249
The covariance matrix is
   0.034931  -0.0014551   0.0016451 -0.00021501   1.051e-05
 -4.4557e-05
 -0.0014551     0.01592   -0.024075  3.4323e-05 -0.00010375
 0.00011482
  0.0016451   -0.024075    0.044716 -4.0599e-05  0.00015086
 -0.00024293
-0.00021501  3.4323e-05 -4.0599e-05  1.7311e-06 -3.7031e-07
 4.0171e-07
  1.051e-05 -0.00010375  0.00015086 -3.7031e-07  7.7517e-07
 -7.5379e-07
-4.4557e-05  0.00011482 -0.00024293  4.0171e-07 -7.5379e-07
 1.933e-06
The change in initial position in m is
-150.0608      146.4039     -181.3796
The change in initial velocity in m/s is
3.1841     -1.7524      1.1074
The change in covariance matrix is
  -0.038438  -0.0069951   0.0098183  0.00065287   4.635e-05
 -8.7277e-05
 -0.0069951   -0.023579    0.037004  4.6891e-05   0.0003383
 -0.00054398
  0.0098183    0.037004   -0.061425 -8.8125e-05  -0.0005419
 0.00097391
 0.00065287  4.6891e-05 -8.8125e-05 -1.2313e-05   6.579e-08
 1.3361e-06
  4.635e-05   0.0003383  -0.0005419   6.579e-08 -5.2338e-06
 8.7979e-06
-8.7277e-05 -0.00054398  0.00097391  1.3361e-06  8.7979e-06
 -1.8054e-05
The covariance matrix improves with more observations.
The actual change in state is relatively small
```

Work

```
    function HW1_P1()
        xoi = [ 1 2 3 4 5 6 7 8 9 10 ]' ;
```

```
%         xoi = [ 1 2 3 4 5 6 7 8 ]' ;
        yoi = [ 1 2 2 3 4 4 5 7 9 9 ]' ;
%         yoi = [ 1 1 2 3 3 4 7 6 ]' ;
        n = length( xoi ) ;
        [ yc , rbar , RMS , P ] = LLS1( xoi , yoi ) ;
        std_a = sqrt( P(1,1) ) ;
        std_b = sqrt( P(2,2) ) ;
        Ibad = find( rbar > RMS ) ;
        jj = 0 ;
        for ii = 1:n
            if ii ~= Ibad
                jj = jj + 1 ;
                xom(jj) = xoi(ii) ;
                yom(jj) = yoi(ii) ;
            else
            end
        end
        [ ycm , rbarm , RMSm , Pm ] = LLS1( xom , yom ) ;
        std_am = sqrt( Pm(1,1) ) ;
        std_bm = sqrt( Pm(2,2) ) ;

        figure
        hold on
        plot( xoi , yoi , '*k' )
        plot( xoi , yc , 'b' )
        plot( xom , ycm , 'r' )
        hold off
        legend( 'Observations' , 'LLS' , 'LLS Edited' , 'Location'
, 'northwest' )
        xlabel( 'x data' )
        ylabel( 'y data' )


        fprintf( 'The RMS decreased from %f to %f by removing point %i
\n' , RMS , RMSm , Ibad )
        fprintf( 'The standard deviation in alpha increases from %f to
%f \n' , std_a , std_am )
        fprintf( 'The standard deviation in beta increases from %f to
%f \n' , std_b , std_bm )
        fprintf( 'Removing the bad data point decreases the RMS
indicating \n' )
        fprintf( 'that the state estimate is more accurate but with so
few \n' )
        fprintf( 'data points the this makes the confidence interval
larger \n' )
    end

    function [ r , v , AtWA , AtWb , UTC0 ] = HW1_P2(data)
        mu = 398600 ;
        d2r = pi/180 ;
        r2d = 180/pi ;

        lat = 21.5748 ;
        long = -158.2706 ;
```

```
        alt = 300.2 ;
        rhoerr = .0925 ;
        azerr = .0224*d2r ;
        elerr = .0139*d2r ;
        UTC = data(:,1:6) ;
        rho = data(:,9) ;
        az = data(:,7).*d2r ;
        el = data(:,8).*d2r ;
        obs = { rho , az , el } ;
        obserr = [ rhoerr , azerr , elerr ] ;
        [ r , v , RMS , P , AtWA , AtWb ] = DCODrazel2rv( obs ,
obserr , lat , long , alt , UTC ) ;
        Ppos = P(1:3,1:3) ;
        [ ev , lambda ] = eig( P ) ;
        lambda(1,1)*1e3 ;
        lambda(2,2)*1e3 ;
        lambda(3,3)*1e3 ;
        UTC0 = UTC(1,:) ;
        fprintf( 'The initial position in km is \n' )
        disp( num2str( r' ) )
        fprintf( 'The initial velocity in km/s is \n' )
        disp( num2str( v' ) )
        fprintf( 'The covariance matrix is \n' )
        disp( num2str( P ) )

    end

    function HW1_P3(newdata, rold , vold , AtWAold , AtWbold , UTC0 )
        mu = 398600 ;
        d2r = pi/180 ;
        r2d = 180/pi ;

        lat = 21.5748 ;
        long = -158.2706 ;
        alt = 300.2 ;
        rhoerr = .0925 ;
        azerr = .0224*d2r ;
        elerr = .0139*d2r ;
        UTC = newdata(:,1:6) ;
        rho = newdata(:,9) ;
        az = newdata(:,7).*d2r ;
        el = newdata(:,8).*d2r ;
        newobs = { rho , az , el } ;
        obserr = [ rhoerr , azerr , elerr ] ;

        [ r , v , P ] = SBF( newobs , obserr , lat , long , alt ,
UTC , UTC0 , rold , vold , AtWAold , AtWbold ) ;
        Ppos = P(1:3,1:3) ;
        [ ev , lambda ] = eig( P ) ;
        lambda(1,1)*1e3 ;
        lambda(2,2)*1e3 ;
        lambda(3,3)*1e3 ;
        fprintf( 'The initial position in km is \n' )
        disp( num2str( r' ) )
```

```
        fprintf( 'The initial velocity in km/s is \n' )
        disp( num2str( v' ) )
        fprintf( 'The covariance matrix is \n' )
        disp( num2str( P ) )
        fprintf( 'The change in initial position in m is \n' )
        disp( num2str( ( r' - rold' )*1e3 ) )
        fprintf( 'The change in initial velocity in m/s is \n' )
        disp( num2str( ( v' - vold' )*1e3 ) )
        fprintf( 'The change in covariance matrix is \n' )
        disp( num2str( P - pinv( AtWAold ) ) )
        fprintf( 'The covariance matrix improves with more
 observations. \n' )
        fprintf( 'The actual change in state is relatively small \n' )
    end
```

Functions

```
function [ yc , rbar , RMS , P ] = LLS1( xoi , yoi )
        n = length( xoi ) ;
        AtA = zeros( 2 ) ;
        AtA(1,1) = n ;
        AtA(1,2) = sum( xoi ) ;
        AtA(2,1) = sum( xoi ) ;
        AtA(2,2) = sum( xoi.^2 ) ;
        Atb = zeros( 2 , 1 ) ;
        Atb(1,1) = sum( yoi ) ;
        Atb(2,1) = sum( yoi.*xoi ) ;
        P = inv( AtA ) ;
        state = P*Atb ;
        alpha = state(1) ;
        beta = state(2) ;
        yc = alpha + beta*xoi ;
        rbar = yoi - yc ;
        RMS = sqrt( sum( rbar.^2 )/n ) ;
end

function [ r , v , RMS , P , AtWA , AtWb ] = DCODrazel2rv( obs ,
 obserr , lat , long , alt , UTC )
% Performs differential correction orbit determination using Weighted
 Least
% Squares technique. The observations for this function should be
 razel
% columns.
    mu = 398600 ;
    d2s = 86400 ;
    rho0 = obs{1} ;
    az0 = obs{2} ;
    el0 = obs{3} ;
    rhoerr = obserr(1) ;
    azerr = obserr(2) ;
    elerr = obserr(3) ;
    n = length(el0) ;
```

```
    W = zeros( 3 ) ;
    W(1,1) = 1/( rhoerr^2 ) ;
    W(2,2) = 1/( azerr^2 ) ;
    W(3,3) = 1/( elerr^2 ) ;
%      W = W/W(1,1) ;
%      W = W/norm(W) ;

    RMS0 = 1 ;
        for ii = 1:n %find position vector from observation
            [ r(:,ii) ] = razel2r( rho0(ii) , az0(ii) , el0(ii) ,
 lat , long , alt , UTC(ii,:) ) ;
            JD(ii) = juliandate( UTC(ii,:) ) ;
            obso(:,ii) = [ rho0(ii) ; az0(ii) ; el0(ii) ] ;
        end
        for ii = 2:(n-1) % find velocities for all but first and last
 observation
            v(:,ii) = Gibbs( r(:,ii-1) , r(:,ii) , r(:,ii+1) ,
 JD(ii-1) , JD(ii) , JD(ii+1) , mu ) ;
        end
        for ii = 2:n-1 % put all states back to initial epoch
            tspan = [ 0 , JD(1) - JD(ii)]*d2s ;
            [ ~ , rback , vback ] = TwoBody( tspan , r(:,ii) ,
 v(:,ii) , mu , 1e-8 ) ;
            r0(:,ii-1) = rback(:,end) ;
            v0(:,ii-1) = vback(:,end) ;
        end
        r0avg = zeros( 3 , 1 ) ;
        v0avg = zeros( 3 , 1 ) ;
        for ii = 1:3 % find average state at initial epoch
            r0avg(ii) = mean( r0(ii,:) ) ;
            v0avg(ii) = mean( v0(ii,:) ) ;
        end
        rnom = r0avg ;
        vnom = v0avg ;
        xnom0 = [ rnom ; vnom ] ;
        xnom0 =
 [5975.29040000000;2568.64000000000;3120.58450000000;-3.98384600000000;-2.07115900
        xnom = zeros( 6 , n ) ;

    tol = 1e-3 ;
    err = 1 ;
    while err >= tol % run until RMS changes by less that %0.1
%          A = zeros( 3 , 6 ) ;
%          btil = zeros( n , 1 ) ;
        AtWA = zeros( 6 , 6 ) ;
        AtWb = zeros( 6 , 1 ) ;
        for ii = 1:n %loop for all observations
            tspan = [ 0 , JD(ii) - JD(1) ]*d2s ;
            if tspan(1) ~= tspan(2)
                [ ~ , rnomi , vnomi ] = TwoBody( tspan , xnom0(1:3) ,
 xnom0(4:6) , mu , 1e-8 ) ;
                xnom(:,ii) = [ rnomi(:,end) ; vnomi(:,end) ] ;
            else
```

```
                xnom(:,1) = xnom0 ;
                rnomi = xnom0(1:3) ;
                vnomi = xnom0(4:6) ;
            end
            [ rhoni , azni , elni , ~ , ~ , ~ , ~ ] =
RAZEL( rnomi(:,end) , vnomi(:,end) , UTC(ii,:) , lat , long , alt ) ;
            btil = [ rho0(ii) - rhoni ; az0(ii) - azni ; el0(ii) -
elni ]  ;
            for jj = 1:6 % finite differencing for every element
                xmod = xnom0(:) ;
                delement = xnom0(jj)*.0001 ;
                xmod(jj) = xnom0(jj) + delement ;
                if tspan(2) ~= 0
                    [ ~ , rmodi , vmodi ] = TwoBody( tspan ,
xmod(1:3) , xmod(4:6) , mu , 1e-8 ) ;
                else
                    rmodi = xmod(1:3) ;
                    vmodi = xmod(4:6) ;
                end
                [ rhomi , azmi , elmi , ~ , ~ , ~ , ~ ] =
RAZEL( rmodi(:,end) , vmodi(:,end) , UTC(ii,:) , lat , long , alt ) ;
                A(:,jj) = [ ( rhomi - rhoni )/delement ; ( azmi -
azni )/delement ; ( elmi - elni )/delement ] ;
            end
            AtWAi = A'*W*A ;
            AtWA = AtWA + AtWAi ;
            AtWbi = A'*W*btil ;
            AtWb = AtWb + AtWbi ;
        end
        P = pinv(AtWA) ;
        delx = P*AtWb ;
        RMS = sqrt( ( btil'*W*btil )/( 3*n ) ) ;
        err = abs( RMS - RMS0 )/RMS ;
        if err >= tol
            xnom0 = xnom0 + delx ;
        end
        RMS0 = RMS ;
    end
    r = xnom0(1:3) ;
    v = xnom0(4:6) ;
end

function [ r ] = razel2r( rho , az , el , lat , long , alt , utc )
    rsite_ecef = lla2ecef( [ lat , long , alt ] )'*1e-3 ;
    rhosez = [ -rho*cos(el)*cos(az) ; rho*cos(el)*sin(az) ;
rho*sin(el) ] ;
    sez2ecef = [ sind(lat)*cosd(long) , -sind(long) ,
cosd(lat)*cosd(long) ; ...
                sind(lat)*cosd(long) , cosd(long) ,
cosd(lat)*sind(long) ; ...
                -cosd(lat) , 0 , sind(lat) ] ;
    rhoecef = sez2ecef*rhosez ;
    eci2ecef = dcmeci2ecef( 'IAU-2000/2006' , utc ) ;
    recef = rsite_ecef + rhoecef ;
```

```
    r = eci2ecef'*recef ;
end

function [ rho , az , el , drho , daz , del , VIS ] = RAZEL( r , v ,
 UTC , lat , long , alt )
% find azimuth and elevation
    Re = 6378 ;
    d2r = pi/180 ;
    JD = juliandate( UTC ) ;
    dcm = dcmeci2ecef( 'IAU-2000/2006' , UTC ) ;
    eci2ecef = dcm ;
    recef = dcm*r ;
    vecef = dcm*v ;
    rsite_ecef = ( lla2ecef( [ lat , long , alt ] )*1e-3 )' ;
    rhoecef = recef - rsite_ecef ;
    drhoecef = vecef ;
    ecef2sez = [ sin(lat*d2r)*cos(long*d2r) ,
 sin(lat*d2r)*sin(long*d2r) , -cos(lat*d2r) ; ...
                -sin(long*d2r) , cos(long*d2r) , 0 ; ...
                cos(lat*d2r)*cos(long*d2r) ,
 cos(lat*d2r)*sin(long*d2r) , sin(lat*d2r) ] ;
    rhosez = ecef2sez*rhoecef ;
%     rhosez = angle2dcm( 0 , (90-lat)*d2r , 0 )*angle2dcm( 0 , 0 ,
 long )*rhoecef ;
%     rhosez = angle2dcm( 0 , lat , 0 )*angle2dcm( 0 , 0 ,
 long )*rhoecef ;
    drhosez = ecef2sez*drhoecef ;
%                 drhosez = angle2dcm( 0 , (90-lat)*d2r ,
 0 )*angle2dcm( 0 , 0 , long )*drhoecef ;
%                 drhosez = angle2dcm( 0 , lat , 0 )*angle2dcm( 0 ,
 0 , long )*drhoecef ;
                rho = norm( rhosez ) ;
                el = asin( rhosez(3)/rho ) ;
                if el ~= pi/2
                    sin_az = rhosez(2) / sqrt( rhosez(1)^2 +
 rhosez(2)^2 ) ;
                    cos_az = -rhosez(1) / sqrt( rhosez(1)^2 +
 rhosez(2)^2 ) ;
                    if sin_az >= 0 && cos_az >= 0
                        az = asin( sin_az ) ;
                    elseif sin_az >= 0 && cos_az <= 0
                        az = pi - asin( sin_az ) ;
                    elseif sin_az <= 0 && cos_az <= 0
                        az = pi - asin( sin_az ) ;
                    elseif sin_az <= 0 && cos_az >= 0
                        az = asin( sin_az ) + 2*pi ;
                    end
                else
                    sin_az = drhosez(2) / sqrt( drhosez(1)^2 +
 drhosez(2)^2 ) ;
                    cos_az = drhosez(1) / sqrt( drhosez(1)^2 +
 drhosez(2)^2 ) ;
                    if sin_az >= 0 && cos_az >= 0
                        az = asin( sin_az ) ;
```

```
                    elseif sin_az >= 0 && cos_az <= 0
                        az = pi - asin( sin_az ) ;
                    elseif sin_az <= 0 && cos_az <= 0
                        az = pi - asin( sin_az ) ;
                    elseif sin_az <= 0 && cos_az >= 0
                        az = asin( sin_az ) + 2*pi ;
                    end
                end
                drho = dot( rhosez , drhosez )/rho ;
                daz = ( drhosez(1)*rhosez(2) - drhosez(2)*rhosez(1) )/
( rhosez(1)^2 + rhosez(2)^2 ) ;
                del = ( rhosez(3) - drho*sin(el) )/sqrt( rhosez(1)^2 +
 rhosez(2)^2 ) ;

    if rhosez(3) >= 0
        rs = SunVector( JD ) ;
        if dot( rs , rsite_ecef ) > 0
            VIS = "Radar Sun" ;
        else
            ang = asin( norm( cross( rs , recef ) )/
( norm( rs )*norm( recef ) ) ) ;
            dist = norm( recef )*cos( ang - pi/2 ) ;
            if dist > Re
                VIS = "Visible" ;
            else
%                 VIS = "Radar Night" ;
                VIS = "Visible" ;
            end
        end
    else
        VIS = "Obscured" ;
    end
end

function [ r , v , P ] = SBF( newobs , obserr , lat , long , alt ,
 UTC , UTC0 , rold , vold , AtWAold , AtWbold )
     mu = 398600 ;
    d2s = 86400 ;
    rho0 = newobs{1} ;
    az0 = newobs{2} ;
    el0 = newobs{3} ;
    rhoerr = obserr(1) ;
    azerr = obserr(2) ;
    elerr = obserr(3) ;
    n = length(el0) ;

    W = zeros( 3 ) ;
    W(1,1) = 1/( rhoerr^2 ) ;
    W(2,2) = 1/( azerr^2 ) ;
    W(3,3) = 1/( elerr^2 ) ;
%     W = W/W(1,1) ;
%     W = W/norm(W) ;

    xnom0 = [ rold ; vold ] ;
```

```
    JD0 = juliandate( UTC0 ) ;


    AtWA = zeros( 6 , 6 ) ;
        AtWb = zeros( 6 , 1 ) ;
        for ii = 1:n %loop for all observations
            JD(ii) = juliandate( UTC(ii,:) ) ;
            tspan = [ 0 , JD(ii) - JD0 ]*d2s ;
            if tspan(1) ~= tspan(2)
                [ ~ , rnomi , vnomi ] = TwoBody( tspan , xnom0(1:3) ,
 xnom0(4:6) , mu , 1e-8 ) ;
                xnom(:,ii) = [ rnomi(:,end) ; vnomi(:,end) ] ;
            else
                xnom(:,1) = xnom0 ;
                rnomi = xnom0(1:3) ;
                vnomi = xnom0(4:6) ;
            end
            [ rhoni , azni , elni , ~ , ~ , ~ , ~ ] =
 RAZEL( rnomi(:,end) , vnomi(:,end) , UTC(ii,:) , lat , long , alt ) ;
            btil = [ rho0(ii) - rhoni ; az0(ii) - azni ; el0(ii) -
 elni ]  ;
            for jj = 1:6 % finite differencing for every element
                xmod = xnom0(:) ;
                delement = xnom0(jj)*.0001 ;
                xmod(jj) = xnom0(jj) + delement ;
                if tspan(2) ~= 0
                    [ ~ , rmodi , vmodi ] = TwoBody( tspan ,
 xmod(1:3) , xmod(4:6) , mu , 1e-8 ) ;
                else
                    rmodi = xmod(1:3) ;
                    vmodi = xmod(4:6) ;
                end
                [ rhomi , azmi , elmi , ~ , ~ , ~ , ~ ] =
 RAZEL( rmodi(:,end) , vmodi(:,end) , UTC(ii,:) , lat , long , alt ) ;
                A(:,jj) = [ ( rhomi - rhoni )/delement ; ( azmi -
 azni )/delement ; ( elmi - elni )/delement ] ;
            end
            AtWAi = A'*W*A ;
            AtWA = AtWA + AtWAi ;
            AtWbi = A'*W*btil ;
            AtWb = AtWb + AtWbi ;
        end

        delx = pinv( AtWA + AtWAold )*( AtWb + AtWbold ) ;
        P = pinv( AtWA + AtWAold ) ;
        r = rold + delx(1:3) ;
        v = vold + delx(4:6) ;
end

end
```

Published with MATLAB® R2019a