```
function [R, V, rv_cov, refrence_time] = batch_angles(times, measurements, R_obs,↙
GPS_times, GPS_measurements,P_GPS, Epoch)
%-------------------------------------------------------------------------
% Batch least squares orbit estimation from ECI RA, DEC, and their rates
%-------------------------------------------------------------------------
% Inputs:
%   times           [Nx1]   times in seconds of angle observations (assumes
%                           ascending order
%
%   measurements    [Nx4]   [RA, DEC, RA_dot, DEC_dot] in radians and rad/s
%
%   R_obs           [4x4]   covariance of each measurement (constant diagonal)
%
%   GPS_measurments [Nx6]   matris of gps measurments to find the
%                           closes time to set as refrence orbit.  Each measurment↙
has r_x, r_y,
%                           r_z [in meters], v_x, v_y, v_z [m/s]
%
%   GPS_times       [NX1]   array of the times from the epoch time [sec]
%                           that the gps measurments were taken
%
%   P_GPS           [6x6]   GPS measurment covariance matrix first 3x3 in [m^2],
%                           second 3x3 in [(m/s)^2]
%
%   Epoch           datetime    The Epoch of all the measurements

% Outputs:
%   kep_est [6x1] [a, e, i, RAAN, omega, nu]
%
%   rv_cov  [6x6] covariance of coodinate system [m^2 and (m/s)^2]
%
%   refrence_time  The time [s] of the final calcualted refrence trajectory
%
% Assumptions:
%   - RA/DEC are in the ECI frame
%   - Keplerian two-body motion (no perturbations)


% Parameters
mu = 3.986004418e14; % m^3/s^2
N_obs = length(times);

% From closest GPS sighting
[~, idx] = min(abs(GPS_times - times(1)));
last_gps_sighting = Epoch + seconds(GPS_times(idx-1));  %idx-1 becuase our↙
measurments start at t=0

% Initial State and error
X_0_star = GPS_measurements(idx-1, :)';
x_0_bar =  X_0_star - X_0_star;  % We set the refrence trajectory equal to the↙
latest gps measrument.  Ajustable for further use
% X_0 = X_0_star + x_0_bar;

% Convergance Limits
```

```
tol = 1e-8;
max_iter = 20;
norm_tracker=[];

for iter = 1:max_iter

    % A priori estimate
    Lambda = P_GPS^-1;
    N = Lambda * x_0_bar;

    for k = 1:N_obs

        % Propogate X to time of measurmant k
        R_k = inv(R_obs);  % Editable for future use in case of non constant R_obs
        dt = Epoch + seconds(times(k)) - last_gps_sighting;
        Phi_k = Kepler_STM_UV(seconds(dt), X_0_star, mu, 'seconds');  % state
transition matrix for keplerian orbits
        X_pred = Phi_k * X_0_star;
        r = [X_pred(1:3)];
        v = [X_pred(4:6)];


        % Compute predicted RA/DEC and their rates in ECI
        RA_pred = wrapTo2Pi(atan2(r(2), r(1)));
        DEC_pred = asin(r(3)/norm(r));
        RA_dot_pred = (r(1)*v(2) - r(2)*v(1)) / (r(1)^2 + r(2)^2);
        DEC_dot_pred = (v(3) - (r(3)/norm(r))*dot(r,v)/norm(r)) / sqrt(r(1)^2 + r(2)
^2);

        dRA = wrapToPi(measurements(k,1) - RA_pred);
        dDEC = measurements(k,2) - DEC_pred;  % or wrapToPi if needed
        dRAdot = measurements(k,3) - RA_dot_pred;
        dDECdot = measurements(k,4) - DEC_dot_pred;
        y_k = [dRA; dDEC; dRAdot; dDECdot];

        % Analytic measurement Jacobian
        H_tilda_k = compute_H_analytic(r, v);
        H_k = H_tilda_k * Phi_k;

        % Update Observation Reading
        Lambda = Lambda + H_k' * (R_k \ H_k);
        N = N + H_k' * (R_k \ y_k);
    end

    % Update Covariance
    P_0 = inv(Lambda);

    % Batch Least Squares update
    x_0_hat = Lambda \ N;
    X_0_star = X_0_star + x_0_hat;
    x_0_bar = x_0_bar - x_0_hat;

    % Convergance Tracker
    norm_tracker=[norm_tracker; norm(x_0_hat)];
```

```
    if norm(x_0_hat) < tol
        break;
    end
end

R = X_0_star(1:3);
V = X_0_star(4:6);

% Covariance
rv_cov = diag(diag(P_0));

refrence_time = seconds(last_gps_sighting-Epoch);
end
```