Table of Contents

```
function [xk_filt_hist, Pk_filt_hist] = kalman_filter_hw8(tvec, F, G, xk, u,
Pk, Qkf, Rkf, ykhist, H)
    % Linearized Kalman Filter as presented in Lec26, Slide 6 of ASEN 5044,
    % Fall 2024 and copied from L26_1Drobotstatefilter.m code
    % Inputs
    % tvec - vector of time steps where T is total time steps [Tx1]
    % F - DT STM matrix [nxn]
    % G - DT control matrix [nxm] where m is total number of control inputs
    % xk - initial xk [nx1]
    % u - DT control vector [mxT]
    % Pk - initial Pk [nxn]
    % Qkf - Kalman Filter Process noise [nxn]
    % Rkf - Kalman Filter measurement covariance [pxp]
    % ykhist - Measurement history [pxT]
    % H - DT measurement matrix [pxn]
    %
    % Outputs
    % xk_filt_hist - Kalman filter estimation states [nxT]
    % Pk_filt_hist - Kalman filter estimation state covar [nxnxT]

    n = length(F);

    for k=1:length(tvec)

        %%Perform prediction step
        xkp1_minus = F*xk + G*u(:,k);
        Pkp1_minus = F*Pk*F' + Qkf;

        %%Compute Kalman gain
        Kkp1 = Pkp1_minus*H'*inv(H*Pkp1_minus*H' + Rkf);

        %%Perform measurement update step
        ykp1_report = ykhist(:,k); %pull report of actual data from sensor
        ykp1_pred = H*xkp1_minus; %predicted measurement
        innov_kp1 = ykp1_report - ykp1_pred; %compute innovation
        xkp1_plus = xkp1_minus + Kkp1*innov_kp1; %compute update to state mean
        Pkp1_plus = (eye(n) - Kkp1*H)*Pkp1_minus; %compute update to covar

        %%store results and cycle for next iteration
        xk = xkp1_plus;
        xk_filt_hist(:,k) = xkp1_plus;
        Pk = Pkp1_plus;
        Pk_filt_hist(:,:,k)=Pkp1_plus;
    end
end
```

```
Not enough input arguments.
```

```
Error in kalman_filter_hw8 (line 20)
    n = length(F);
```

Published with MATLAB® R2024a