Initialize

```
clc;
clear;
close all;

startup;
% [xhat, meas] = Task5_filterTemplate()
[xhat, meas] = filterTemplate()


% load("placeflat.mat");
```

plot

```
t= meas.t;

% plotRead(meas.acc, t, 'acc',0);
% plotRead(meas.gyr, t, 'gyro',0);
% plotRead(meas.mag, t, 'mag',0)
% plotRead(meas.orient,t, 'orient',1)
```

histogram & Gaussian

```
% % plot acc
plotHistograms(meas.acc, 'acc');
%
% % plot gyro
plotHistograms(meas.gyr, 'gyro');
%
% plot mag
plotHistograms(meas.mag, 'mag');

close all;
```

Task 11

```
clc;
clear;
close all;
showIP;
startup;
% load('final_all.mat')
figPosition = [744.2000  300.2000  872.8000  749.6000];
f = @GyroandMag;

[xhat, meas] = f();
close all;

[t_filter, filter] = myclear(xhat.t, xhat.x);
[t_measure, phone] = myclear(meas.t, meas.orient);

newFig = figure;
set(newFig, 'Position', figPosition);

% sgtitle('Compare filter and phone with MAG ACC GYRO'); % 添加总标题
hold on;

subplot(3,1,1);
hold on;
plot(t_filter,filter(1,:),'LineWidth',2,'LineStyle',':');
plot(t_measure,phone(1,:), 'LineWidth',2)
legend('filter','phone\_measurement')
ylabel('yaw');

subplot(3,1,2);
hold on;
plot(t_filter,filter(2,:),'LineWidth',2,'LineStyle',':');
plot(t_measure,phone(2,:), 'LineWidth',2)
legend('filter','phone\_measurement')
ylabel('pitch');

subplot(3,1,3);
hold on;
plot(t_filter,filter(3,:),'LineWidth',2,'LineStyle',':');
plot(t_measure,phone(3,:), 'LineWidth',2)
legend('filter','phone\_measurement')
ylabel('roll');
```

help function: plotHistograms

```
function plotHistograms(data, sensorName)
    % NaN
    dataX = data(1, 1:end-10);
    dataX = dataX(~isnan(dataX));
```

```
    dataY = data(2, 1:end-10);
    dataY = dataY(~isnan(dataY));
    dataZ = data(3, 1:end-10);
    dataZ = dataZ(~isnan(dataZ));

    % mean and std
    meanX = mean(dataX);
    stdX = std(dataX);
    fprintf('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%');
    fprintf('\n');
    fprintf('The mean of %s -X is %f \n, the covariance of %s -X is %f  \n\n',...
        sensorName, meanX, sensorName, stdX);

    meanY = mean(dataY);
    stdY = std(dataY);
    fprintf('The mean of %s -Y is %f \n, the covariance of %s -Y is %f \n\n',...
        sensorName, meanY, sensorName, stdY);

    meanZ = mean(dataZ);
    stdZ = std(dataZ);
    fprintf('The mean of %s -Z is %f \n, the covariance of %s -Z is %f \n\n',...
        sensorName, meanZ, sensorName, stdZ);
    % histogram
    figure;
    subplot(3,1,1);
    histogram(dataX, 'Normalization', 'pdf');
    title(sprintf('%s - X', sensorName));

    subplot(3,1,2);
    histogram(dataY, 'Normalization', 'pdf');
    title(sprintf('%s - Y', sensorName));

    subplot(3,1,3);
    histogram(dataZ, 'Normalization', 'pdf');
    title(sprintf('%s - Z', sensorName));
    fprintf('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%\n');
end
```

help function: myclear

```
function [tout, eulerout] = myclear(t, data)
    valid_idx = ~isnan(data(1,:));
    tout = t(valid_idx);
%     data_out = zeros(4,length(valid_idx));
    for i =1:4
    data_out(i,:) = data(i,valid_idx);
    end
    eulerout = q2euler(data_out) * 180/pi;
```

```
end
```

Task function

mu_g

```
function [x, P] = mu_g(x, P, yacc, Ra, g0)
%g0 is gravity vector
%Ra is eak
%fak =0, assumption
%yak=Q' *g0 +Ra
%following chapter 4 kalman filter update

Q = Qq(x);
h = Q'*g0;

[Q0, Q1, Q2, Q3] = dQqdq(x);
% H = [Q0, Q1, Q2, Q3]*g0; this cannot works
H = [Q0'*g0 Q1'*g0 Q2'*g0 Q3'*g0];

%innovation covariance
S = H * P * H' + Ra;

%innovation
v = yacc- h;

%kalman gain
K = P * H' * inv(S);

x = x + K* v;
P = P - K* S *K';

end
```

mu_m

```
function [x, P] = mu_m(x, P, mag, m0, Rm)

Q = Qq(x);
h = Q'*m0;

[Q0, Q1, Q2, Q3] = dQqdq(x);
% H = [Q0, Q1, Q2, Q3]*g0; this cannot works
H = [Q0'*m0 Q1'*m0 Q2'*m0 Q3'*m0];

%innovation covariance
S = H * P * H' + Rm;

%innovation
```

```
v = mag- h;

%kalman gain
K = P * H' * inv(S);

x = x + K* v;
P = P - K* S *K';

end
```

mu_normalizeQ

```
function [x, P] = mu_normalizeQ(x, P)
% MU_NORMALIZEQ  Normalize the quaternion

  x(1:4)  = x(1:4) / norm(x(1:4));
  x = x*sign(x(1));
end
```

tu_qw

```
function [x,P] = tu_qw(x, P, omega, T, Rw)
% instead of thinking T as the time since the last measurement, it can be
% better to think it as the sampling interval.

F = eye(size(x, 1)) + (T/2) * Somega(omega);
G = (T/2) * Sq(x);

x = F * x;
P = F * P * F' + G * Rw * G';

% else
% x = x;
% P = P + Rw;
end
```