```
%% extra function code
function [x,y] = normpdf2(mu, sigma2, level, N)
    x = linspace(mu-level*sqrt(sigma2),
mu+level*sqrt(sigma2),N);
    y = normpdf(x, mu, sqrt(sigma2));
end


%% main code
% clear all;
% clc;

[xhat, meas] = sundar_filt()

 %%

acc_mean = mean(meas.acc(:, ~any(isnan(meas.acc), 1)), 2)
mag_mean =mean(meas.mag(:, ~any(isnan(meas.mag), 1)), 2)
gyr_mean =mean(meas.gyr(:, ~any(isnan(meas.gyr), 1)), 2)


acc_cov = cov(meas.acc(:, ~(isnan(meas.acc(1,:))))')
mag_cov = cov(meas.mag(:, ~(isnan(meas.mag(1,:))))')
gyr_cov = cov(meas.gyr(:, ~(isnan(meas.gyr(1,:))))')

%% Task 2-a
[xhat, meas] = filterTemplate()
close all;

figure
subplot(3,1,1)
hold on;
grid on;
plot(meas.t', meas.acc', 'LineWidth', 2)
xlim([0, max(meas.t)])
xlabel('time')
ylabel('acceleration')
title ('Accelerometer')
legend('x','y','z')

subplot(3,1,2)
hold on;
grid on;
plot(meas.t',meas.gyr','LineWidth', 1)
```

```matlab
xlim([0, max(meas.t)])
xlabel('time')
ylabel('angular velocity')
title ('Gyroscope')
legend('x','y','z')

subplot(3,1,3)
hold on;
grid on;
plot(meas.t', meas.mag', 'LineWidth', 2)
xlim([0, max(meas.t)])
xlabel('time')
ylabel('magnetic field')
title ('Magnetometer')
legend('x','y','z')

%% Task 2-b
[xhat, meas] = filterTemplate()

close all;
axis_name = {'x','y','z'};

accelero = meas.acc(:,~isnan(meas.acc(1,:)))';

mu = mean(accelero)
Ra = cov(accelero)

figure
for i=1:3
    subplot(3,1,i);
    hold on;
    grid on;
    histogram(accelero(:,i), 50, 'Normalization','pdf')
    xlabel([axis_name{i},' [m/s^2]']);
    [x,y] = normpdf2(mu(i), Ra(i,i), 3, 100);
    plot(x,y,'LineWidth',2);
end
sgtitle('acceleration')

gyro = meas.gyr(:,~isnan(meas.gyr(1,:)))';

mu = mean(gyro)
Rw = cov(gyro)
```

```
figure
for i=1:3
    subplot(3,1,i);
    hold on;
    grid on;
    histogram(gyro(:,i), 'Binwidth', 1.01e-3,
'Normalization','pdf')
    xlabel([axis_name{i},' [rad/s]']);
    [x,y] = normpdf2(mu(i), Rw(i,i), 3, 100);
    plot(x,y,'LineWidth',2);
end
sgtitle('angular velocity')

magneto = meas.mag(:,~isnan(meas.mag(1,:)))';

mu = mean(magneto)
Rm = cov(magneto)

figure
for i=1:3
    subplot(3,1,i);
    hold on;
    grid on;
    histogram(magneto(:,i), 'Binwidth', 1.01e-1,
'Normalization','pdf')
    xlabel([axis_name{i},' [uT]']);
    [x,y] = normpdf2(mu(i), Rm(i,i), 3, 100);
    plot(x,y,'LineWidth',2);
end
sgtitle('magnetic field')

%% Question 12
close all;

[xhat, meas] = sundar_filt()

figure;
sgtitle('Orientation estimation comparison')

subplot(3,1,1)
hold on;
grid on;
plot(xhat.t(:,:,1)',180/pi*([1 0
0]*q2euler(xhat.x(:,:,1)))','LineWidth',1.5)
```

```
plot(xhat.t(:,:,1)',180/pi*([1 0
0]*q2euler(meas.orient(:,:,1)))','LineWidth',1.5)
xlabel('time')
ylabel ('phi')
legend('Own','Google')

subplot(3,1,2)
hold on;
grid on;
plot(xhat.t(:,:,1)',180/pi*([0 1
0]*q2euler(xhat.x(:,:,1)))','LineWidth', 1.5)
plot(xhat.t(:,:,1)',180/pi*([0 1
0]*q2euler(meas.orient(:,:,1)))','LineWidth', 1.5)
xlabel('time')
ylabel('theta')
legend('Own','Google')

subplot(3,1,3)
hold on;
grid on;
plot(xhat.t(:,:,1)',180/pi*([0 0
1]*q2euler(xhat.x(:,:,1)))','LineWidth', 1.5)
plot(xhat.t(:,:,1)',180/pi*([0 0
1]*q2euler(meas.orient(:,:,1)))','LineWidth', 1.5)
xlabel ('time')
ylabel ('psi')
legend('Own','Google')




%% filter code


function [xhat, meas] = sundar_filt(calAcc, calGyr, calMag)
% FILTERTEMPLATE  Filter template
%
% This is a template function for how to collect and filter
data
% sent from a smartphone live.  Calibration data for the
% accelerometer, gyroscope and magnetometer assumed
available as
% structs with fields m (mean) and R (variance).
```

```
%
% The function returns xhat as an array of structs
comprising t
% (timestamp), x (state), and P (state covariance) for each
% timestamp, and meas an array of structs comprising t
(timestamp),
% acc (accelerometer measurements), gyr (gyroscope
measurements),
% mag (magnetometer measurements), and orint (orientation
quaternions
% from the phone).  Measurements not availabe are marked
with NaNs.
%
% As you implement your own orientation estimate, it will
be
% visualized in a simple illustration.  If the orientation
estimate
% is checked in the Sensor Fusion app, it will be displayed
in a
% separate view.
%
% Note that it is not necessary to provide inputs (calAcc,
calGyr, calMag).

  %% Setup necessary infrastructure
  import('com.liu.sensordata.*');  % Used to receive data.

  %% Filter settings
  t0 = [];  % Initial time (initialize on first data
received)
  nx = 4;   % Assuming that you use q as state variable.
  % Add your filter settings here.

g0 = [0.0696 -0.0410 9.8216]';
rangeSkipAcc = 0.2; % accept acc meas. if 80%|g0| < |acc| <
120%|g0|
rangeSkipMag = 0.1;

mag_mean = [-87.1983 48.6800 -73.3492]; %obtained from
smartphone
m0= [0 99.86 -73.35]'; %from eqn 9
mx=0;
my=99.86;
mz=-73.35;
```

```
gyr_mean =1.0e-03 * [0.0389 0.2201 -0.0710]';


Ra = 1.0e-03 * [0.0723    0.0014   -0.0009
                0.0014    0.0624    0.0007
               -0.0009    0.0007    0.1149];


Rm = [ 0.1939    0.0101   -0.0088
       0.0101    0.1984    0.0242
      -0.0088    0.0242    0.1963];


Rw =1.0e-06 * [0.2858    0.0594   -0.0128
               0.0594    0.5395   -0.0362
              -0.0128   -0.0362    0.3160];
L = norm(m0);
alpha = 0.01;

  % Current filter state.
  x = [1; 0; 0 ;0];
  P = eye(nx, nx);

  % Saved filter states.
  xhat = struct('t', zeros(1, 0),...
                'x', zeros(nx, 0),...
                'P', zeros(nx, nx, 0));

  meas = struct('t', zeros(1, 0),...
                'acc', zeros(3, 0),...
                'gyr', zeros(3, 0),...
                'mag', zeros(3, 0),...
                'orient', zeros(4, 0));
  try
    %% Create data link
    server = StreamSensorDataReader(3400);
    % Makes sure to resources are returned.
    sentinel = onCleanup(@() server.stop());

    server.start();  % Start data reception.

    % Used for visualization.
    figure(1);
    subplot(1, 2, 1);
    ownView = OrientationView('Own filter', gca);  % Used
for visualization.
    googleView = [];
    counter = 0;  % Used to throttle the displayed frame
rate.

    %% Filter loop
    while server.status()  % Repeat while data is available
      % Get the next measurement set, assume all
measurements
      % within the next 5 ms are concurrent (suitable for
sampling
      % in 100Hz).
      data = server.getNext(5);

      if isnan(data(1))  % No new data received
        continue;        % Skips the rest of the look
      end
      t = data(1)/1000;  % Extract current time

      if isempty(t0)  % Initialize t0
        t0 = t;
      end
acc = data(1, 2:4)';
if ~any(isnan(acc)) % Acc measurements are available.
    if abs(norm(acc)-norm(g0)) < norm(g0)*rangeSkipAcc
        [x, P] = mu_g(x, P, acc, Ra, g0);
        [x, P] = mu_normalizeQ(x, P);
        ownView.setAccDist(0);
    else
        % skip measurement update
        ownView.setAccDist(1);
    end
end

gyr = data(1, 5:7)';
if ~any(isnan(gyr)) % Gyro measurements are available.
    [x, P] = tu_qw( x, P, gyr, 0.001, Rw); %where T=sample
time=0.001
    [x, P] = mu_normalizeQ(x, P);
else
Rq = eye(4)*0.1;
[x, P] = randwalk( x, P, Rq);
```

```
[x, P] = mu_normalizeQ(x, P);
end

mag = data(1, 8:10)';
if ~any(isnan(mag)) % Mag measurements are available.
    L = (1-alpha)*L+alpha*norm(mag);
    if abs(L-norm(mag)) < L*rangeSkipMag
        [x, P] = mu_m(x, P, mag, Rm, m0);
        [x, P] = mu_normalizeQ(x, P);
        ownView.setMagDist(0);
    else
        ownView.setMagDist(1);
    end
end

      orientation = data(1, 18:21)';  % Google's
orientation estimate.

      % Visualize result
      if rem(counter, 10) == 0
        setOrientation(ownView, x(1:4));
        title(ownView, 'OWN', 'FontSize', 16);
        if ~any(isnan(orientation))
          if isempty(googleView)
            subplot(1, 2, 2);
            % Used for visualization.
            googleView = OrientationView('Google filter',
gca);
          end
          setOrientation(googleView, orientation);
          title(googleView, 'GOOGLE', 'FontSize', 16);
        end
      end
      counter = counter + 1;

      % Save estimates
      xhat.x(:, end+1) = x;
      xhat.P(:, :, end+1) = P;
      xhat.t(end+1) = t - t0;

      meas.t(end+1) = t - t0;
      meas.acc(:, end+1) = acc;
      meas.gyr(:, end+1) = gyr;
      meas.mag(:, end+1) = mag;
      meas.orient(:, end+1) = orientation;
    end
  catch e
    fprintf(['Unsuccessful connecting to client!\n' ...
      'Make sure to start streaming from the phone
*after*'...
             'running this function!']);
  end
end
```