Problem Set #2 (ver 1.0)

Due: 12:00Noon PST, Nov 30, 2020

URL: https://docs.google.com/document/d/1U9n25TT9PgdkpdSiawzmji4aeJE5q6Zw-nywh7hGLPw

Instructions:

1. *This problem set is to be done individually.*
2. *There are no no-penalty late days. However, you may take up to four late days (24-hour intervals) with a penalty of 20% (of the score you get) per full or fractional day. After that score becomes 0.*
3. *You have to submit this homework via Gradescope under PSET #2 (PDF). You have already been enrolled there using the official roster. In the PDF that you upload on Gradescope, please start the answer to every problem on a new page and make sure to assign pages to each problem/subproblem correctly within Gradescope.*
4. *We recommend that you make a copy of this Google Doc, and then edit it to include your answers, and then print to PDF. We have provided space for your answers in this file, but feel free to insert additional pages (just insert a page break from the Insert menu).*

In this assignment, you will turn your smartphone into a wireless inertial measurement unit that can stream inertial data to your computer in real-time. You will then use the data to try out various attitude and pose estimation filters and see how they respond to smartphone movement. The hands-on part of the assessment will be followed by a set of conceptual questions at the end.

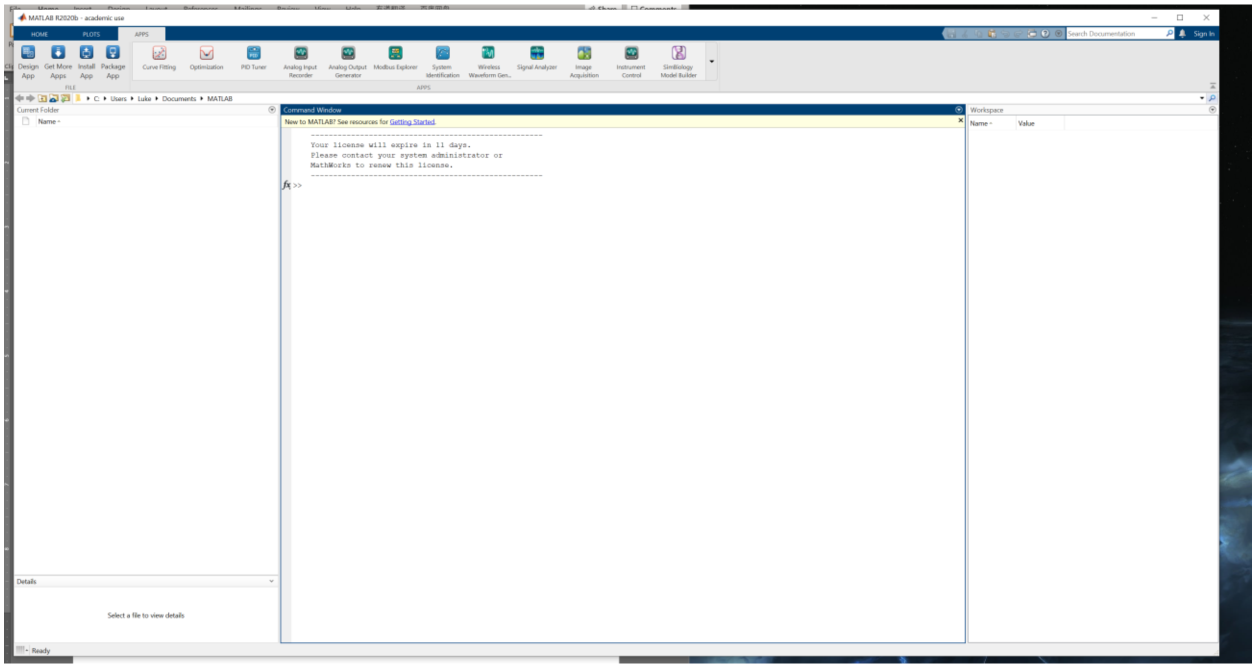
Necessary hardware components:

- Smartphone (Android recommended but iOS would work as well)
- Computer (Mac or Windows recommended but Linux should work as well) with at least 35 GB of free storage (free storage required only if you do not have MATLAB 2020 installed).
- Smartphone and computer must be connected to the same Wi-Fi network.

Problem 1: Install and Run MATLAB on your computer [0.5 points]

- Visit https://softwarecentral.ucla.edu/matlab-getmatlab to find out how to get the latest version of MATLAB (R2020a or R2020b) free-of-cost. Follow the steps to download the installer for your particular machine.
- Use the guide at https://www.youtube.com/watch?v=f1UoHTf_Kgk to install MATLAB. At “Select products to install” screen, select all the components.
- To receive credit for this problem, launch MATLAB and paste a screenshot of the MATLAB IDE in the space below.
- If you do not know the MATLAB programming language, we encourage you to look up tutorials on MATLAB online. A couple of good starting points are:
  https://www.youtube.com/playlist?list=PL7CAABC40B2825C8B
  https://www.youtube.com/playlist?list=PLnVYEpTNGNtX6FcQm90I0WXdvhoEJPp3p
- In addition, unlike Python, every entity of MATLAB has detailed documentation. If you do not understand a certain portion of code or want to look something up, you can search on Google and get official documentation.

Answer:



New to MATLAB? See resources for Getting Started.

```
    --------------------------------------------------
    Your license will expire in 11 days.
    Please contact your system administrator or
    MathWorks to renew this license.
    --------------------------------------------------
fx >>
```

Problem 2: Installing smartphone application [0.5 points]

If you have an Android Smartphone: Download the *Sensorstream IMU+GPS* app for your android phone. The app allows you to log inertial sensor and GPS data to storage as well as stream the data over Wi-Fi via User Datagram Protocol (UDP)

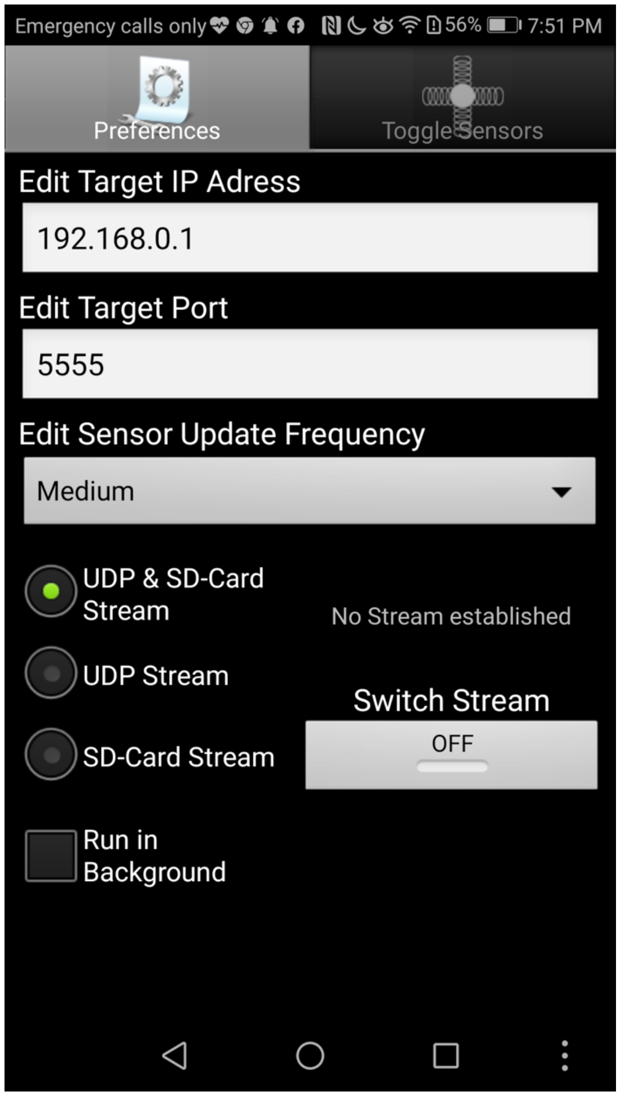
https://play.google.com/store/apps/details?id=de.lorenz_fenster.sensorstreamgps

If you an iOS Smartphone: Download *SensorLog*, which is the iOS equivalent of the *Sensorstream IMU+GPS* app on Android. Note that it's a paid application. You are welcome to use any iOS app with similar functionality if you can find one.

https://apps.apple.com/us/app/sensorlog/id388014573

To get credit for this problem, launch the installed app on your smartphone and paste a screenshot of the app's landing page as your answer.

Answer:



Edit Target IP Adress: 192.168.0.1

Edit Target Port: 5555

Edit Sensor Update Frequency: Medium

[✓ UDP & SD-Card Stream] UDP Stream SD-Card Stream

No Stream established

Switch Stream: OFF

Run in Background

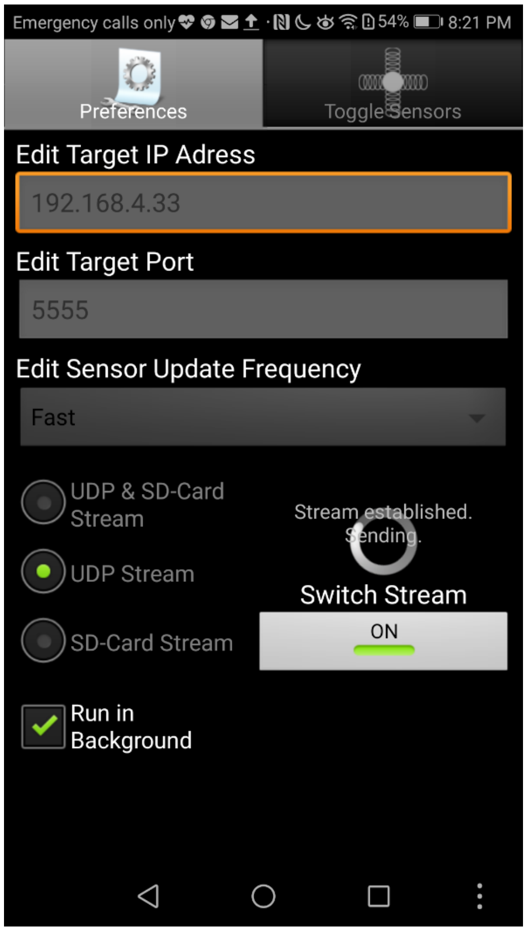
Problem 3: Streaming sensor data from phone to computer [2 points]:

- Ensure your smartphone and computer are on the same Wi-Fi network.
- Download the MATLAB script for reading UDP streams from the link below
  https://github.com/andresmendes/android-sensors-2-matlab
- Find out the public IP address of your smartphone and computer.
- Smartphone app configuration
  - the target IP address should be set to the IP address of the computer with any random port (e.g. 5555, do not set to auto or keep blank)
  - the protocol should be set to UDP/UDP Stream (for iOS app, set "mode" to server).
  - sensor update frequency / logging rate should be set to "fast" or "50 Hz". Additionally for the iOS app, set "log to stream" option to ON and set "log format" to "csv" with delimiter as ",".
  - If there's an option to run in the background, turn it on.
  - Sensors to record:
    - Android: Accelerometer, Gyroscope, Magnetic Field, Orientation, Lin. Acc., Gravity (check "include user-checked… in stream" option)
    - iOS: CMAccelerometer, CMGyroData, CMMagnetometerData, CMDeviceMotion (which consists of (yaw, roll, pitch) - orientation, rotationRate, userAcceleration – lin. acc., attitudeReferenceFrame, quaternions, gravity, magneticField, magneticField.accuracy (p.s. if you use any other app, record the equivalent entities mentioned in the android sensors).
- MATLAB configuration:
  - the IP address in the script should be set to the IP address of the phone with the port you entered on the smartphone app.

To get credit for this problem:

1. Paste a screenshot of the smartphone app streaming data
2. Modify the MATLAB script to store the decoded UDP messages in an array/matrix instead of showing plots or just printing to the command window. Include the modified MATLAB script below and the first 5 rows of the matrix.
   - *Hint: First 5 rows of the matrix should correspond to decoded UDP packets received at the first 5 timestamps. You may also need to modify the buffer size. Be careful not to overflow the array, you may keep the size to a finite length for now.*

Answer:



Edit Target IP Adress

192.168.4.33

Edit Target Port

5555

Edit Sensor Update Frequency

Fast

UDP & SD-Card Stream

[✓ UDP Stream]

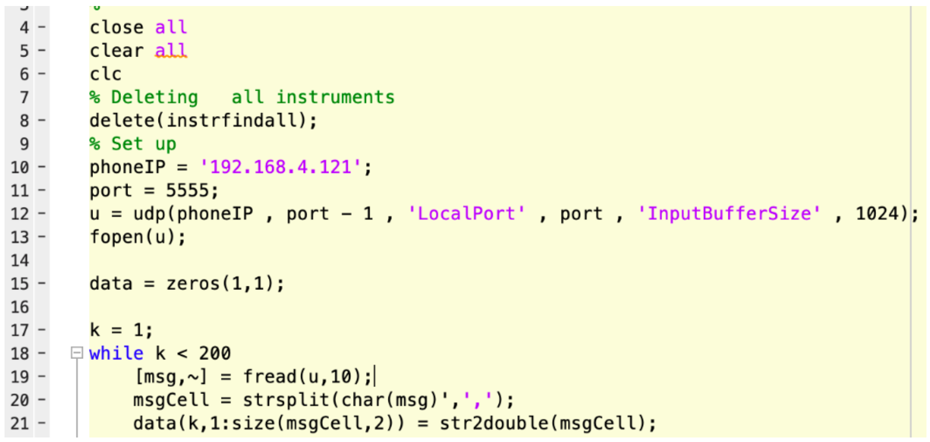
SD-Card Stream

Stream established. Sending.

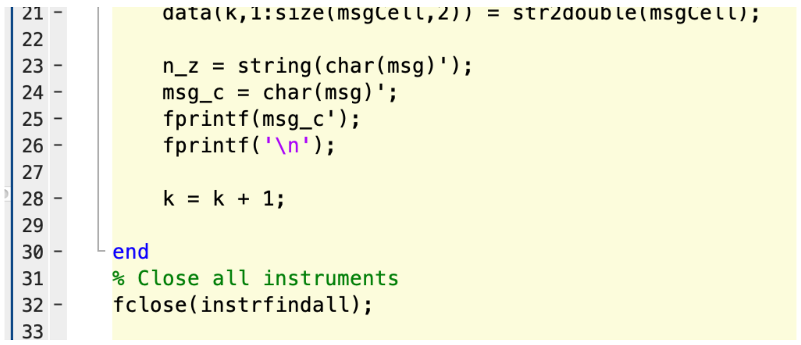
Switch Stream

ON

[✓ Run in Background]

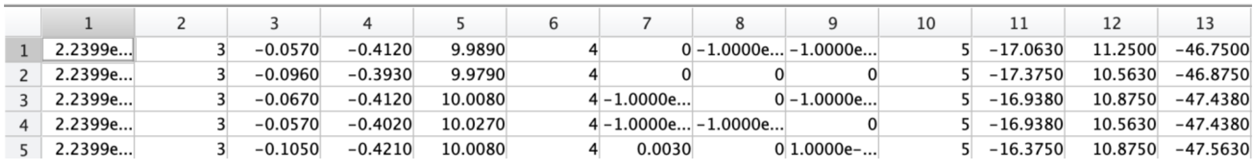


```
%
close all
clear all
clc
% Deleting   all instruments
delete(instrfindall);
% Set up
phoneIP = '192.168.4.121';
port = 5555;
u = udp(phoneIP , port - 1 , 'LocalPort' , port , 'InputBufferSize' , 1024);
fopen(u);

data = zeros(1,1);

k = 1;
while k < 200
    [msg,~] = fread(u,10);
    msgCell = strsplit(char(msg)',',');
    data(k,1:size(msgCell,2)) = str2double(msgCell);
```

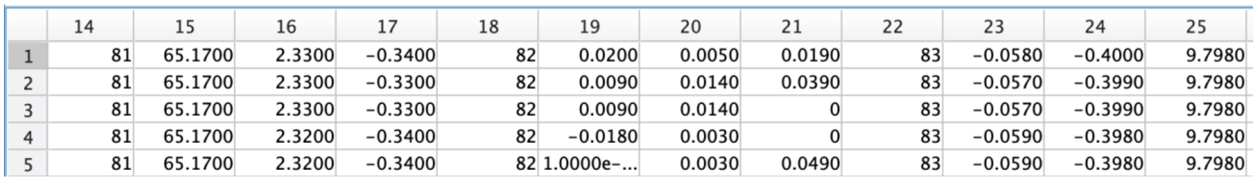


data(k,1:size(msgCell,2)) = str2double(msgCell);

```

    n_z = string(char(msg)');
    msg_c = char(msg)';
    fprintf(msg_c');
    fprintf('\n');

    k = k + 1;

end
% Close all instruments
fclose(instrfindall);

```



	1	2	3	4	5	6	7	8	9	10	11	12	13
1	2.2399e...	3	−0.0570	−0.4120	9.9890	4	0	−1.0000e...	−1.0000e...	5	−17.0630	11.2500	−46.7500
2	2.2399e...	3	−0.0960	−0.3930	9.9790	4	0	0	0	5	−17.3750	10.5630	−46.8750
3	2.2399e...	3	−0.0670	−0.4120	10.0080	4	−1.0000e...	0	−1.0000e...	5	−16.9380	10.8750	−47.4380
4	2.2399e...	3	−0.0570	−0.4020	10.0270	4	−1.0000e...	−1.0000e...	0	5	−16.9380	10.5630	−47.4380
5	2.2399e...	3	−0.1050	−0.4210	10.0080	4	0.0030	0	1.0000e−...	5	−16.3750	10.8750	−47.5630



	14	15	16	17	18	19	20	21	22	23	24	25
1	81	65.1700	2.3300	−0.3400	82	0.0200	0.0050	0.0190	83	−0.0580	−0.4000	9.7980
2	81	65.1700	2.3300	−0.3300	82	0.0090	0.0140	0.0390	83	−0.0570	−0.3990	9.7980
3	81	65.1700	2.3300	−0.3300	82	0.0090	0.0140	0	83	−0.0570	−0.3990	9.7980
4	81	65.1700	2.3200	−0.3400	82	−0.0180	0.0030	0	83	−0.0590	−0.3980	9.7980
5	81	65.1700	2.3200	−0.3400	82	1.0000e−...	0.0030	0.0490	83	−0.0590	−0.3980	9.7980

Problem 4: Extracting various portions of the decoded UDP stream and storing them in separate variables [3 points]

Modify the script to extract the accelerometer, gyroscope, magnetometer, lin. acc, orientation, and gravity in the current frame, store them in 6 separate 3X1 or 1X3 vectors, and display those vectors for the current frame in the command window. In addition, extract the timestamps.

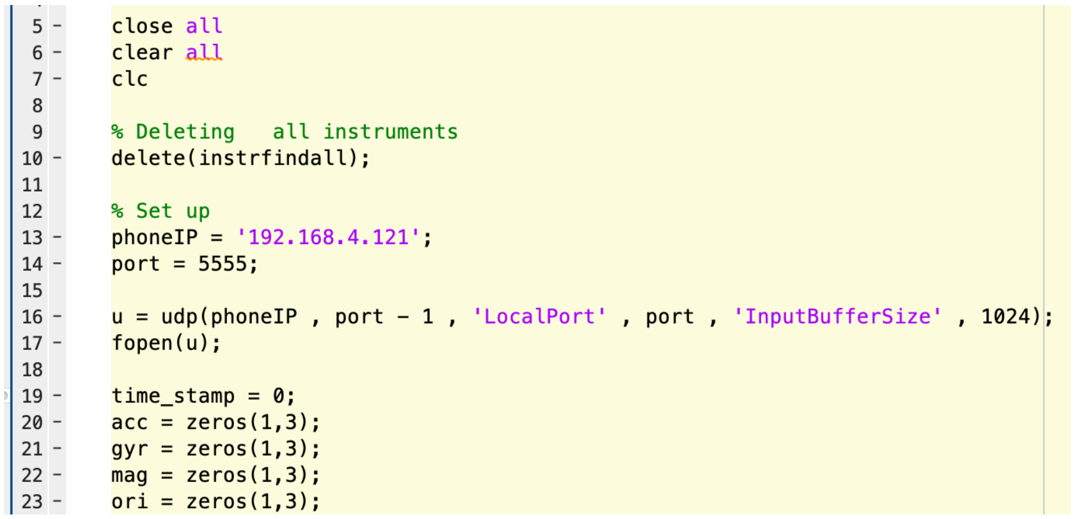
To get credit:

1. Provide the modified code below.
2. Provide 5 frames of the 6 separate vectors. For example, one frame of your output should look like this, all on a single line:

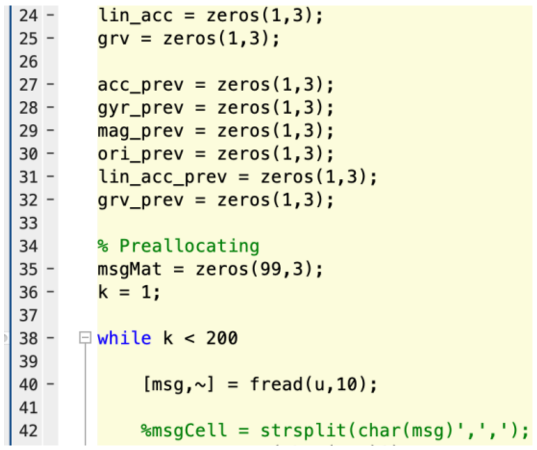
   Timestamp: 226841.2096- Acc: [2.383 2.444 9.193], Gyr: [0 0.001 -0.001], Mag: [27.199 -1.028 -57.262], Ori.: [288.437 -14.915 14.056], Lin. acc.: [-0.001 0.002 0.004], Grav.: [2.382 2.449 9.192]

   Hint: You can use your knowledge about strings, delimiters, splits, and data conversion. It is possible to perform the conversion in a single line for each entity using string and char data types.

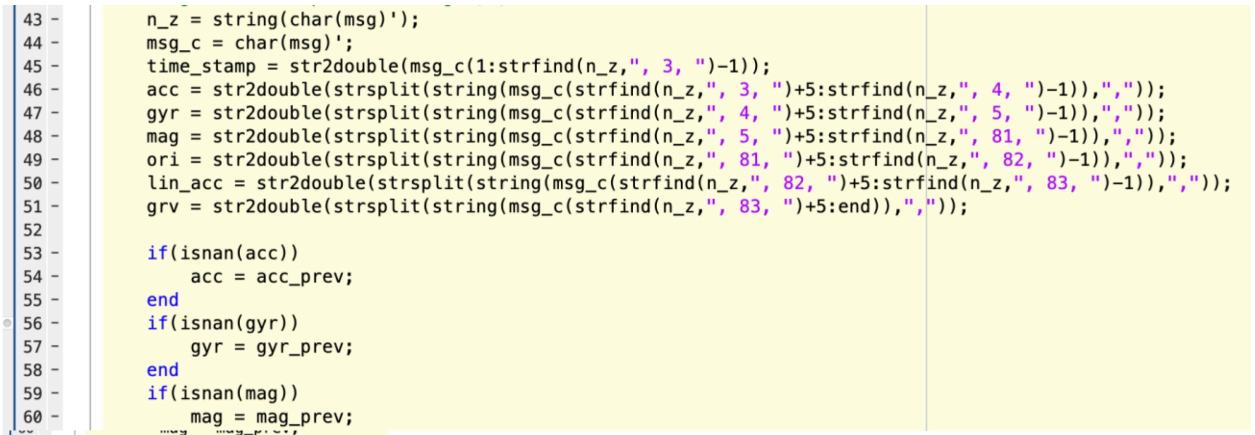
Answer:



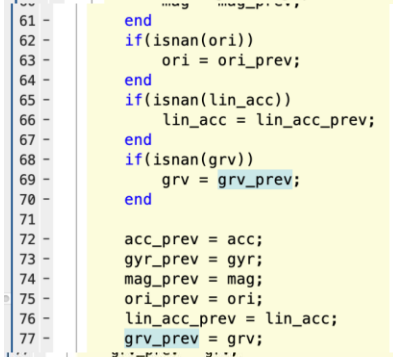
```
close all
clear all
clc

% Deleting   all instruments
delete(instrfindall);

% Set up
phoneIP = '192.168.4.121';
port = 5555;

u = udp(phoneIP , port - 1 , 'LocalPort' , port , 'InputBufferSize' , 1024);
fopen(u);

time_stamp = 0;
acc = zeros(1,3);
gyr = zeros(1,3);
mag = zeros(1,3);
ori = zeros(1,3);
```



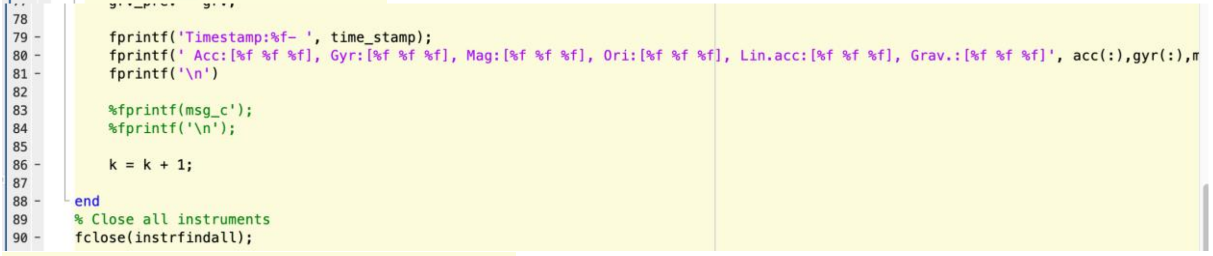
```
lin_acc = zeros(1,3);
grv = zeros(1,3);

acc_prev = zeros(1,3);
gyr_prev = zeros(1,3);
mag_prev = zeros(1,3);
ori_prev = zeros(1,3);
lin_acc_prev = zeros(1,3);
grv_prev = zeros(1,3);

% Preallocating
msgMat = zeros(99,3);
k = 1;

while k < 200

    [msg,~] = fread(u,10);

    %msgCell = strsplit(char(msg)',',');
```



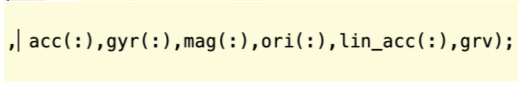
```
    n_z = string(char(msg)');
    msg_c = char(msg)';
    time_stamp = str2double(msg_c(1:strfind(n_z,", 3, ")-1));
    acc = str2double(strsplit(string(msg_c(strfind(n_z,", 3, ")+5:strfind(n_z,", 4, ")-1)),","));
    gyr = str2double(strsplit(string(msg_c(strfind(n_z,", 4, ")+5:strfind(n_z,", 5, ")-1)),","));
    mag = str2double(strsplit(string(msg_c(strfind(n_z,", 5, ")+5:strfind(n_z,", 81, ")-1)),","));
    ori = str2double(strsplit(string(msg_c(strfind(n_z,", 81, ")+5:strfind(n_z,", 82, ")-1)),","));
    lin_acc = str2double(strsplit(string(msg_c(strfind(n_z,", 82, ")+5:strfind(n_z,", 83, ")-1)),","));
    grv = str2double(strsplit(string(msg_c(strfind(n_z,", 83, ")+5:end)),","));

    if(isnan(acc))
        acc = acc_prev;
    end
    if(isnan(gyr))
        gyr = gyr_prev;
    end
    if(isnan(mag))
        mag = mag_prev;
```



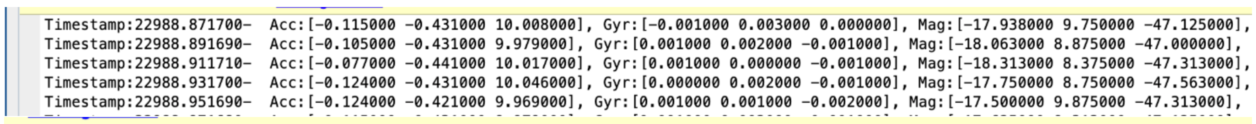
```
    end
    if(isnan(ori))
        ori = ori_prev;
    end
    if(isnan(lin_acc))
        lin_acc = lin_acc_prev;
    end
    if(isnan(grv))
        grv = grv_prev;
    end

    acc_prev = acc;
    gyr_prev = gyr;
    mag_prev = mag;
    ori_prev = ori;
    lin_acc_prev = lin_acc;
    grv_prev = grv;
```



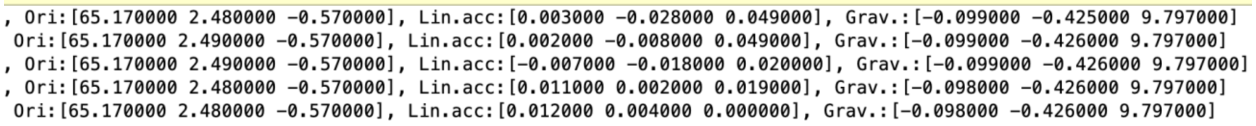
```

    fprintf('Timestamp:%f- ', time_stamp);
    fprintf(' Acc:[%f %f %f], Gyr:[%f %f %f], Mag:[%f %f %f], Ori:[%f %f %f], Lin.acc:[%f %f %f], Grav.:[%f %f %f]', acc(:),gyr(:),m
    fprintf('\n')

    %fprintf(msg_c');
    %fprintf('\n');

    k = k + 1;

end
% Close all instruments
fclose(instrfindall);
```



```
,| acc(:),gyr(:),mag(:),ori(:),lin_acc(:),grv);
```



```
Timestamp:22988.871700-  Acc:[-0.115000 -0.431000 10.008000], Gyr:[-0.001000 0.003000 0.000000], Mag:[-17.938000 9.750000 -47.125000],
Timestamp:22988.891690-  Acc:[-0.105000 -0.431000 9.979000], Gyr:[0.001000 0.002000 -0.001000], Mag:[-18.063000 8.875000 -47.000000],
Timestamp:22988.911710-  Acc:[-0.077000 -0.441000 10.017000], Gyr:[0.001000 0.000000 -0.001000], Mag:[-18.313000 8.375000 -47.313000],
Timestamp:22988.931700-  Acc:[-0.124000 -0.431000 10.046000], Gyr:[0.000000 0.002000 -0.001000], Mag:[-17.750000 8.750000 -47.563000],
Timestamp:22988.951690-  Acc:[-0.124000 -0.421000 9.969000], Gyr:[0.001000 0.001000 -0.002000], Mag:[-17.500000 9.875000 -47.313000],
```



```
, Ori:[65.170000 2.480000 -0.570000], Lin.acc:[0.003000 -0.028000 0.049000], Grav.:[-0.099000 -0.425000 9.797000]
 Ori:[65.170000 2.490000 -0.570000], Lin.acc:[0.002000 -0.008000 0.049000], Grav.:[-0.099000 -0.426000 9.797000]
, Ori:[65.170000 2.490000 -0.570000], Lin.acc:[-0.007000 -0.018000 0.020000], Grav.:[-0.099000 -0.426000 9.797000]
, Ori:[65.170000 2.480000 -0.570000], Lin.acc:[0.011000 0.002000 0.019000], Grav.:[-0.098000 -0.426000 9.797000]
 Ori:[65.170000 2.480000 -0.570000], Lin.acc:[0.012000 0.004000 0.000000], Grav.:[-0.098000 -0.426000 9.797000]
```

Problem 5: Using 9DoF Indirect Linear KF to track smartphone attitude in real-time [10 points]

Using the following link as a reference

https://www.mathworks.com/help/fusion/ug/Estimating-Orientation-Using-Inertial-Sensor-Fusion-and-MPU-9250.html

write a MATLAB script to display the orientation of the smartphone on the computer screen in real-time, using 9-axis indirect linear KF (AHRS filter). You can skip the calibration step as smartphone IMUs calibrate themselves from time-to-time as discussed in the lecture. Use the same noise parameters given in the link.

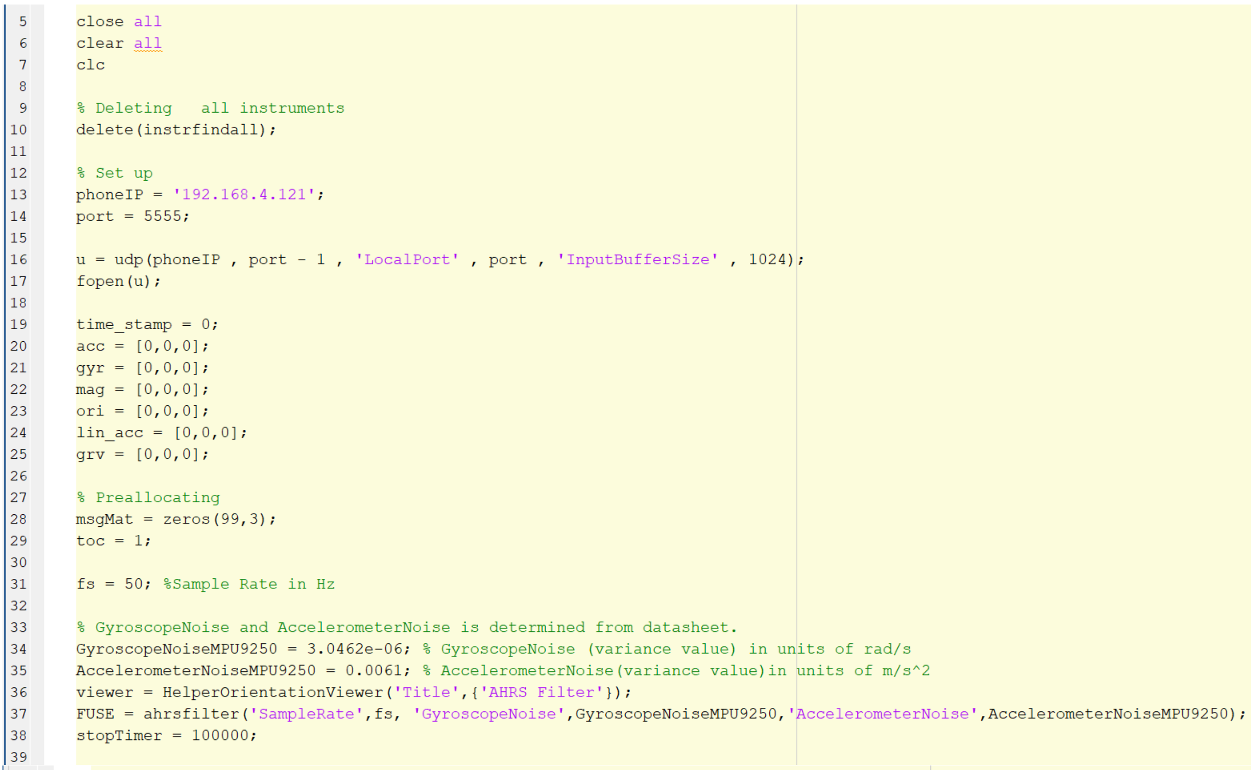
To get credit for this section:

1. Provide the entire code below in the space for the answer.
2. Record a video of yourself moving the smartphone and the corresponding graphic on the screen responding to those movements. Upload the video on Youtube (set the view option to “Unlisted”) and provide the video link below.

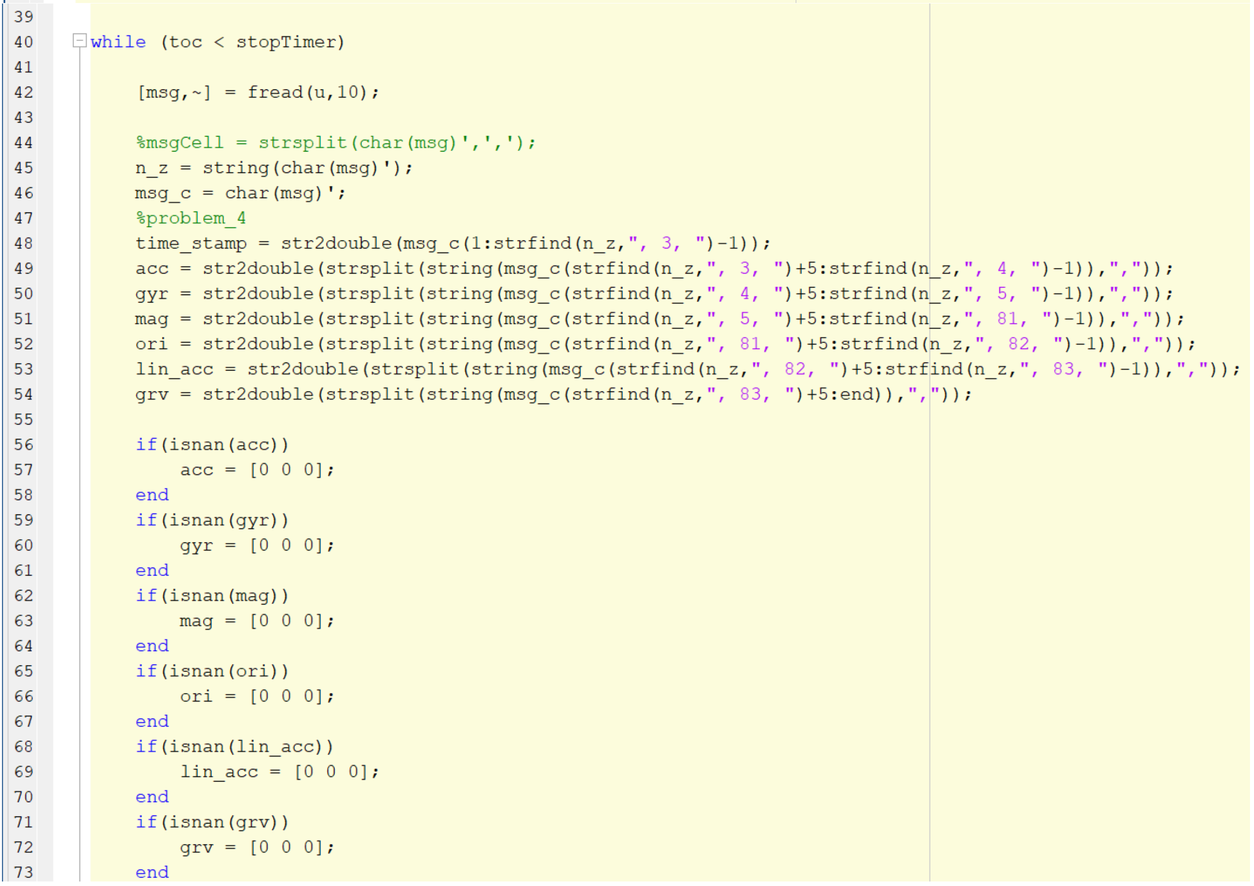
Note:

- You might observe that the smartphone does not send data at all some of the time and MATLAB registers those as NaN or missing value. You can either interpolate those values (nearest or moving average) or consider them as 0.
- The IMU sampling rate will not be exactly 50 Hz and will have some jitter. The KF should be able to handle that regardless.
- You will observe the orientation estimate drifting with time.

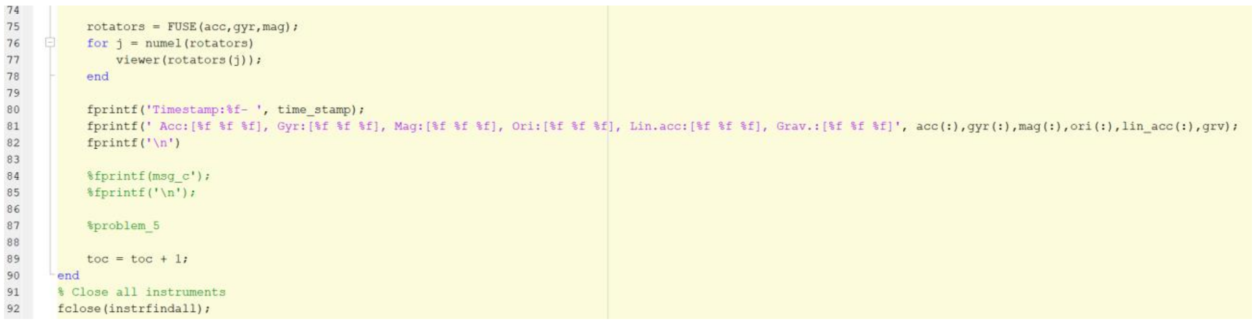
Answer:



```
close all
clear all
clc

% Deleting   all instruments
delete(instrfindall);

% Set up
phoneIP = '192.168.4.121';
port = 5555;

u = udp(phoneIP , port - 1 , 'LocalPort' , port , 'InputBufferSize' , 1024);
fopen(u);

time_stamp = 0;
acc = [0,0,0];
gyr = [0,0,0];
mag = [0,0,0];
ori = [0,0,0];
lin_acc = [0,0,0];
grv = [0,0,0];

% Preallocating
msgMat = zeros(99,3);
toc = 1;

fs = 50; %Sample Rate in Hz

% GyroscopeNoise and AccelerometerNoise is determined from datasheet.
GyroscopeNoiseMPU9250 = 3.0462e-06; % GyroscopeNoise (variance value) in units of rad/s
AccelerometerNoiseMPU9250 = 0.0061; % AccelerometerNoise(variance value)in units of m/s^2
viewer = HelperOrientationViewer('Title',{'AHRS Filter'});
FUSE = ahrsfilter('SampleRate',fs, 'GyroscopeNoise',GyroscopeNoiseMPU9250,'AccelerometerNoise',AccelerometerNoiseMPU9250);
stopTimer = 100000;

```



```

while (toc < stopTimer)

    [msg,~] = fread(u,10);

    %msgCell = strsplit(char(msg)',',');
    n_z = string(char(msg)');
    msg_c = char(msg)';
    %problem_4
    time_stamp = str2double(msg_c(1:strfind(n_z,", 3, ")-1));
    acc = str2double(strsplit(string(msg_c(strfind(n_z,", 3, ")+5:strfind(n_z,", 4, ")-1)),","));
    gyr = str2double(strsplit(string(msg_c(strfind(n_z,", 4, ")+5:strfind(n_z,", 5, ")-1)),","));
    mag = str2double(strsplit(string(msg_c(strfind(n_z,", 5, ")+5:strfind(n_z,", 81, ")-1)),","));
    ori = str2double(strsplit(string(msg_c(strfind(n_z,", 81, ")+5:strfind(n_z,", 82, ")-1)),","));
    lin_acc = str2double(strsplit(string(msg_c(strfind(n_z,", 82, ")+5:strfind(n_z,", 83, ")-1)),","));
    grv = str2double(strsplit(string(msg_c(strfind(n_z,", 83, ")+5:end)),","));

    if(isnan(acc))
        acc = [0 0 0];
    end
    if(isnan(gyr))
        gyr = [0 0 0];
    end
    if(isnan(mag))
        mag = [0 0 0];
    end
    if(isnan(ori))
        ori = [0 0 0];
    end
    if(isnan(lin_acc))
        lin_acc = [0 0 0];
    end
    if(isnan(grv))
        grv = [0 0 0];
    end
```



```

    rotators = FUSE(acc,gyr,mag);
    for j = numel(rotators)
        viewer(rotators(j));
    end

    fprintf('Timestamp:%f- ', time_stamp);
    fprintf(' Acc:[%f %f %f], Gyr:[%f %f %f], Mag:[%f %f %f], Ori:[%f %f %f], Lin.acc:[%f %f %f], Grav.:[%f %f %f]', acc(:),gyr(:),mag(:),ori(:),lin_acc(:),grv);
    fprintf('\n')

    %fprintf(msg_c');
    %fprintf('\n');

    %problem_5

    toc = toc + 1;
end
% Close all instruments
fclose(instrfindall);
```

https://youtu.be/2z2q0SAJUFU

Problem 6: Comparing attitude tracking error for various algorithms [20 points]

Using the following link as a reference

https://www.mathworks.com/help/fusion/ref/complementaryfilter-system-object.html

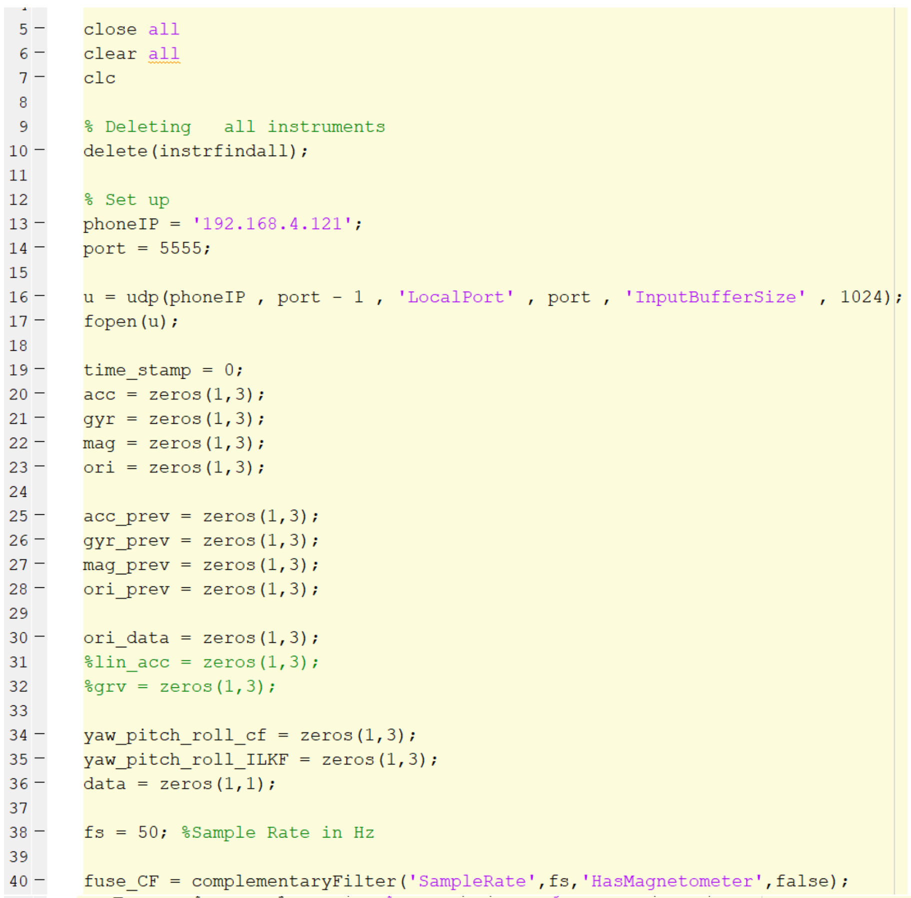
implement a 6DoF complementary filter (CF) that works in real-time like the Indirect LKF you implemented in the last section.

- Record the yaw, pitch, and roll for both ILKF, CF, and ground truth (orientation estimate from the phone, which is already yaw, pitch, and roll) in three separate matrices for smartphone motions of your choice for 20 seconds.
- Perform necessary coordinate transforms (using rotation matrices) to make sure all three state estimates are in the same reference frame.

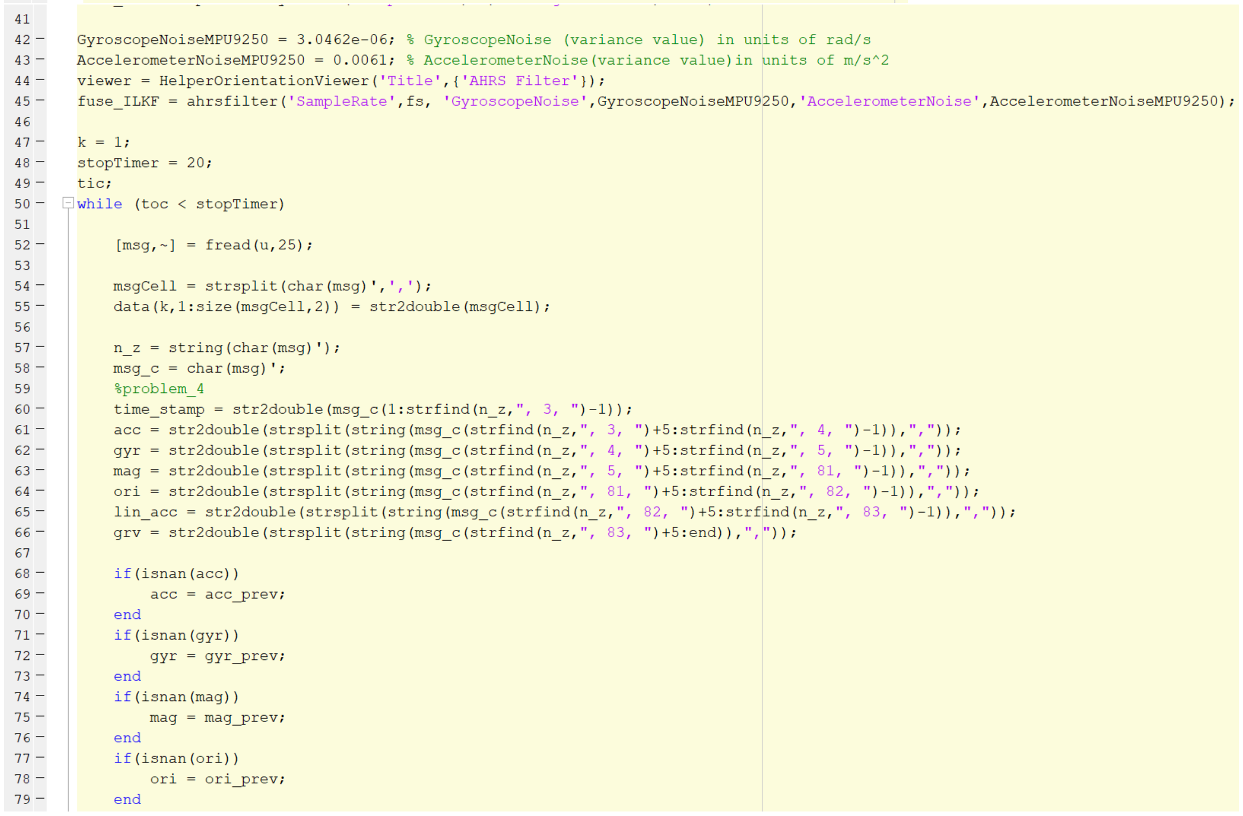
To get credit for this problem:

1. Provide the code for the implemented complementary filter below. You DO NOT need to record a video for this filter in action as you did inthe pprevious problem.
2. On the same figure, plot the roll of the smartphone estimated by ILKF, CF, and ground truth filter versus time. Provide the figure below.
3. Calculate the root-mean-squared error (RMSE) of the roll, pitch, and yaw for ILKF and CF against the ground truth orientation for all samples in the 20-second frame. On three subplots:
    a. Plot the roll RMSE of ILKF and CF
    b. Plot the pitch RMSE of ILKF and CF
    c. Plot the yaw RMSE of ILKF and CF
    d. Provide the plots below in the space for the answer.
4. Which filter provides comparatively lower RMSE and why do you think the “better filter” provides a more accurate state estimate? Explain in a non-mathematical manner.

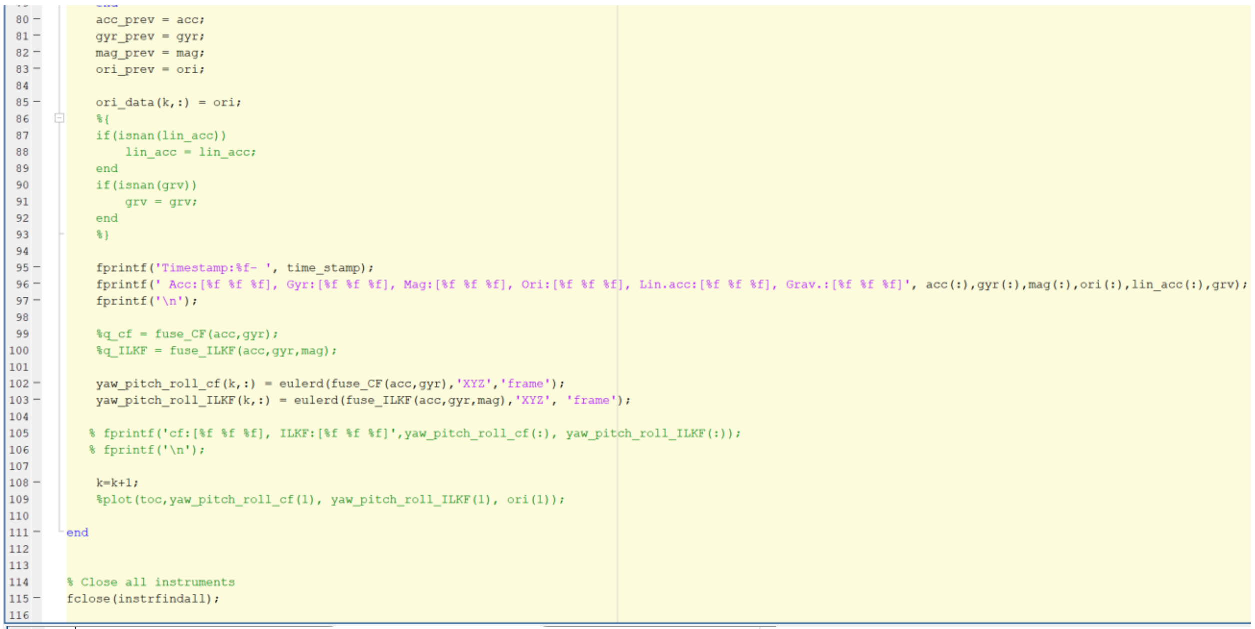
Answer:



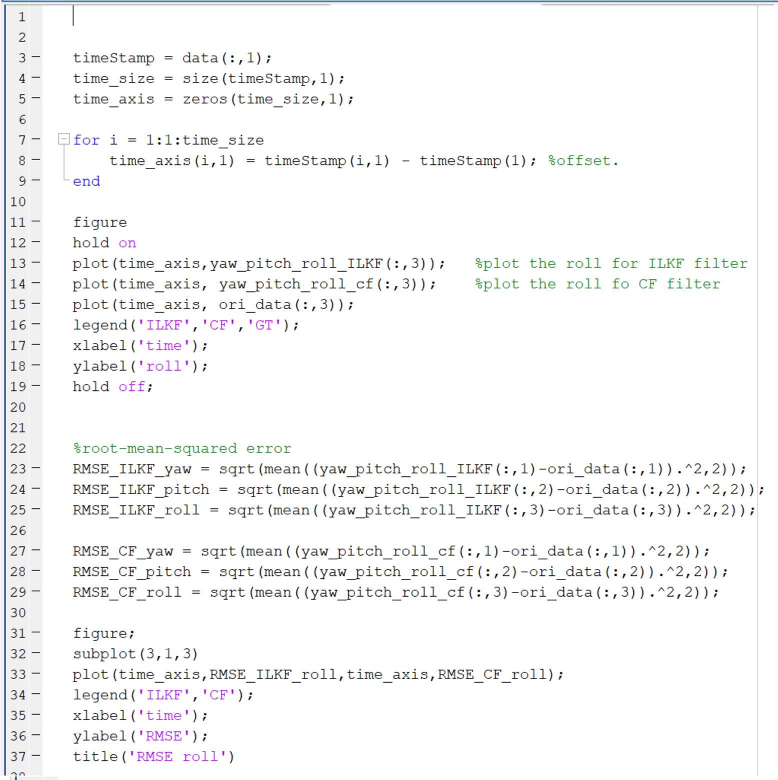
```matlab
close all
clear all
clc

% Deleting   all instruments
delete(instrfindall);

% Set up
phoneIP = '192.168.4.121';
port = 5555;

u = udp(phoneIP , port - 1 , 'LocalPort' , port , 'InputBufferSize' , 1024);
fopen(u);

time_stamp = 0;
acc = zeros(1,3);
gyr = zeros(1,3);
mag = zeros(1,3);
ori = zeros(1,3);

acc_prev = zeros(1,3);
gyr_prev = zeros(1,3);
mag_prev = zeros(1,3);
ori_prev = zeros(1,3);

ori_data = zeros(1,3);
%lin_acc = zeros(1,3);
%grv = zeros(1,3);

yaw_pitch_roll_cf = zeros(1,3);
yaw_pitch_roll_ILKF = zeros(1,3);
data = zeros(1,1);

fs = 50; %Sample Rate in Hz

fuse_CF = complementaryFilter('SampleRate',fs,'HasMagnetometer',false);
```



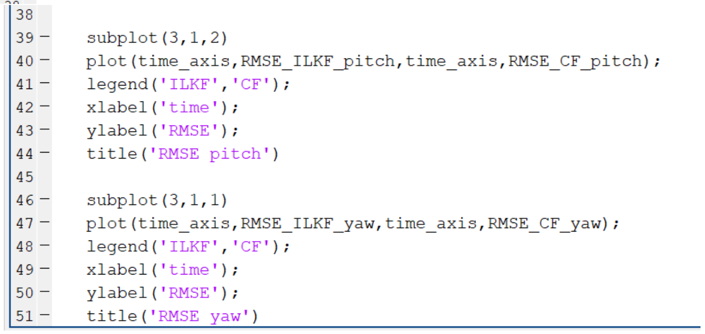
```matlab

GyroscopeNoiseMPU9250 = 3.0462e-06; % GyroscopeNoise (variance value) in units of rad/s
AccelerometerNoiseMPU9250 = 0.0061; % AccelerometerNoise(variance value)in units of m/s^2
viewer = HelperOrientationViewer('Title',{'AHRS Filter'});
fuse_ILKF = ahrsfilter('SampleRate',fs, 'GyroscopeNoise',GyroscopeNoiseMPU9250,'AccelerometerNoise',AccelerometerNoiseMPU9250);

k = 1;
stopTimer = 20;
tic;
while (toc < stopTimer)

    [msg,~] = fread(u,25);

    msgCell = strsplit(char(msg)',',');
    data(k,1:size(msgCell,2)) = str2double(msgCell);

    n_z = string(char(msg)');
    msg_c = char(msg)';
    %problem_4
    time_stamp = str2double(msg_c(1:strfind(n_z,", 3, ")-1));
    acc = str2double(strsplit(string(msg_c(strfind(n_z,", 3, ")+5:strfind(n_z,", 4, ")-1)),","));
    gyr = str2double(strsplit(string(msg_c(strfind(n_z,", 4, ")+5:strfind(n_z,", 5, ")-1)),","));
    mag = str2double(strsplit(string(msg_c(strfind(n_z,", 5, ")+5:strfind(n_z,", 81, ")-1)),","));
    ori = str2double(strsplit(string(msg_c(strfind(n_z,", 81, ")+5:strfind(n_z,", 82, ")-1)),","));
    lin_acc = str2double(strsplit(string(msg_c(strfind(n_z,", 82, ")+5:strfind(n_z,", 83, ")-1)),","));
    grv = str2double(strsplit(string(msg_c(strfind(n_z,", 83, ")+5:end)),","));

    if(isnan(acc))
        acc = acc_prev;
    end
    if(isnan(gyr))
        gyr = gyr_prev;
    end
    if(isnan(mag))
        mag = mag_prev;
    end
    if(isnan(ori))
        ori = ori_prev;
    end
```



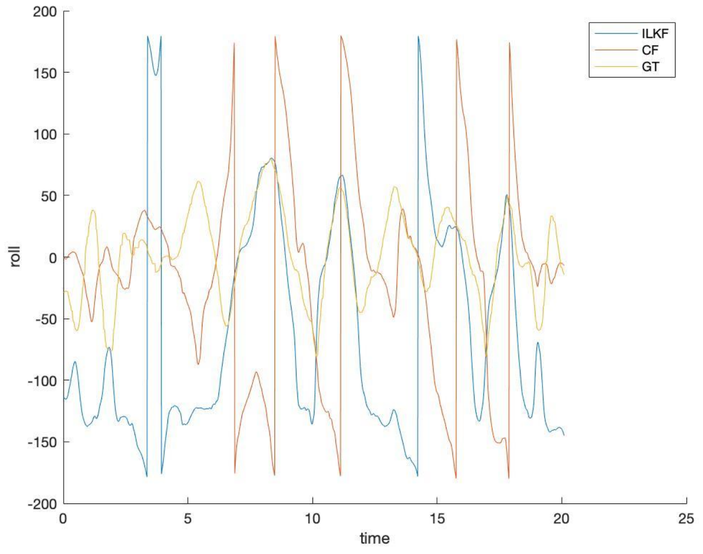
```
    acc_prev = acc;
    gyr_prev = gyr;
    mag_prev = mag;
    ori_prev = ori;

    ori_data(k,:) = ori;
    %{
    if(isnan(lin_acc))
        lin_acc = lin_acc;
    end
    if(isnan(grv))
        grv = grv;
    end
    %}

    fprintf('Timestamp:%f- ', time_stamp);
    fprintf(' Acc:[%f %f %f], Gyr:[%f %f %f], Mag:[%f %f %f], Ori:[%f %f %f], Lin.acc:[%f %f %f], Grav.:[%f %f %f]', acc(:),gyr(:),mag(:),ori(:),lin_acc(:),grv);
    fprintf('\n');

    %q_cf = fuse_CF(acc,gyr);
    %q_ILKF = fuse_ILKF(acc,gyr,mag);

    yaw_pitch_roll_cf(k,:) = eulerd(fuse_CF(acc,gyr),'XYZ','frame');
    yaw_pitch_roll_ILKF(k,:) = eulerd(fuse_ILKF(acc,gyr,mag),'XYZ', 'frame');

   % fprintf('cf:[%f %f %f], ILKF:[%f %f %f]',yaw_pitch_roll_cf(:), yaw_pitch_roll_ILKF(:));
   % fprintf('\n');

    k=k+1;
    %plot(toc,yaw_pitch_roll_cf(1), yaw_pitch_roll_ILKF(1), ori(1));

end


% Close all instruments
fclose(instrfindall);

```

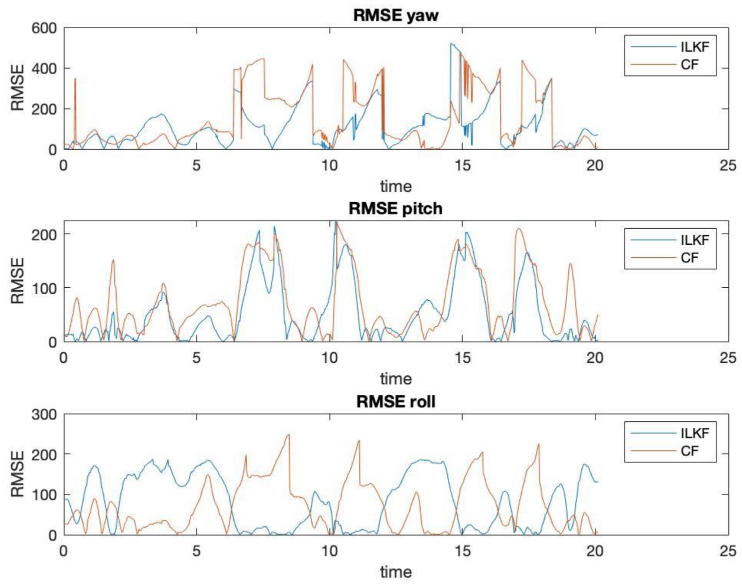


```


timeStamp = data(:,1);
time_size = size(timeStamp,1);
time_axis = zeros(time_size,1);

for i = 1:1:time_size
    time_axis(i,1) = timeStamp(i,1) - timeStamp(1); %offset.
end

figure
hold on
plot(time_axis,yaw_pitch_roll_ILKF(:,3));   %plot the roll for ILKF filter
plot(time_axis, yaw_pitch_roll_cf(:,3));    %plot the roll fo CF filter
plot(time_axis, ori_data(:,3));
legend('ILKF','CF','GT');
xlabel('time');
ylabel('roll');
hold off;


%root-mean-squared error
RMSE_ILKF_yaw = sqrt(mean((yaw_pitch_roll_ILKF(:,1)-ori_data(:,1)).^2,2));
RMSE_ILKF_pitch = sqrt(mean((yaw_pitch_roll_ILKF(:,2)-ori_data(:,2)).^2,2));
RMSE_ILKF_roll = sqrt(mean((yaw_pitch_roll_ILKF(:,3)-ori_data(:,3)).^2,2));

RMSE_CF_yaw = sqrt(mean((yaw_pitch_roll_cf(:,1)-ori_data(:,1)).^2,2));
RMSE_CF_pitch = sqrt(mean((yaw_pitch_roll_cf(:,2)-ori_data(:,2)).^2,2));
RMSE_CF_roll = sqrt(mean((yaw_pitch_roll_cf(:,3)-ori_data(:,3)).^2,2));

figure;
subplot(3,1,3)
plot(time_axis,RMSE_ILKF_roll,time_axis,RMSE_CF_roll);
legend('ILKF','CF');
xlabel('time');
ylabel('RMSE');
title('RMSE roll')
```



```

subplot(3,1,2)
plot(time_axis,RMSE_ILKF_pitch,time_axis,RMSE_CF_pitch);
legend('ILKF','CF');
xlabel('time');
ylabel('RMSE');
title('RMSE pitch')

subplot(3,1,1)
plot(time_axis,RMSE_ILKF_yaw,time_axis,RMSE_CF_yaw);
legend('ILKF','CF');
xlabel('time');
ylabel('RMSE');
title('RMSE yaw')
```





Based on the RMSE plot, it cannot tell which filter has lower RMSE. However, I think the CF filter is the better filter because it can only take accelerator and gyroscope to calculate the orientation. It has less calculation and implementation compared to the ILKF filter.

Problem 7: Pose estimation using step detection + Weinberg SLE [24 points]

From https://lopsi.weebly.com/teaching.html download the MATLAB code for pedestrian dead-reckoning from the link provided as well as the tutorial.

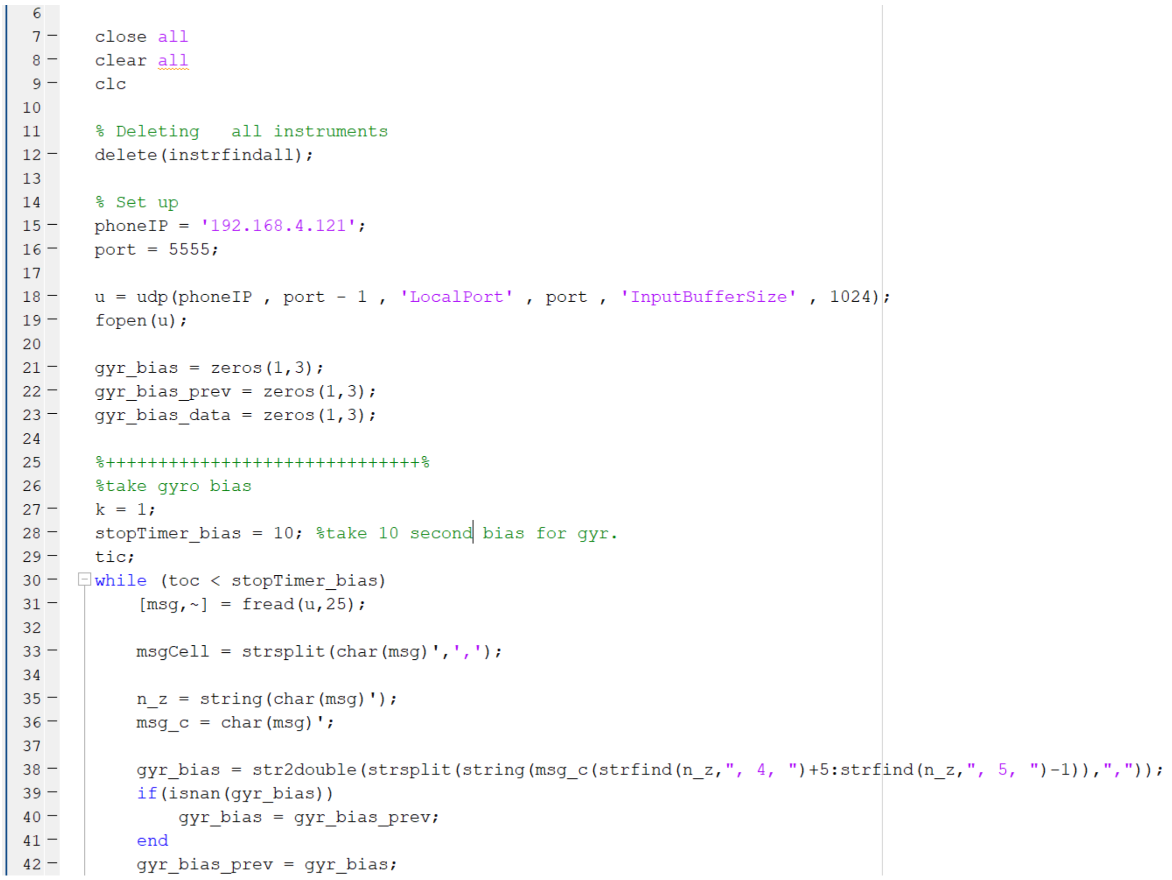
You will be implementing “Practice 3”.

- Record 30 seconds of accelerometer and gyroscope data (as well as timestamps) to your computer using the smartphone app you used earlier while walking in a predictable motion around a room. Make sure to cover enough rectangular space around the room and keep your smartphone static while walking. Make sure to record a video of you walking so that you can reference it later.
- Use accelerometer-based step detection and Weinberg SLE to calculate the position (not in real-time) in the x and y direction (the horizontal plane of the room) in the NED coordinate frame.

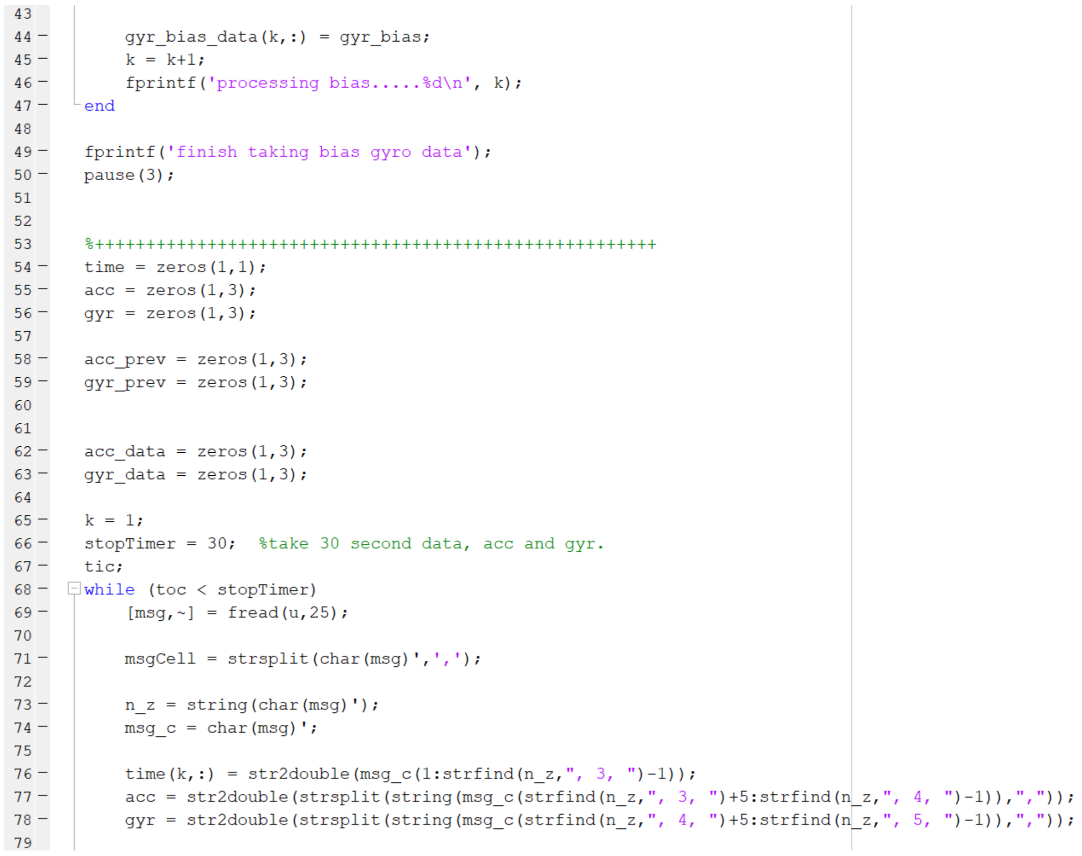
To get credit for this section:

1. Provide the MATLAB script to record 30 seconds of accelerometer and gyroscope data.
2. Provide the MATLAB script performing the position estimation.
3. Show the plot of position in the NED coordinate frame.
4. Qualitatively explain any difference you see in your walking trajectory from the video and the estimated trajectory.

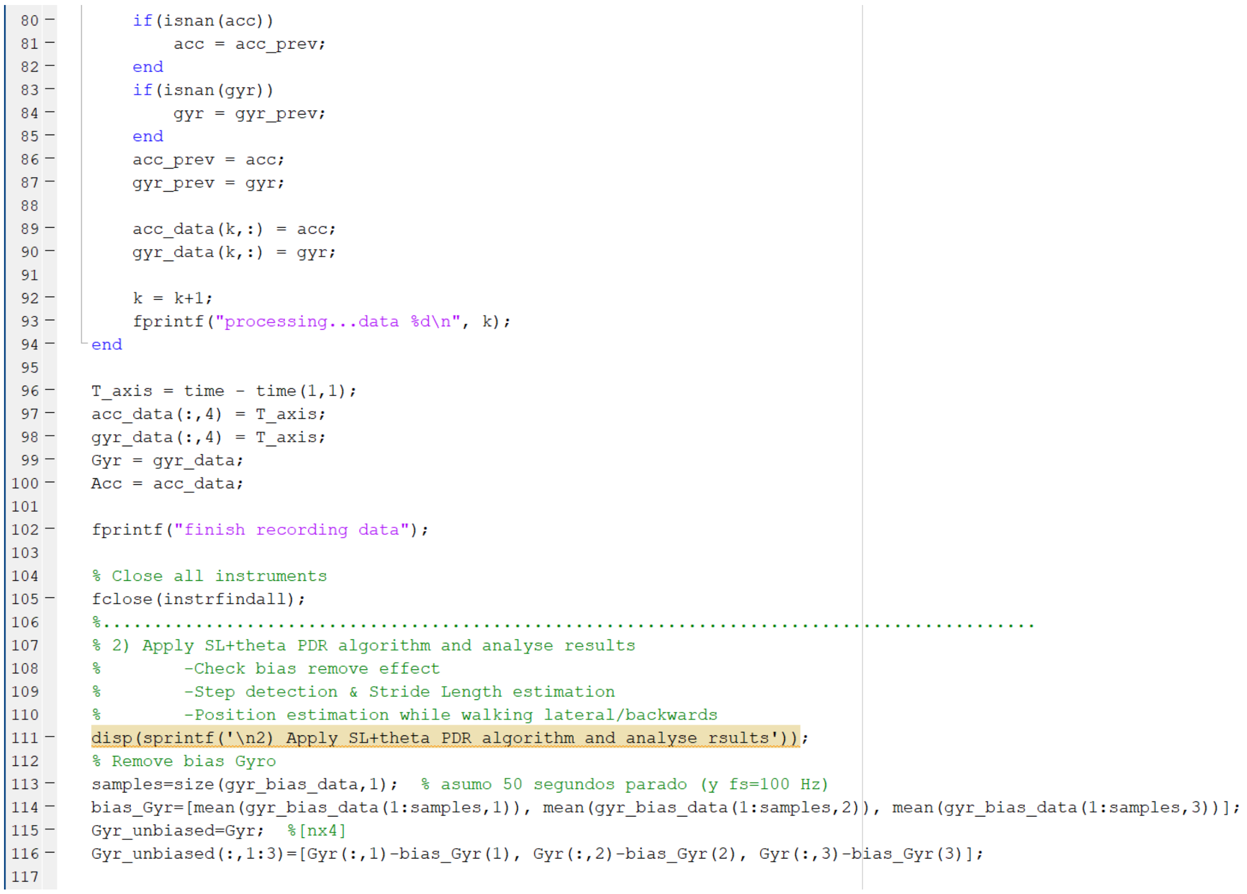
Answer:



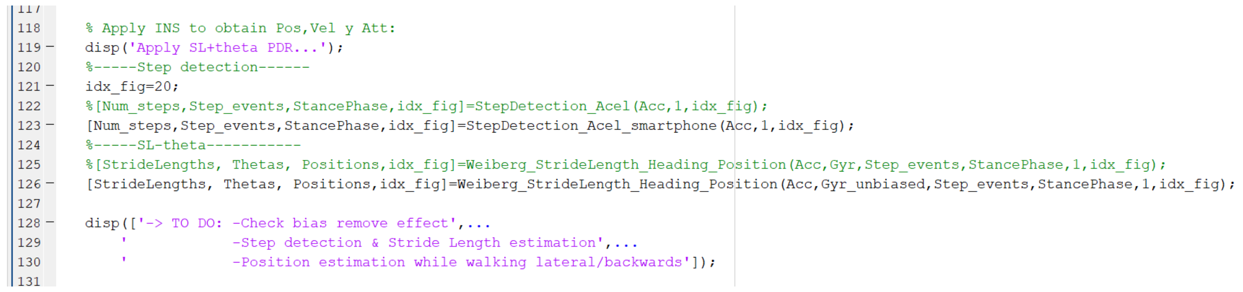
```matlab

close all
clear all
clc

% Deleting   all instruments
delete(instrfindall);

% Set up
phoneIP = '192.168.4.121';
port = 5555;

u = udp(phoneIP , port - 1 , 'LocalPort' , port , 'InputBufferSize' , 1024);
fopen(u);

gyr_bias = zeros(1,3);
gyr_bias_prev = zeros(1,3);
gyr_bias_data = zeros(1,3);

%++++++++++++++++++++++++++++++%
%take gyro bias
k = 1;
stopTimer_bias = 10; %take 10 second bias for gyr.
tic;
while (toc < stopTimer_bias)
    [msg,~] = fread(u,25);

    msgCell = strsplit(char(msg)',',');

    n_z = string(char(msg)');
    msg_c = char(msg)';

    gyr_bias = str2double(strsplit(string(msg_c(strfind(n_z,", 4, ")+5:strfind(n_z,", 5, ")-1)),","));
    if(isnan(gyr_bias))
        gyr_bias = gyr_bias_prev;
    end
    gyr_bias_prev = gyr_bias;
```



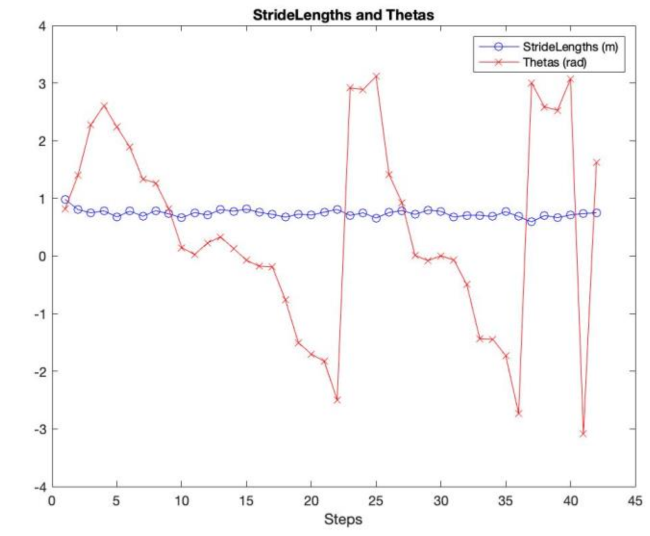
```matlab

    gyr_bias_data(k,:) = gyr_bias;
    k = k+1;
    fprintf('processing bias.....%d\n', k);
end

fprintf('finish taking bias gyro data');
pause(3);


%+++++++++++++++++++++++++++++++++++++++++++++++++++++++
time = zeros(1,1);
acc = zeros(1,3);
gyr = zeros(1,3);

acc_prev = zeros(1,3);
gyr_prev = zeros(1,3);


acc_data = zeros(1,3);
gyr_data = zeros(1,3);

k = 1;
stopTimer = 30;  %take 30 second data, acc and gyr.
tic;
while (toc < stopTimer)
    [msg,~] = fread(u,25);

    msgCell = strsplit(char(msg)',',');

    n_z = string(char(msg)');
    msg_c = char(msg)';

    time(k,:) = str2double(msg_c(1:strfind(n_z,", 3, ")-1));
    acc = str2double(strsplit(string(msg_c(strfind(n_z,", 3, ")+5:strfind(n_z,", 4, ")-1)),","));
    gyr = str2double(strsplit(string(msg_c(strfind(n_z,", 4, ")+5:strfind(n_z,", 5, ")-1)),","));

```

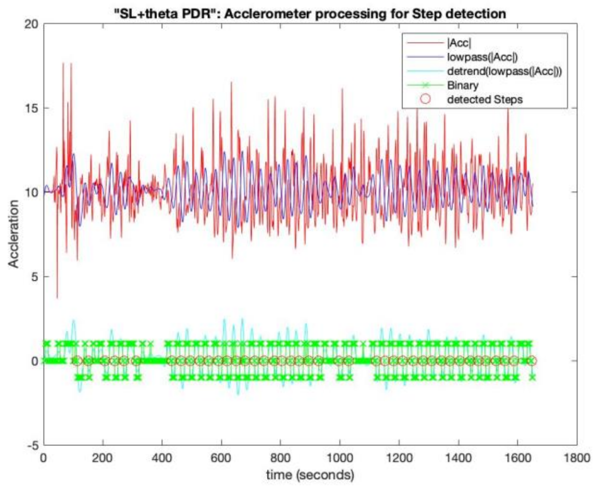


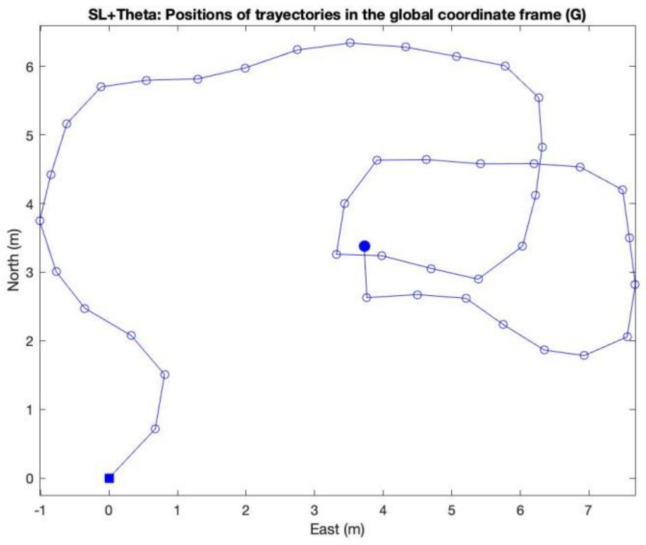
```
    if(isnan(acc))
        acc = acc_prev;
    end
    if(isnan(gyr))
        gyr = gyr_prev;
    end
    acc_prev = acc;
    gyr_prev = gyr;

    acc_data(k,:) = acc;
    gyr_data(k,:) = gyr;

    k = k+1;
    fprintf("processing...data %d\n", k);
end

T_axis = time - time(1,1);
acc_data(:,4) = T_axis;
gyr_data(:,4) = T_axis;
Gyr = gyr_data;
Acc = acc_data;

fprintf("finish recording data");

% Close all instruments
fclose(instrfindall);
%..................................................................................
% 2) Apply SL+theta PDR algorithm and analyse results
%        -Check bias remove effect
%        -Step detection & Stride Length estimation
%        -Position estimation while walking lateral/backwards
disp(sprintf('\n2) Apply SL+theta PDR algorithm and analyse rsults'));
% Remove bias Gyro
samples=size(gyr_bias_data,1);  % asumo 50 segundos parado (y fs=100 Hz)
bias_Gyr=[mean(gyr_bias_data(1:samples,1)), mean(gyr_bias_data(1:samples,2)), mean(gyr_bias_data(1:samples,3))];
Gyr_unbiased=Gyr;  %[nx4]
Gyr_unbiased(:,1:3)=[Gyr(:,1)-bias_Gyr(1), Gyr(:,2)-bias_Gyr(2), Gyr(:,3)-bias_Gyr(3)];
```



```
% Apply INS to obtain Pos,Vel y Att:
disp('Apply SL+theta PDR...');
%-----Step detection------
idx_fig=20;
%[Num_steps,Step_events,StancePhase,idx_fig]=StepDetection_Acel(Acc,1,idx_fig);
[Num_steps,Step_events,StancePhase,idx_fig]=StepDetection_Acel_smartphone(Acc,1,idx_fig);
%-----SL-theta-----------
%[StrideLengths, Thetas, Positions,idx_fig]=Weiberg_StrideLength_Heading_Position(Acc,Gyr,Step_events,StancePhase,1,idx_fig);
[StrideLengths, Thetas, Positions,idx_fig]=Weiberg_StrideLength_Heading_Position(Acc,Gyr_unbiased,Step_events,StancePhase,1,idx_fig);

disp(['-> TO DO: -Check bias remove effect',...
    '           -Step detection & Stride Length estimation',...
    '           -Position estimation while walking lateral/backwards']);
```


StrideLengths and Thetas
StrideLengths (m)
Thetas (rad)
Steps


"SL+theta PDR": Acclerometer processing for Step detection
|Acc|
lowpass(|Acc|)
detrend(lowpass(|Acc|))
Binary
detected Steps
Accleration
time (seconds)


SL+Theta: Positions of trayectories in the global coordinate frame (G)
North (m)
East (m)

The estimated trajectory has predicted my real walking trajectory correctly. I started moving out of my room and went to the living room. Then I walked the rectangular track in the living room. However, the estimated trajectory in figure 1 was not accurate because it was not a full rectangular. It could be affected by signal processing because the phone sometimes transits the loss data to the computer. It might need more time to estimate the trajectory when walking on the same trajectory.

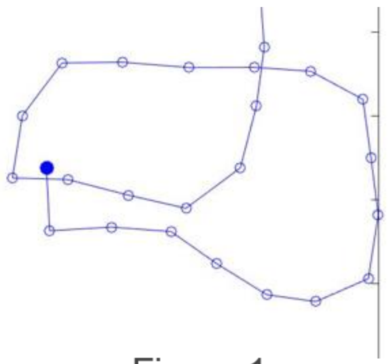


Figure 1

Problem 8: Miscellaneous Conceptual Questions [50 points]

1. A^3 introduced a complementary opportunistic calibration technique, using one or more sensors to calibrate each other. Briefly outline the opportunistic IMU calibration algorithm and provide two benefits of using the algorithm over classical IMU calibration techniques presented in the lecture. [5 points]

   Ref: Zhou, Pengfei, Mo Li, and Guobin Shen. "Use it free: Instantly knowing your phone attitude." Proceedings of the 20th annual international conference on Mobile computing and networking. 2014.

Answer:

It includes three types of sensor which are compass, accelerometer, and gyroscope. Most of the time, the gyroscope provides accurate attitude estimation. However, because of the base errors from the previous attitude estimation, the instant (short period) attitude estimation of gyroscope is not accurate. The opportunistic IMU calibration technique can provide a good reference for attitude estimation if the gyroscope is not accurate. The opportunistic calibration technique uses the compass and gravity (accelerometer) outputs which have positive calibration opportunity to estimate the short period attitude changes. It compares the attitude change derived from the compass and gravity. If the compass and gravity have the same change of phone attitude estimation, then the compass and gravity make an accurate attitude estimation. And the attitude estimation from the compass and gravity would replace the current phone attitude estimation. Therefore, the errors from the gyroscope can be calibrated, and it provides a high accuracy of attitude estimation.

Benefits:

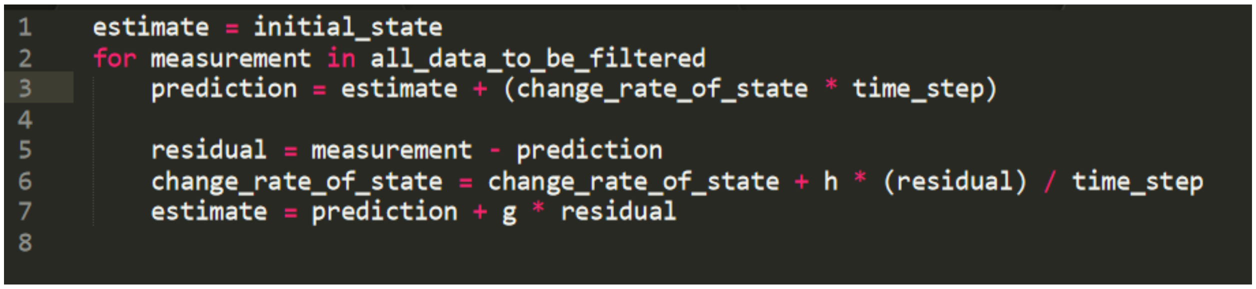
1. It provides a long time run high accuracy of attitude estimation. The opportunistic calibration technique calibrates the drift errors from the gyroscope.
2. The opportunistic calibration can provide a good reference if the gyroscope could not measure accurately especially the phone has higher moving motion.

2. A simplified version of the stochastic Kalman filter is the g-h or alpha-beta filter.
   a. Write down the measurement and correction blocks for the alpha-beta filter [3 points]
   b. What effect does altering g and h have on the system output [2 points]?
   c. Design a g-h filter for obtaining the attitude from gyroscope and accelerometer readings in the Euler coordinate system. You can reference complementary filters [5 points]

   Ref: Labbe, Roger. "Kalman and bayesian filters in python."

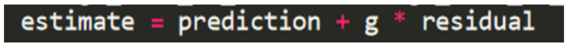
Answer:

a.



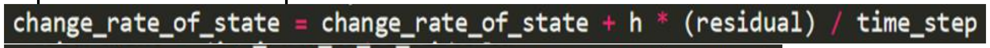
```
estimate = initial_state
for measurement in all_data_to_be_filtered
    prediction = estimate + (change_rate_of_state * time_step)

    residual = measurement - prediction
    change_rate_of_state = change_rate_of_state + h * (residual) / time_step
    estimate = prediction + g * residual

```

b. Altering g will impact the estimation significantly. From this equation, we can see that if g is equal to 1, the estimation would be close to the measurement which is the prediction + residual. If g is smaller, the estimation would be approaching to the prediction. Therefore, the g will significantly impact on the estimation.

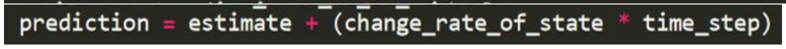


```
estimate = prediction + g * residual
```

Altering h will affect the change rate of state variable. Higher h results in a higher change rate of state, with the higher change rate, the change of prediction would be higher. Otherwise, lower h would have lower change rate and the change of prediction. Therefore, the h will significantly impact the next state prediction.

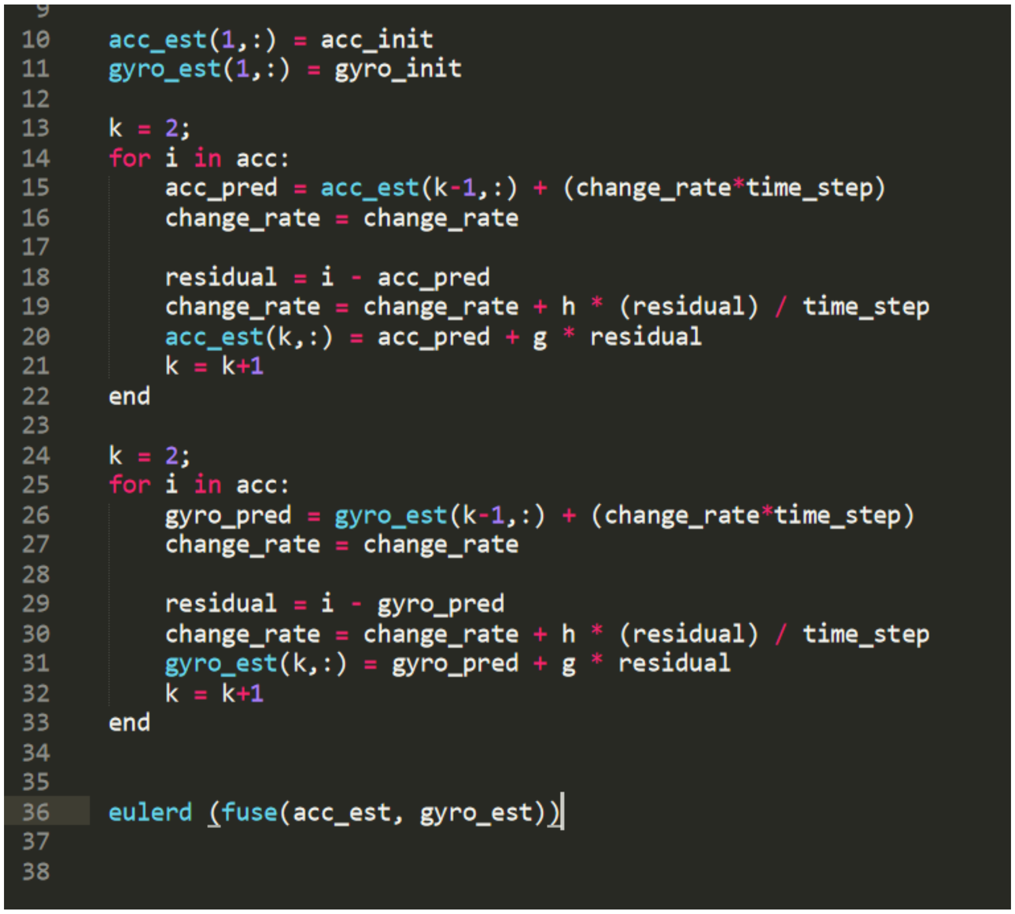


```
change_rate_of_state = change_rate_of_state + h * (residual) / time_step
```



```
prediction = estimate + (change_rate_of_state * time_step)
```

c.



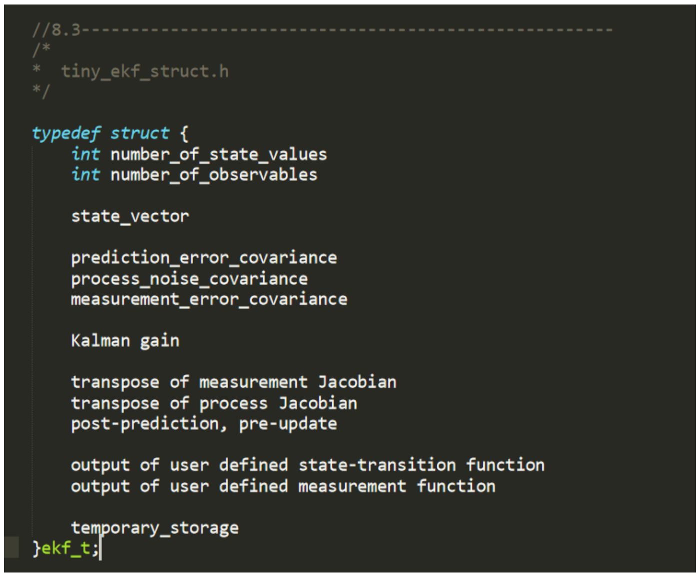
```
acc_est(1,:) = acc_init
gyro_est(1,:) = gyro_init

k = 2;
for i in acc:
    acc_pred = acc_est(k-1,:) + (change_rate*time_step)
    change_rate = change_rate

    residual = i - acc_pred
    change_rate = change_rate + h * (residual) / time_step
    acc_est(k,:) = acc_pred + g * residual
    k = k+1
end

k = 2;
for i in acc:
    gyro_pred = gyro_est(k-1,:) + (change_rate*time_step)
    change_rate = change_rate

    residual = i - gyro_pred
    change_rate = change_rate + h * (residual) / time_step
    gyro_est(k,:) = gyro_pred + g * residual
    k = k+1
end


eulerd (fuse(acc_est, gyro_est))
```

Acc and gyro go through the g-h filter and then go to the Euler coordinate function to get the result.

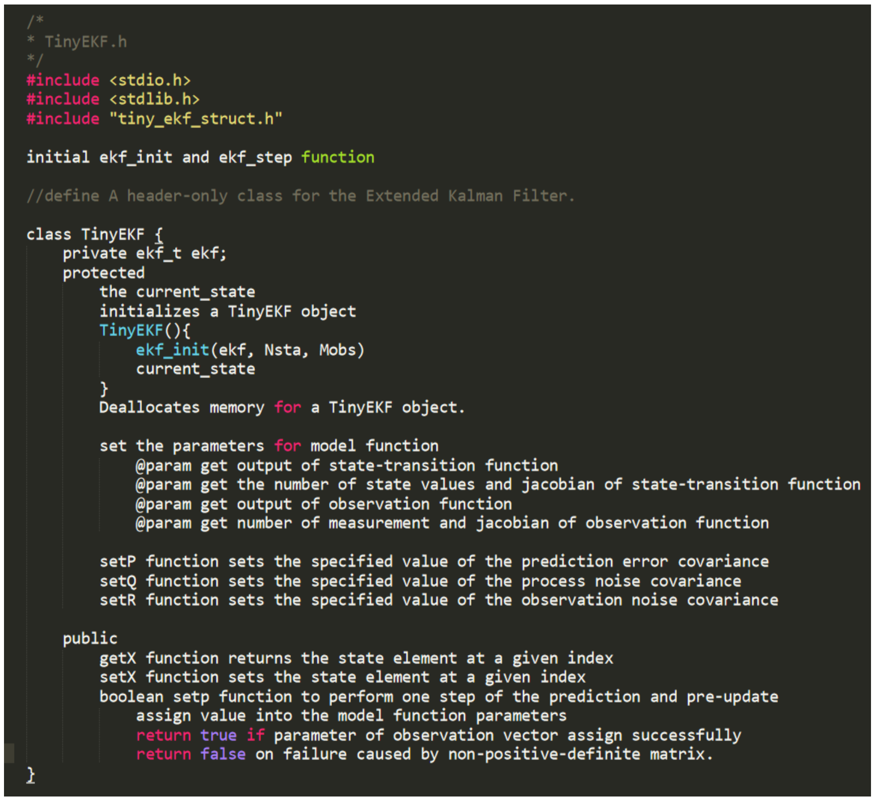
3. Write pseudocode in C for implementing an extended Kalman filter to fuse IMU and GPS data on resource-constrained platform such as Arduino or STM32 Nucleo for pose estimation. [20 points]

   You may find the following link useful: https://github.com/simondlevy/TinyEKF

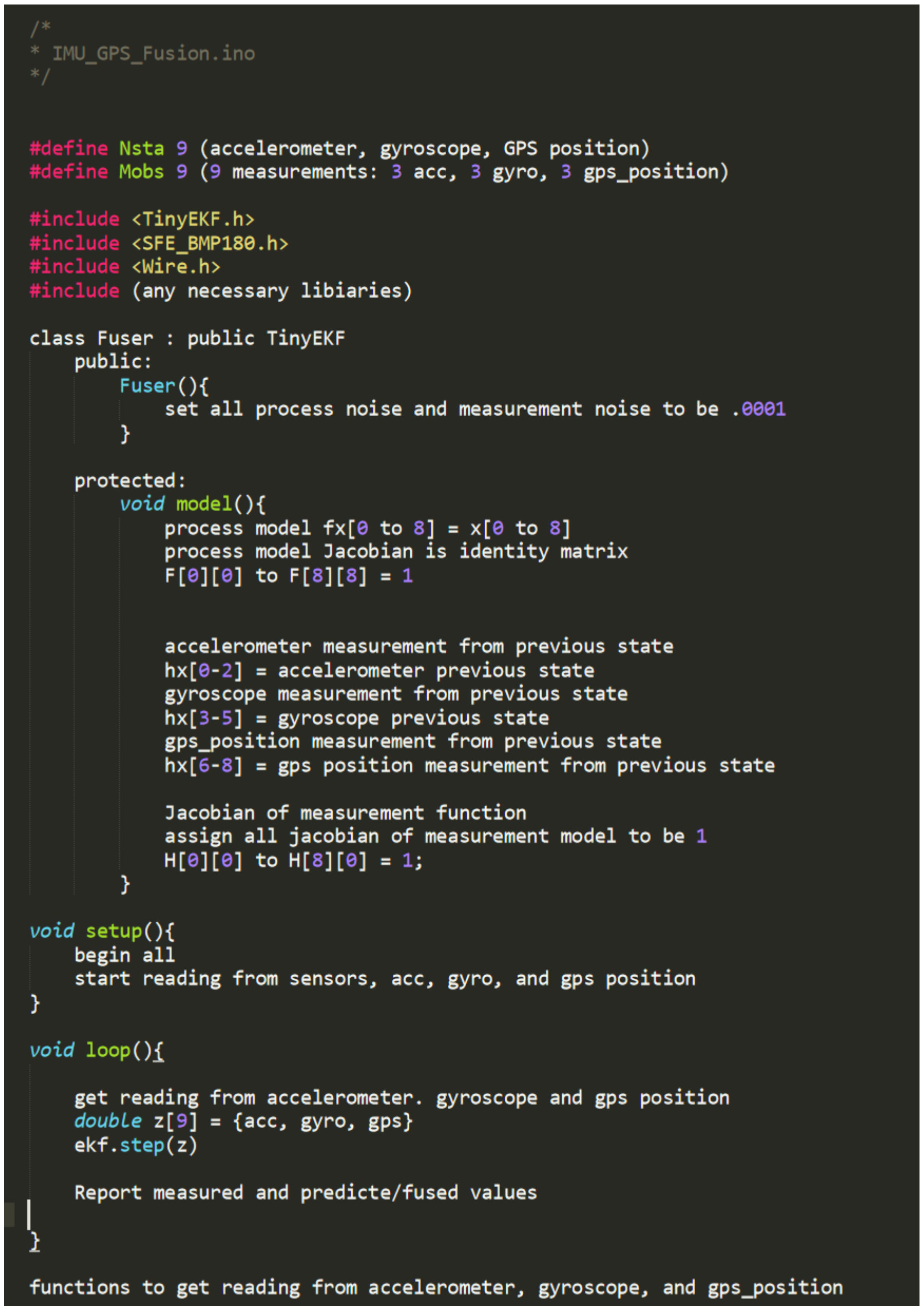
Answer:



```
//8.3---------------------------------------------------
/*
 *  tiny_ekf_struct.h
 */

typedef struct {
    int number_of_state_values
    int number_of_observables

    state_vector

    prediction_error_covariance
    process_noise_covariance
    measurement_error_covariance

    Kalman gain

    transpose of measurement Jacobian
    transpose of process Jacobian
    post-prediction, pre-update

    output of user defined state-transition function
    output of user defined measurement function

    temporary_storage
}ekf_t;
```



```
/*
* TinyEKF.h
*/
#include <stdio.h>
#include <stdlib.h>
#include "tiny_ekf_struct.h"

initial ekf_init and ekf_step function

//define A header-only class for the Extended Kalman Filter.

class TinyEKF {
    private ekf_t ekf;
    protected
        the current_state
        initializes a TinyEKF object
        TinyEKF(){
            ekf_init(ekf, Nsta, Mobs)
            current_state
        }
        Deallocates memory for a TinyEKF object.

        set the parameters for model function
            @param get output of state-transition function
            @param get the number of state values and jacobian of state-transition function
            @param get output of observation function
            @param get number of measurement and jacobian of observation function

        setP function sets the specified value of the prediction error covariance
        setQ function sets the specified value of the process noise covariance
        setR function sets the specified value of the observation noise covariance

    public
        getX function returns the state element at a given index
        setX function sets the state element at a given index
        boolean setp function to perform one step of the prediction and pre-update
            assign value into the model function parameters
            return true if parameter of observation vector assign successfully
            return false on failure caused by non-positive-definite matrix.
}
```



```
/*
* IMU_GPS_Fusion.ino
*/


#define Nsta 9 (accelerometer, gyroscope, GPS position)
#define Mobs 9 (9 measurements: 3 acc, 3 gyro, 3 gps_position)

#include <TinyEKF.h>
#include <SFE_BMP180.h>
#include <Wire.h>
#include (any necessary libiaries)

class Fuser : public TinyEKF
    public:
        Fuser(){
            set all process noise and measurement noise to be .0001
        }

    protected:
        void model(){
            process model fx[0 to 8] = x[0 to 8]
            process model Jacobian is identity matrix
            F[0][0] to F[8][8] = 1


            accelerometer measurement from previous state
            hx[0-2] = accelerometer previous state
            gyroscope measurement from previous state
            hx[3-5] = gyroscope previous state
            gps_position measurement from previous state
            hx[6-8] = gps position measurement from previous state

            Jacobian of measurement function
            assign all jacobian of measurement model to be 1
            H[0][0] to H[8][0] = 1;
        }

void setup(){
    begin all
    start reading from sensors, acc, gyro, and gps position
}

void loop(){

    get reading from accelerometer. gyroscope and gps position
    double z[9] = {acc, gyro, gps}
    ekf.step(z)

    Report measured and predicte/fused values

}

functions to get reading from accelerometer, gyroscope, and gps_position
```

4. The lecture lists several key AI-driven dead-reckoning algorithms.
    a. List three key challenges for developing learning-enabled dead-reckoning systems [3 points]
    b. Why do most AI dead-reckoning systems regress velocity and not position? [2 points]
    c. Some of the seminal works in AI-based dead-reckoning include the use of time convolutional neural networks as the target network architecture. Why do TCNs, in general, provide superior regression/classification performance over LSTMs, CNNs or LSTM-CNN? [5 points]

       You may use the following paper as a reference:
          Lea, Colin, et al. "Temporal convolutional networks for action segmentation and detection." proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.
    d. Suppose, you want to implement one of the data-driven dead-reckoning algorithms provided in the slide for real-time dead-reckoning on a resource-constrained embedded device like Arduino or STM32Nucleo. Outline the steps (including any specific programming tools) you would take to translate the trained model (which will usually be a Pytorch or Tensorflow model) to C code such that it can be used in real-time on the deployed edge device. Keep track of the system constraints of the microcontroller while making decisions. [5 points]

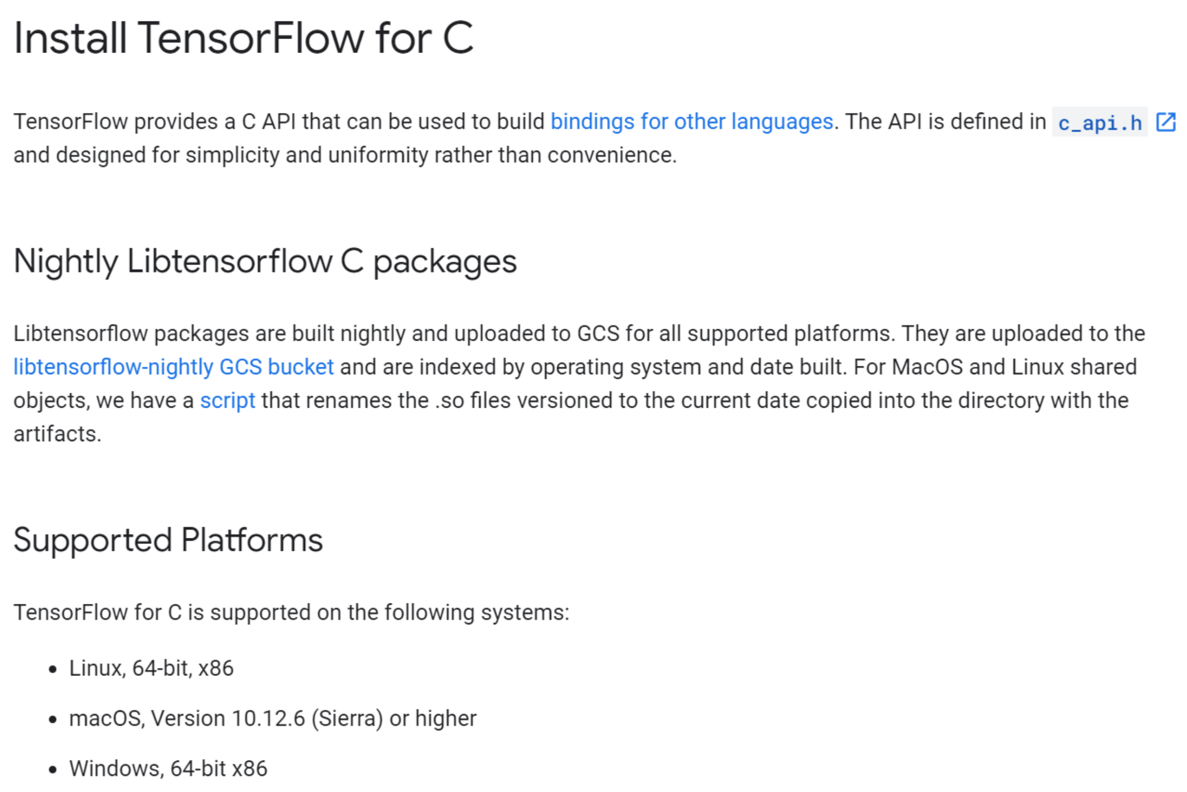
Answer:
a. Three key challenges:
    1. power consumption is one of the challenges because the deep learning algorithms require a lot of usage of power and energy.
    2. The system needs to train a model and spend time to collect data. Different environments would need to collect different data. It takes a lot of time to process.
    3. constrained in the usage of memory and CPU.

b. The Dead reckoning is subject to cumulative errors. And the dead reckoning systems use the data from IMU senor instead of GPS. Therefore, it is better to regress velocity instead of position.

c. The TCNs uses a hierarchy of convolutions to capture long-range temporal patterns. It would allow to capture complex patterns including compositions, action durations, robust to time-delays. Therefore, the TCNs has superior regression/classification performance than other NN algorithms.

d. I found out that the TensorFlow has library for C code, which is called Nightly Libtensorflow C packages. However, it is only available for Linux, macOS, and

Windows. It could be a challenge to implement into embedded device since it is not supposed.



Install TensorFlow for C

TensorFlow provides a C API that can be used to build bindings for other languages. The API is defined in `c_api.h` and designed for simplicity and uniformity rather than convenience.

Nightly Libtensorflow C packages

Libtensorflow packages are built nightly and uploaded to GCS for all supported platforms. They are uploaded to the libtensorflow-nightly GCS bucket and are indexed by operating system and date built. For MacOS and Linux shared objects, we have a script that renames the .so files versioned to the current date copied into the directory with the artifacts.

Supported Platforms

TensorFlow for C is supported on the following systems:

- Linux, 64-bit, x86
- macOS, Version 10.12.6 (Sierra) or higher
- Windows, 64-bit x86

In my opinion, the most popular models that can be used in embedded device is TensorFlow Lite. We can train our model with TensorFlow Lite and implement the trained model into the embedded device. TensorFlow Lite is a good model for embedded device and other limited microcontrollers.