MTE-546 Multi-sensor Data Fusion
Final Exam

Release time: Thursday, April 10, 2025 12:00 pm
Deadline Thursday, April 18, 2025 11:59 pm

Mechanical and Mechatronics Engineering Department
University of Waterloo, **Instructor: Arash Arami**

Instruction:

The following questions are the final take-home exam for the Winter 2025 MTE 546 Multi-sensor Data Fusion offering. Each student is expected to complete the required analysis, simulation, and discussion for each question. Ensure that all assumptions and conclusions are explained with engineering judgment. If you use ideas from other papers, you need to cite them in your exam report properly. Using another student's solutions or communication with third parties will constitute an academic offense and be addressed according to departmental guidelines. In addition, any communication with someone other than the course instructor will be viewed as an academic offense under UW Police 71.

Make sure all plots are legible (use fonts of 12 or larger), numbered, have a caption and their axis are defined and labeled properly.

If you feel a required assumption is missing, make that assumption, state it clearly in your solution, and continue with it.

The problem set and its accompanied data should not be shared with others.

3 Problems, one week

Problem		Mark	Page
1		**35 (+10)**	**2-4**
2		**35 (+10)**	**5-9**
3		**30 (+10)**	**10-12**
Total:		**100(+30)**	

Problem 1: Troubleshooting an Existing Sensor Model

You are working at a company that designs pressure sensors for monitoring the flow of a hydraulic system. You have recently released a beta version of your sensor to a client, and they have been testing the sensor for their application. The client states that they are "having problems with the consistency of the sensors", but they have not yet provided further clarification of the exact problem. They indicate that the sensors will "need to perform better in order to continue with the next purchase order". You are tasked with investigating and resolving the client's requests.

The sensor works by measuring the deformation of a thin plate due to the flow of fluid across the element. Two sensor elements on the plate produce voltage measurements, $\boldsymbol{v} = [v_1, v_2]^T \in \mathbb{R}^2$, which are then used to compute the output, $z \in \mathbb{R}$.

The current sensor model is:

$$\hat{z} = 0.017v_1 + 0.042v_2 + 0.0015 + \eta \qquad (0)$$
$$\eta \sim \mathcal{N}(0, 0.0005)$$

The sensor model was created using data from the following test:

- The flow of fluid through the system is driven by two inputs, $\boldsymbol{q}(t)$, that are controlled to follow a triangle wave chirp signal with an amplitude of 0.3 from 0.1 to 1 Hz over a period of 20 sec

$$\tau(t) = f_0 t + \frac{1}{2}\frac{(f_T - f_0)}{T}t^2$$
$$f_0 = 0.1\ Hz, \qquad f_T = 1\ Hz, \qquad T = 20\ sec$$
$$\boldsymbol{q}(t) = \begin{bmatrix} q_1(t) \\ q_2(t) \end{bmatrix} = 0.3 \begin{bmatrix} \text{tri}(\tau(t) + 0.25) \\ \text{tri}(\tau(t) + 0.50) \end{bmatrix}$$

where the equation of a triangle waveform is:

$$\text{tri}(x) = 2|2(x - \text{floor}(x + 0.5))| - 1$$

You do not need to use or implement these saw-shape signals, and those are mentioned for your information.

- The desired response between the sensor measurements and the controls can be computed analytically as a linear function

$$z = -0.0081q_1 + 0.0589q_2 - 0.000625$$

This test was performed in-house before sending the sensors to the client. The client was also asked to repeat this test and send their data back to you.

Attached is a dataset containing the test results for 2 of the sensor units returned by the customer. There are 4 comma-separated value (CSV) files corresponding to the in-house and client tests for each sensor. The first row of each file contains the headers for each column and the second row contains the units.

Part 1: Initial Analysis

Plot the desired and estimated outputs of the sensor from inhouse_sensor_21.csv as two timeseries overlaid on the same plot.

Plot a scatterplot of the desired and estimated outputs where each point corresponds to a pair of desired outputs in the x-axis, and estimated outputs in the y-axis at the same timestep.

- **Show the points for inhouse_sensor_21.csv in one color, and the points for client_sensor_21.csv in another**
- **Plot the line where the points should lie if the model is perfect**

Inspect the estimated outputs compared to the desired outputs and examine the quality of the model. **State 2 observations about the data and model from the plot**.

Part 2A: Independent Model Development

Given the client's recent complaints, you have been tasked with independently redesigning the model.

Design a new model for the sensors using linear regression. Note that the model should quantify both the outputs and uncertainty of the estimate.

- **Describe any preprocessing steps**
- **Describe the method used to obtain the model**
- **Justify why you used the method that you chose**
- **State the model equations and parameters**

Use the combined data from inhouse_sensor_21.csv and client_sensor_21.csv to fit the model. **Do not use the client_sensor_22.csv or client_sensor_23.csv for training since they will be for validation.**

Part 2B: Model Generalizability and Model 2

The sensor model you designed needs to generalize to the other sensors that the customer has currently installed on their machine, for which there is limited data. Use the following procedure, called bootstrapping, to evaluate how well the linear regression coefficients generalize to other models.

1. Use bootstrapping to find the distribution of coefficients
    a. For 100 iterations:
        i. Draw a random subsample, with replacement, of 1000 timesteps from the combined data from inhouse_sensor_21.csv and client_sensor_21.csv.
        ii. Fit the model using your method from Part 2A and the subsamples
        iii. Save the coefficients of the model
    b. **Plot a histogram for each coefficient of your model using the 100 values for each coefficient. Use subplots as necessary to reduce the number of figures**
    c. Compute the mean coefficient vector and covariance matrix of coefficients computed over the 100 values of the coefficient vector

2. Fit a model (similar regressors to part 2A) to the data combined from client_sensor_22.csv

1. Compute the distance between the coefficients from sensor 22 Compute the Mahalanobis distance between the coefficients from sensor 22 (Step 2) and the mean coefficient vector (Step 1c). This distance quantifies the distance between an observed point, and the mean of a Gaussian distribution in units of standard deviations. The Mahalanobis distance between a point, $\boldsymbol{x}$, and a multivariate Gaussian distribution, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is:

$$d(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})}$$

   a. **State the Mahalanobis distance**
   b. **State whether the obtained distance is evidence for or against the generalizability of the model**

Part 3: Model Comparison

Compare your model from Part 2A with the existing model with the model (equation 0) using the following metrics on the data from client_sensor_22.csv:

- **Average error**
- **Variance of error**
- **Root Mean Squared Error**
- **Coefficient of Determination (R^2)**

State whether your model should be used in the next production batch of sensors and justify your reasoning.
Compute the estimated outputs for client_sensor_23.csv and save the outputs to a file named outputs_client_sensor_23.csv. The results will be evaluated using a script so place the outputs in the first column with no headers.

Note: The top 3 performances on the **client_sensor_23 in terms of RMS error will receive a 10% bonus to their final exam mark.**

Problem 2. EKF-Based State Estimation for the Improbability Drive

The *Heart of Gold* spaceship uses an *Improbability Drive* that moves the ship by selecting highly improbable outcomes. As the navigation computer of the spaceship has broken, they pick a UW engineer to develop an alternative state estimator by fusing their motion models and sensors. Your task is implementing an **Extended Kalman Filter (EKF)** to estimate the spaceship's state based on noisy, improbability-tuned measurements.

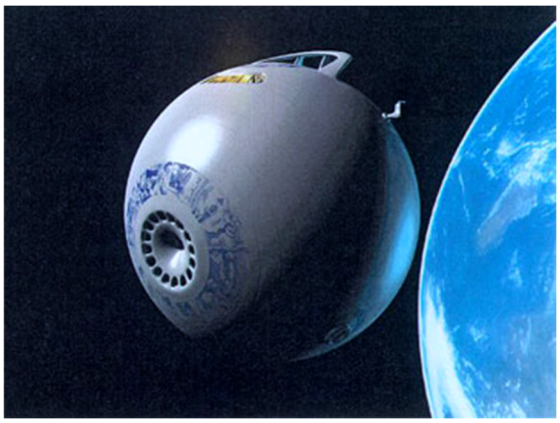

Figure 1. Heart of Gold passing by the blue planet

Background on Improbability Drive:

The improbability ($\mathcal{I}$) is defined using an inverse function of probability (P) of an event (A):

$$Improb(A) = \mathcal{I}(A) = \frac{1}{P(A)} - 1 \tag{1}$$

This is an interesting concept used by intergalactic engineers to develop the engine of Heart of Gold. Improbability Drive can release vast energy hidden in improbable events as their low probability can generate and unbounded improbability, leading to an unbounded energy source for a short period of time.
For instance, if the chance of picking up an intergalactic hitchhiker by a spaceship close to the Earth is 1 to 10^6 ($P(A)$=0.000001), such an unlikely event can generate an improbability of $\mathcal{I}(A) = 10^6 - 1$.
Imagine if that be injected as the engine force!
The problem is often the fact that when Improbability Drive is triggered, there will be an increase of uncertainty on the generated engine force, the underlying dynamics/mechanics of the spaceship, sensors and even some people believe the Universe. So be ready for a bit more uncertainty here and there!
Important: According to what is explained, after the trigger of Improbability Drive, it usually takes up to 4 seconds for the dynamics of spaceship to settle. This is even longer for the sensors (up to 8 seconds to settle).
You are tasked here to implement and tune an Extended Kalman Filter to track the state of Heart of Gold. The details are provided after the system description.

System description

For these you are provided with a descriptive mathematical model below, actual sensors measurement of the spaceship for 150 samples (150 seconds of measurements,

positional information in meter and rad), 150 seconds of Improbability Drive engine force (the one that Marvin[1] believes to be correct), also the occurrence of events and their Probabilities (can be used to calculate the improbability $\mathcal{I}$) as well as 100 samples of actual states (for tunning purposes, as well as computation of errors and interpretations).

The spaceship's state (X) consists of:

- Position in 3D space: *x, y, z*
- Forward velocity: *v*
- Orientation in terms of Euler angles: ϕ, θ, ψ

1. **Nonlinear Motion Model:**

$$X_{i+1} = \begin{bmatrix} x_{i+1} \\ y_{i+1} \\ z_{i+1} \\ v_{i+1} \\ \phi_{i+1} \\ \theta_{i+1} \\ \psi_{i+1} \end{bmatrix} = g(X_i, u_i) + q_i \tag{2}$$

$$u_i = \begin{bmatrix} a_i \\ \omega_{x,i} \\ \omega_{y,i} \\ \omega_{z,i} \end{bmatrix} \tag{3}$$

where *g* is a nonlinear function mapping previous states X_i and Improbability Drive control action u_i (including the linear acceleration a_i and angular velocity Ω_i vector) into the new states X_{i+1}. q_i is a 7-dimensional noise from a Gaussian distribution with a covariance of *Q*. $q_i \sim \mathcal{N}(0, Q)$

The first three state transitions can be found using below equation, when not considering the noise:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ z_{i+1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + R(\phi_i, \theta_i, \psi_i) \begin{bmatrix} v_i \Delta T + 0.5 a_i \Delta T^2 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

where $a_i = u_i(1)$ (first dimension of control action of Improbability Drive), ΔT is the time step (Sampling time), and *R* is a rotation matrix with description below:

[1] Ignore Marvin but use the Improbability Drive's *u* as the input to your model! Well, Marvin is a robot with a brain approximately the size of the Universe, but people usually ignore him. So feel free to do so!

$$R(\phi_i, \theta_i, \psi_i) = \begin{pmatrix} C_{\phi_i}C_{\theta_i} & C_{\phi_i}S_{\theta_i}S_{\psi_i} - S_{\phi_i}C_{\psi_i} & C_{\phi_i}S_{\theta_i}C_{\psi_i} + S_{\phi_i}S_{\psi_i} \\ S_{\phi_i}C_{\theta_i} & S_{\phi_i}S_{\theta_i}S_{\psi_i} + C_{\phi_i}C_{\psi_i} & S_{\phi_i}S_{\theta_i}C_{\psi_i} - C_{\phi_i}S_{\psi_i} \\ -S_{\theta_i} & C_{\theta_i}S_{\psi_i} & C_{\theta_i}C_{\psi_i} \end{pmatrix} \quad (5)$$

Note that *C* and *S* are shorthand for cosine and sine functions. For example, $C_{\phi_i} = \cos(\phi_i)$

Velocity state update follows special relativity due to high velocities of the spaceship, so it uses a corrective Lorentz factor

$$v_{i+1} = v_i + a_i\sqrt{1 - \frac{(v_i)^2}{(V_c)^2}} \quad (6)$$

where V_c is the speed of light: V_c =299,792,458 m/s

Finally, the orientation update is assumed to follow and is approximated by the below state transition:

$$\begin{bmatrix} \phi_{i+1} \\ \theta_{i+1} \\ \psi_{i+1} \end{bmatrix} = \begin{bmatrix} \phi_i \\ \theta_i \\ \psi_i \end{bmatrix} + \begin{bmatrix} 1 & S_{\phi_i}tan_{\theta_i} & C_{\phi_i}tan_{\theta_i} \\ 0 & C_{\phi_i} & -S_{\phi_i} \\ 0 & \frac{S_{\phi_i}}{C_{\theta_i}} & \frac{C_{\phi_i}}{C_{\theta_i}} \end{bmatrix} \begin{bmatrix} \omega_{x,i} \\ \omega_{y,i} \\ \omega_{z,i} \end{bmatrix} \quad (7)$$

Note that, for simplicity, we did not repeat the noise term in equations 3,5 and 6, but in general as you can see in equation 1, the process contains some noise (unmodeled dynamics).

Your simulation of state transition should not account for that noise in the update of states but in the error covariance estimation (according to EKF formulation).

2. **Measurement Model:**

Spaceship measurements include the 3D position of the spaceship and its ϕ angle. ξ_i is the measurement noise vector at time *I* which is a 4x1 vector, and believed to have a zero-mean Normal distribution with covariance of *R*.

R can be affected by large improbability (will be temporarily increased) and would take about 7-8 seconds to return to normal covariance (an exponential decay is not a terrible model for it).

$$Z_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ \phi_i \end{bmatrix} + \xi_i \tag{8}$$

$$\xi_i \sim \mathcal{N}(0, R(\mathcal{I}(A_i)))$$

3. Data provided: Heart_of_Gold_Improp_drive.mat

Event: the events happening in the vicinity of the spaceship at each moment

Events_probability: self-explanatory, to be used to compute improbability (probably needed for T3-1)

T: ΔT, sampling time of 1 sec

u: available interpreted engine forces and heading changes of Heart of Gold. u(:,1) applies at time 1 to change the current state of the system X(:,1) and transition to X(:,2)

x_trunc: this is truncated actual X vectors for the first 100 seconds so that you can compute the actual error of your estimation. Note that we did not provide the 50 last seconds of actual X so that we can evaluate your EKF performance on that as a test set.

Z: Sensors' measurements for the 150 seconds, in the EKF loop you don't need to use Z(:,1) but from Z(:,2) onward, you should be able to use Z in the correction steps of EKF.

Your tasks – detailed version

Your task is to implement an EKF and evaluate and interpret the results

T1) mathematical derivation of needed terms such as Jacobian(s). Copy a parametric form of those matrices in your solution report.
T2) implement an EKF for this spaceship in one of these languages (MATLAB, Python, or C, C++). **Append the code in your submission.**
T3) first tune *Q* and *R* as constant matrices. Use diagonal *Q* and *R* matrices. For initial guess, feel free to start with these:
$Q_0 = diag([1,1,1,2.5,0.5,0.5,0.5\ ])$
$R_0 = diag([500,500,500,1\ ])$
Let's call the EKF you are happy with at this stage EKF1.
T3-1) Implement at least one adaptive version of Q and R. Copy the closed-form equation for your adaptive Q and R in the solution. The **EKF2** is the name if your EKF with variable Q and R.
Hint: *The Q_i and R_i, their diagonal elements, may increase (in the actual system) with a*

trigger of improbability, i.e., due to the large value of improbability. R_i *is also sensitive to the norm of sensor measurement and can be increased by the norm of* Z_i.

T4) Save Kalman gains, state estimation error for EKF1 and EKF2 for the first 100 trials (100 seconds).

T4-1) Generate legible graphs (with legends, axis labels and captions) to compare the **actual states vs EKF1 and EKF2 estimated states vs measurements (Z) for X(1,:) For X(4,:), only show the results of EKF1 and EKF2 vs the X(4,:)**

One graph for each mentioned state for the whole **1-100 samples** (or use subplots 2x2)

One graph for **1-40 samples**

One graph for **41-70 samples**

One graph for **71-100 samples**

Discuss what you see in each of the 3 last graphs and if one can interpret which EKF outperforms the other one.

T4-2) compare the errors of the **two EKFs in terms of RMS error, mean error, and standard deviation of error for X(1,:), X(4,:)**.

T4-3) compute the norm of Kalman gains (squared root of sum of all squared elements) for each time instance for both EKFs and **compare them in a graph and numerically** (using mean +- std in each of 1-40, 41-70 and 71-100 windows).

T4-4) use the results of 4-1, 4-2, and 4-3 and discuss the strengths and weaknesses of your designs.

T4-5) give some suggestions to improve the state estimator (here you can suggest something that possibly deviates from an EKF. We don't need math for this part, but just rationales and descriptions)

T5) generate estimation of states for samples 101-150 (samples you are not provided with the actual state values). **Submit them as a csv file (7x50), and without any heading. Use this naming format for your file: ImprobDrive_xxxxxxxx.csv**

where xxxxxxxx should be replaced by your student ID.

Top 3 performances on the last 50 samples (101-150) in terms of RMS error will receive 10% bonus to their final exam mark.

Important note:

Especially, if your estimations are spiky and a bit off! EKF is basic when dealing with a complex system like improbability drive!

Accordingly, most of the marks here are assigned to your EKF implementation, clarity of figures, results in your report and your discussion based on obtained results rather than the performance itself.

Also note that this is simpler than it looks!

Problem 3: Feedforward Neural Network for Sensor Fusion in Wind Turbine Condition Monitoring (with possible alternative Bayesian approach)

Monitoring wind turbine health is crucial for predictive maintenance and preventing mechanical failures. Consider a wind turbine that experiences vibrations and slight bending due to varying wind conditions. Two 3-axis accelerometers are installed: one at the top of the nacelle and another on the bearing. Your task is to analyze the health condition of the wind turbine (healthy or faulty: fractures, imbalance, etc.) based solely on these accelerometer readings.

You are provided with accelerometer data averaged per second at different timestamps. The file **Neural_network_data2.zip** contains all train and test data. Using these accelerometer readings, you will predict the turbine's operational state.

Tasks:

A) Separate Predictions:

1. Construct two Feedforward Neural Networks (ANN1 and ANN2), each predicting the turbine's health state independently – one using the nacelle accelerometer data and another using the bearing accelerometer data.

   Recommendation: you don't have to follow this!

   a) For output layer use two neurons, and code the class labels into
   $\hat{y}_i$=[output1 output2] = [1 0] for class label 0
   and $\hat{y}_j$= [output1 output2] = [0 1] for class label 1
   This means that you need to code your labels ($\hat{y}_i$) to this format as well (~binary coding)

   b) Use a categorical cross-entropy loss function, that evaluates how similar the vector of [output1 output2] of your network (on training data) is to the similar coding of the labels.

$$L = -\sum_{i=1}^{2} \left(y_i log(\hat{y}_i) \right) \tag{9}$$

   c) Use a softmax activation function for the last layer

2. Find a network architecture (number of layers[2], neurons, hidden layer activation functions), data preprocessing techniques (e.g., normalization), and optimizer to achieve an acceptable accuracy (ideally greater than 67%) on the *X_val* and *y_val* for each sensor individually.
3. Clearly document and explain your chosen neural net architecture and the reasoning behind the selected optimizer, loss function, and preprocessing methods (if applicable).
4. Provide learning curves (training, test and validation accuracy vs epochs) for each model.

B) Fusion Predictions:

1. Utilize a similar network architecture developed previously to implement sensor fusion methods:
    a. **Early Fusion:** Append the data from both accelerometers and use that augmented input for a neural network (ANN3). Train and test the network similar to part A. (target for performance on validation data above 78%)
    b. **Late Fusion:** Use the separately trained networks for each accelerometer (ANN1 and ANN2 from part A). Do not transform those outputs to categorical 1 and 0. Use the Product T-norm to fuse them.
    $\mathbf{T_{prod}(a,b)=a.b}$
    For example if the output of ANN1 and ANN2 are **0.2** and **0.4**, then you get a late-fused output of **0.08** (probability of being in class1) → class label 0.
    If your ANN1 and ANN2 outputs are binary-vector encoded, e.g., [0.8 **0.2**] and [0.6 **0.4**] then you would get [0.48 **0.08**] (0.08 probability of being in class1) → class label 0.
2. Compare and clearly document the performance of each fusion approach. Perform quantitative analysis (accuracy, and confusion matrices) and discuss which fusion methods perform better.

    Your goal is to achieve and exceed an accuracy of **78%** on the validation data and hopefully around 80% on the test dataset using your fusion models.

Deliverables:

- Clear explanations of your chosen architecture, optimizer, and preprocessing steps (if applicable).

[2] You don't have to go beyond 4 hidden layers to get the desired performance

- Learning curves and performance metrics (including the confusion matrix on training dataset and validation dataset) for the models (for ANN1 and ANN2 from part A, and Early Fusion (ANN3)).
- Detailed comparisons and justifications for the effectiveness of each fusion approach. State the winning fusion approach.
- Submit your predictions from the best fusion approach on the test dataset as a .csv file along with your code and detailed report. Use this naming format:
  Test_xxxxxxxx.csv
  where xxxxxxxx should be replaced by your student ID.

Top 3 performances neural networks on the test data (the one we reserved for our test) will receive 10% bonus to their final exam mark.

Alternative solution approach:

If for any reason, including lack of time and unfamiliarity, you are unable to implement and train ANNs, you can opt to train sensor-specific and combined-sensor **Bayesian Classifiers and your choice of late fusion, at the expense of 10 points**. If you opt for this make sure to detail what method, you used.