# False Alarm Reduction

Rishabh Garg (2020CHB1051)

Sarbjot Singh (2020CHB1054)

Uttam Saroj (2020CHB1057)

Vaidehi Patidar (2020CHB1058)

A report submitted

for the degree of

Bachelor of Technology in Chemical Engineering

Department of Chemical Engineering

April, 2023

# Table of Contents

## List of Figures

## List of Tables

## List of Equations

## List of Abbreviations

| | |
|---|---|
| ECG | Electrocardiogram |
| AACN | American Association of Critical-Care Nurses |
| RNs | Registered Nurses |
| ICU | Intensive Care Unit |
| HR | Heart Rate |
| BP | Blood Pressure |
| ABP | Arterial Blood Pressure |
| PPG | Photoplethysmogram |
| EEG | Electroencephalogram |
| FA | False Alarm |
| SQI | Standard Quality Indices |
| RF | Random Forest |
| MCMC | Markov chain Monte Carlo |
| SpO2 | Saturation of Peripheral Oxygen |
| PGIMER | Postgraduate Institute of Medical Education & Research |
| IEC | Institute Ethical Committee |

# Acknowledgments

We would like to express our heartfelt gratitude to our course instructors **Dr. Navin Gopinathan** and **Dr. Sarang Gumfekar** for providing us with invaluable guidance and support throughout this project. We are deeply grateful for your encouragement and assistance. We would also like to thank **Dr. Jayaram Valluru** for guiding us about the technical aspects of our project. He even attended some of our presentations, correcting and improvising us whenever necessary. **Dr. Madhumita Premkumar**, Department of Hepatology, PGIMER also helped us throughout our journey. She provided us with all the necessary information about the procedure.

Finally, we would like to express our gratitude to all those who have been a part of our journey in one way or another, and we hope that this project report will prove to be useful to the readers.

# Abstract

Walking onto any hospital floor, it is easy to quickly become overwhelmed by the hums of ventilators and the blaring of alarms. These devices are designed to inform healthcare providers of critical medical situations, but can actually be hazardous and create alarm fatigue. This in turn, can hinder medical care and lead to adverse patient outcomes. False alarms are a common problem causing potentially dangerous consequences. They can lead to a decrease in trust in alarm systems, a rise in staff stress and weariness, as well as significant harm from patients. In this report we aim to provide a comprehensive overview of the current state-of-the-art in false alarm reduction and practical solutions for improving alarm system performance in medical and industrial environments. In our study we examined the causes of these false alarms, the work that has been done and the ways we could make our contribution.

# Introduction

Frequent false alarms are a bane of modern healthcare, intensive and in-patient monitoring systems. They are known to lead to a disruption of care, and in certain scenarios may adversely impact the patient and the clinical staff. This is likely to occur through noise disturbances (regular/irregular), and desensitization to warnings. As a result, there is a lag in response times. Moreover, patients experience a decreased quality of care, disrupted sleep patterns, sleep deprivation, stress for both patients and staff, and depressed immune systems.

An alarm is generally triggered when a variable (e.g., heart rate) derived from the waveform (e.g., ECG) is above or below a preset (or adjustable) threshold for a given length of time. This is regardless of whether the irregularity is caused by a transition in the physiological state, by an artifact or by medical interventions. The latter may include activities such as moving or positioning the patient, drawing blood and flushing the arterial line, or disconnecting the patient from the ventilator for endotracheal suctioning. Additionally, other reasons also account for the high rate of false alarms. Examples include the use of univariate alarm algorithms and the application of simple numeric thresholds in clinical bedside monitors.[1]

The low specificity of alarm threshold settings, high rate of inactive alarms, and exposure to constant alarms in intensive care units cause depersonalization because of emotional overload.

The researchers define alarm standards as an umbrella term that encompasses three domains: machine learning (for smart alarms using specific algorithms), alarm configuration (ability to alter the alarm parameters to meet patient-specific needs), and alarm designs (standardization of alarm sounds). [2]

As per our discussion with the medical staff and medical officers regarding this issue, we got confirmation on the severity of this false alarm problem. Moreover, They also agreed to provide any sort of help and guidance, which may be required on this topic.

So, we will try to solve as many problems as possible related to false alarms in our project with the help of anonymous patients' data from the hospitals.

# Literature Survey

## Alarm Fatigue -

Alarm fatigue is defined as desensitization and apathy of healthcare providers to the sound of an overwhelming number of repetitive or simultaneous alarms. This is a major problem recognized by the AACN and the Joint Commission. Between 2009 and 2012, the Joint Commission received 98 alarm-related adverse patient event reports, 80 of which were fatal and 13 caused permanent loss of function. Additionally, between 2005 and 2008, the United States Food and Drug Administration database received 566 reports of inpatient deaths related to alarm fatigue.Therefore, in 2014, the Joint Commission made alarm management a national patient safety goal, aiming to reduce the prevalence of alarm fatigue in nurses to ultimately prevent adverse patient events. [2]

Nurses from all around the world agree that alarms disturb patient care, are burdensome, and diminish trust in alarm systems. The following efforts are made in order to reduce the alarm fatigue in the nursing and medical staff -

i) Educational programmes have been implemented to improve the nursing staff's understanding about alarm fatigue. After the nursing staff completed the educational programmes, there was found to be a significant increase in nursing knowledge about the alarm management. However, there were no statistically significant differences in the number of alarms. This educational programme was helpful in providing the knowledge about the alarm fatigue concepts. But it was not clear in teaching them how to apply this knowledge to practice which may be responsible for why there was no statistically significant reduction in nuisance alarms.[2]

ii) The other one was after performing a literature review, the researchers defined alarm standards as an umbrella term that encompassess three domains: machine learning, alarm configuration, and alarm designs. Machine learning refers to utilizing "smart" alarms that use specific algorithms to determine if the alarms were alerting RNs of critical information or if they were falsely activated prior to generating a sound. Alarm configuration refers to the ability of RNs to alter the alarm parameters to meet patient-specific needs to reduce the number of unnecessary alarms. Finally, alarm design refers to the standardization of alarm sounds based on the machine's perception of urgency, so each sound correlates with a different level of urgency. They emphasize that it is not enough to simply know the domains of alarm standards, but it is crucial for RNs to actually apply this knowledge into their practice to effectively reduce alarm fatigue. The researchers suggest that this gap is due to the lack of collaboration among technical, human, and organizational factors in alarm management programs. However, this study does not propose an alarm management program that may bridge this gap, so it is not feasible to predict what the impact of such a program would have on alarm fatigue in RNs.[2]

As informed by the National Medical Commission (NMC), there are 13,08,009 allopathic doctors registered with the State Medical Councils and NMC as on June, 2022. Assuming 80 per cent availability of registered allopathic doctors and 5.65 lakh AYUSH doctors, the doctor-population ratio in the country is 1:834 against the WHO norms of 1:1,000. The shortage of trained nurses is more dire, with a nurse-to-population ratio of 1:670 against the WHO norm of 1:300.

In most of the studies done and data collected it was seen that around or more than 90% alarms were false. Studies have shown that adverse outcomes, including death and permanent disability, emerge in healthcare environments because of poor management of alarms, which play a critical role in healthcare units. Physiological data can be severely corrupted by artifacts (e.g. from movement), noise (e.g. from electrical interference) and missing data (e.g. from transducer ‘pop’ leading to impedance or pressure changes and a resultant signal saturation). Staff and patients in intensive care units (ICU) are confronted with an excessive amount of monitoring alarms on a daily basis . Although alarming ICU staff of a deviation of a vital parameter is necessary in the context of delivering intensive care , the majority of the alarms in the ICU are reported as false positives, caused by patient movement or poor skin contact of electrodes for example, and are often clinically non-actionable.

This adversely affects the ability of the nurses to fulfill their critical patient care responsibilities, and thus, nurses resort to risky measures that adversely affect patient safety such as lowering the alarm volume to an inaudible level or turning off the alarm completely, ignoring the alarms, slowing the response, and delaying rapid response to the patient in emergencies, which results in a decrease in the number of interventions to the patient. Thus, critical patient events may go unnoticed, and damages to the patient that may be avoidable may happen.

Various noise cancellation algorithms such as median filtering or Kalman filtering have been used to suppress false alarms. While transient noise can be removed by median filtering it is brutally non-adaptive. Kalman filtering, on the other hand, is an optimal state estimation method, which has been used to improve heart rate (HR) and blood pressure (BP) estimation during noisy periods and arrhythmias. A promising solution to the false alarm issue comes from multiple variable data fusion, such as HR estimation by fusing the information from synchronous ECG, ABP and photoplethysmogram (PPG) from which oxygen saturation is derived.

It is imperative for hospitals to develop evidence-based alarm management programs that are tailored to bridging the gap of simply knowing alarm management standards and actually applying them into practice. The greatest barrier in the implementation of an alarm management program is the lack of an existing evidence-based standardized program due to insufficient high-quality evidence.[3]

## Problem Statement

Alarm fatigue is a critical issue in the healthcare industry, leading to desensitization and apathy among healthcare providers to the sound of an overwhelming number of repetitive or simultaneous alarms. The high prevalence of false alarms is a major contributor to alarm fatigue, resulting in reduced patient safety and avoidable adverse events. The aim of this project is to analyze medical alarm data and reduce the false alarm rates through the implementation of a comprehensive alarm management program. The project also aims to gather frontline staff feedback, observe nursing practice and process in real-time, and conduct noise data analysis to ensure effective implementation of the alarm management program. The goal of this project is to improve patient safety by reducing alarm fatigue and enhancing the efficiency of alarm systems in healthcare settings.

## Objectives

1. Collect and analyze the medical alarm data
2. Reduce the false alarm rates
3. Frontline staff feedback
4. Real-time observation of nursing practice and process
5. Noise data analysis/interpretation

## Our Approach

The current alarm systems in intensive care units (ICUs) are based on univariate algorithms, which can lead to false alarms. To address this issue, the proposed approach is to integrate multiple alarms by gathering anonymous ICU system data from various hospitals and training a machine learning algorithm based on filtered parameters and features determined by a panel of medical specialists. The trained algorithm will then be validated across multiple hospital systems to create a robust algorithm that can fit the majority of ICU systems.

To maintain the accuracy and efficiency of the ICU systems, predictive maintenance techniques will be incorporated to detect and address hardware issues before they lead to false alarms. Furthermore, the opinions and suggestions of nurses, medical staff, and doctors will be gathered to incorporate their feedback and improve the design of the algorithm.

A recommender system will be developed based on the features used in the algorithm to prioritize patient care in situations when multiple alarms from different patients are sounding simultaneously. Additionally, an alarm tune standardization system will be implemented based on vital alarm correspondence, where a separate tune will be assigned for each alarm type, such as a unique tune for heart rate alarms. This system will enable health workers to prioritize patients without having to look at their monitors to identify the respective alarm type.

The proposed approach also aims to use the medical history data of anonymous patients to predict suitable thresholds for their vitals. For instance, if a patient has a history of diabetes or alcohol consumption, a threshold can be set for their vitals to reduce the occurrence of false alarms.

In conclusion, the proposed approach aims to address the limitations of current ICU alarm systems by integrating multiple alarms, developing a robust algorithm, and incorporating feedback from medical professionals. The use of predictive maintenance techniques and medical history data can further enhance the accuracy and efficiency of the ICU systems. The recommender system and alarm tune standardization can help prioritize patient care and enable quick identification of the respective alarm types. Overall, this approach has the potential to significantly improve the effectiveness of ICU alarm systems and enhance patient care.

# Tools and Technology

After doing the research we got to know that the main source of errors are :

- Noise in the signals
- Univariate algorithms
- Sensitive Alarm thresholds

**Noise in the Signals :**

An alarm is generally triggered when a variable (e.g., heart rate) derived from the waveform (e.g., ECG) is above or below a preset (or adjustable) threshold for a given length of time. This is regardless of whether the irregularity is caused by a transition in the physiological state, by an artifact or by medical interventions. Electrical equipment such as cell phones, computers, or other electronic devices in the vicinity of the monitoring equipment can cause electrical interference, leading to noisy data. [2]

So, to solve this problem and get a noise free and smooth data Kalman filter can be applied-

**Kalman Filter :** Kalman filter is an algorithm used for filtering and smoothing the noisy signals. It uses a recursive mathematical model to estimate the state of a system based on a series of noisy measurements. The Kalman Filter assumes a linear model, so if there is a non-linearity in the process, first we need to convert it into a linear model using Taylor series expansion. Then we need to convert this linear continuous time model into a discrete time representation or state space model.

$$\frac{dX}{dt} = f(X,U) \quad \text{(Non-linear continuous time model)}$$

$$\Big\Downarrow$$

$$\frac{d\dot{X}}{dt} = A\dot{X} + B\dot{U} \quad \text{(Linear continuous time model)}$$

$$\Big\Downarrow$$

$$X_k = \Phi X_{k-1} + \Gamma U_{k-1} \quad \text{(Discrete time representaton)}$$

**Equation-1**

So after converting a non-linear continuous time model to a linear discrete time model, Kalman filter uses this model as a mathematical form of a process to represent its states, also called state space model. Now this filter estimate the states and remove the noise mainly using two steps -

**i) Prediction step -** In this step the filter predicts the current step based on the previous state estimates and state space model of process. The Kalman filter also predicts the error covariance, which quantifies the uncertainty associated with the state estimation. It takes into account both the process noise and measurement noise.

**Linear Process**

$$\begin{aligned} \mathrm{x}_k &= \Phi \mathrm{x}_{k-1} + \Gamma \mathrm{u}_{k-1} + \mathrm{w}_{k-1} \\ \mathrm{y}_k &= \mathrm{C}\mathrm{x}_k + \mathrm{v}_k \end{aligned}$$

**Equation-2**

$w_{k-1}$ is the process noise and $v_k$ is the measurement noise, $x_k$ is the true state variable and $y_k$ true measured variable, Φ, Γ and C are system dynamics parameters.

$w_k$, $v_k$ and $x_k$ follow Gaussian distribution, so they only need mean and covariance to predict the information about the input state variable.

In the step of prediction we have a previous belief $P(x_{k-1}|y_{k-1})$ and we want to know what can be predicted about $x_k$ i.e. we want to find $P(x_k|y_{k-1})$. We use the Chapman-Kolmogorov equation:

$$P(x_k|y_{k-1}) = P(x_k, y_{k-1}) / P(y_{k-1})$$

Assuming the state at time k is only dependent on the state at time k − 1 and is independent of the observation history $y_{k-1}$ when $x_{k-1}$ is given. In the above equation $P(x_k|x_{k-1})$ is derived from the state equation.

**ii) Update Step -** The step of update uses new measurement $y_k$ to construct the posterior $P(x_k|y_k)$. The update or corrector is carried out via the Bayes rule.

$$P(x_k|y_k) = P(x_k|y_k, y_{k-1})$$

$$= P(y_k|x_k, y_{k-1})P(x_k|y_{k-1}) / P(y_k|y_{k-1})$$

Assuming that new measurement $y_k$ is independent of the previous measurements $y_{k-1}$ we may find the update or corrector:

$$P(x_k|y_k) = P(y_k|x_k)P(x_k|y_{k-1}) / P(y_k|y_{k-1})$$

Using MAP estimation:

$$\widehat{X}_{k|k} = argmax\, P(x_k|y_k)$$

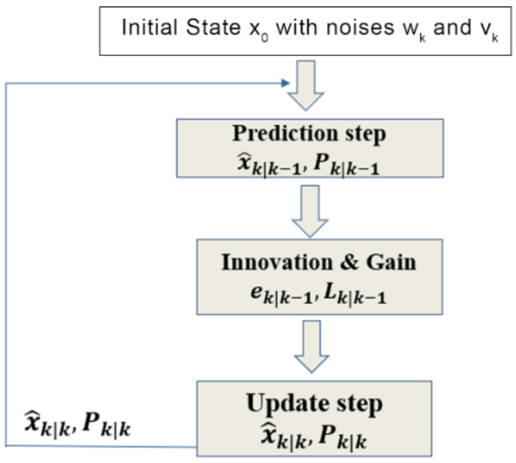

**Fig :1**

The process then repeats by going back to the prediction step and continues to update the state estimate as new measurements are obtained.

**Univariate Algorithms:**

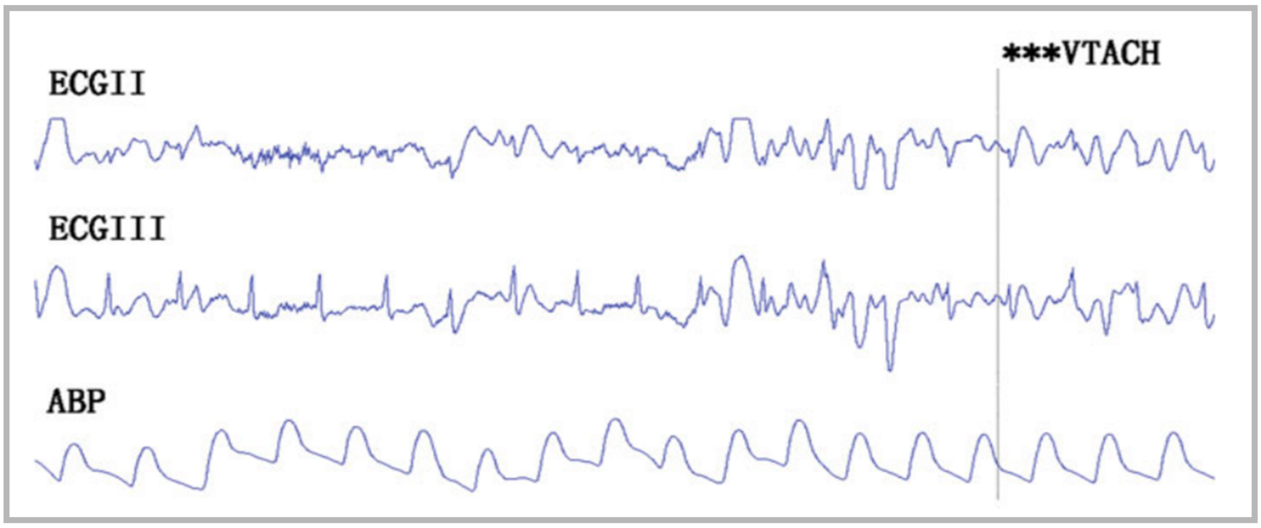

**Fig 2- False ventricular tachycardia alarm, 30 s snapshot of two leads of ECG and an arterial blood pressure signal (ABP)**

In Fig 2, we can see that after half stage there is significant noise in the ECGII and ECG III signals which triggers corresponding alarms while the parallel readings of ABP signals suggest it is a normal behavior. In medical science ABP signals are considered as more important vitals as compared to ECGII and ECGIII which are enough evidence to suggest that it is normal and this alarm is a false alarm.[2]

Thus if we use multivariate algorithms which consider the combination of such signals (i.e., which signal reading should be given how much weightage, can be decided in consultation with medical professionals), it will reduce the false alarm count significantly.

**Sensitive Alarm thresholds :** In hospitals many alarms are triggered because of minute movements of the patient, e.g. turning while sleeping, moving your hands, neck, lower body etc. These are very normal movements but because of the alarm thresholds being highly sensitive, it causes the alarm to buzz unnecessarily. Although medical workers have the access to adjust these thresholds, because in their training they are only taught how to operate these machines, not how to make an educated adjustment in the threshold limits. As a result they make the changes according to their experience or knowledge which often results in suppression of true alarms.

To take on these problems, we are proposing a data-driven Machine Learning approach in which we propose to make a panel of medical professionals who will help in identifying a particular medical alarm as true or false. Based upon their feedback we will train a Machine Learning model which will tell whether an triggered alarm is true or false.

In the research paper, by Qiao Li and Gari D. Clifford [2], they showed significant false alarm reduction and were able to avoid suppression of true alarms simultaneously. The problem with this algorithm was that the data on which it was cross-validated, came from the same devices on which it was trained.

The algorithm is proposed as follows:

**Data Set:**

- Data set contained simultaneous ECG, ABP and PPG recording corresponding to 2301 life-threatening alarms.
- Wave data extracted from 30 sec before and 10 sec after onset of alarms have been used to train the model.
- Team of 3 experts labeled the data as FA(false alarm) or true alarm.

**Data Pre-Processing:**

- Total 147 features + SQI( standard quality indices) metrics were extracted from ECG, ABP, PPG, and SpO2.
- Typical features such as Heart rate, blood pressure , SpO2 levels and PPG peak are taken into consideration for training the algorithm.
- Each feature has five sub-feature : minimum, maximum, median, variance, and gradient (derived from a robust least squares fit over the entire window).

**Algorithm:**

Before going into the depth of the algorithm, it’s important to introduce decision trees.
**Decision Trees:** It is a binary classification algorithm in which it selects a feature and puts it under some random constraints and divides the data points into those two categories, one which follows the constraint and the others which don’t. Now from these divided datasets it again selects some features( it can be the same as the previous one) and applies any random constraint on it. This cycle continues until the whole data is totally segregated. Now which feature should be constrained to what conditions is decided on the basis that the algorithm chooses the shortest path to do the data segregation. This shortest path is our final model.
Now this algorithm is a data sensitive algorithm and we often get stuck at local minima instead of the global minima. Biasing and overfitting is also very common. To overcome these challenges we use a much superior algorithm named Random Forest (RF).

**Random Forest:** In this algorithm we consider multiple independently trained decision trees and we do majority voting. Based upon this majority voting we decide to which category a particular data point belongs. Now this algorithm decreases the chances of hitting the local minima and is more robust. The accuracy of the algorithm increases with increase in no. of decision trees. Only problem is the training time of the algorithm is high, so there's always a tradeoff between no. of decision trees and accuracy.
**Modified Random Forest:** Now we randomly pick two trees from the trained forest and re-initialize them by changing their initial features and then calculate the forest contribution or likelihood( how many outputs of a particular tree matches the final RF output). If this contribution is higher than some specified limit, we update the forest with this new one otherwise it's left untouched.
We keep doing the random picking and reinitialization until 20% of the user specified iterations are reached, after which we start saving the Forests. Once user defined iterations are completed, we will have a set of forests as our final output.

Now, we will do the majority voting of these forests and this will further increase the chances of reaching the global minimum.

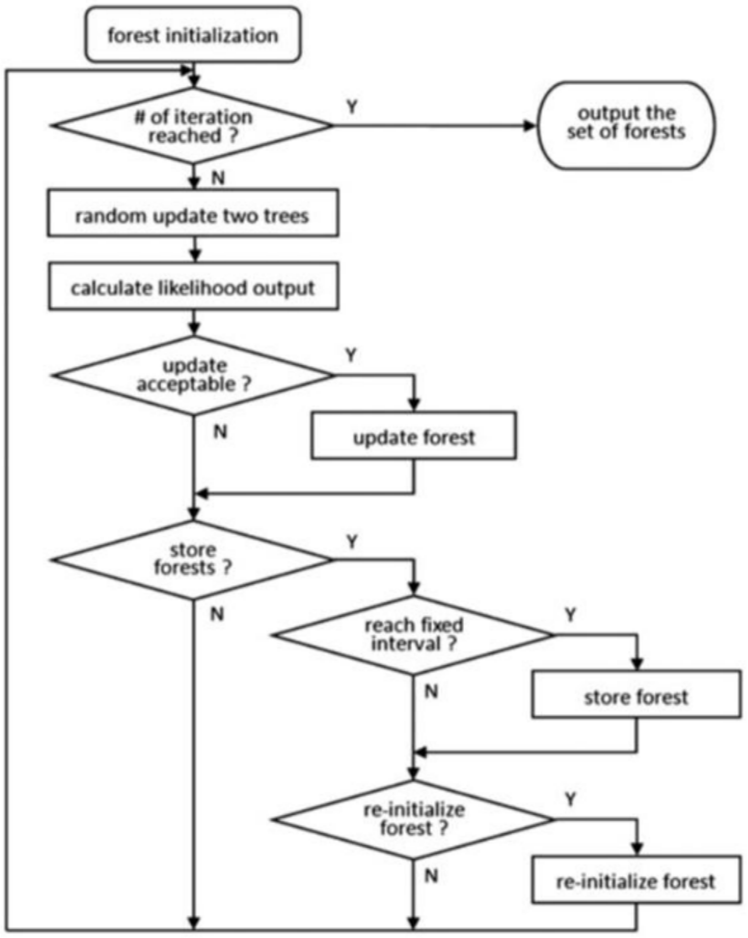

**Fig-3: How the algorithm works each step at a time.**

**Result Visualization:**

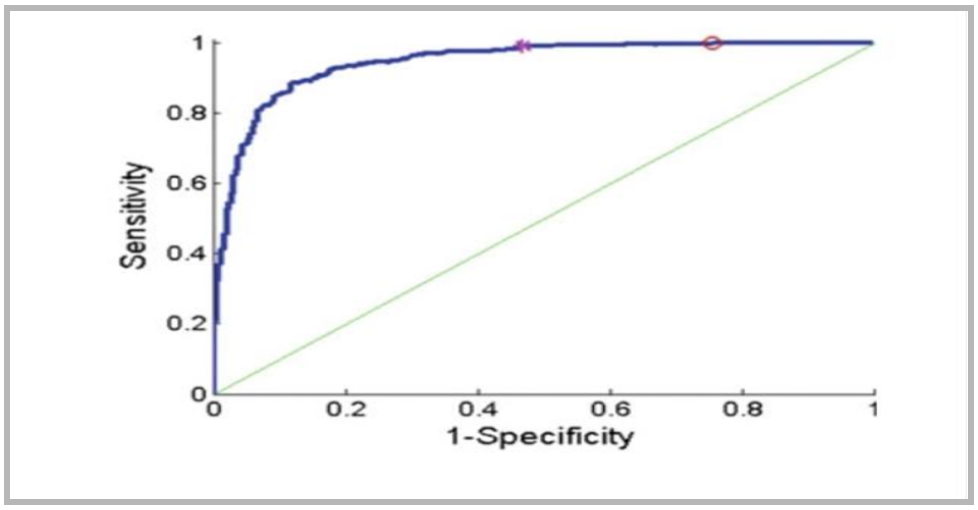

**Fig-4**

Here in Fig-4 Sensitivity is the measure of true alarm suppression. 100% sensitivity means that no true alarm is suppressed. On X-axis Specificity is the measure of false alarm suppression. 20% Specificity means that we have successfully suppressed 20% false alarms. Now according to medical experts 1% tolerance in sensitivity is acceptable. Thus we prefer to operate at suboptimal points as red circled in Fig-4 . At sub-optimal point, sensitivity = 99.2% & specificity = 53.3%.

However, we wanted to test their algorithm on different devices combined with our approach, which is to reduce noise using a series of robust Kalman Filters. Unfortunately, we couldn't get relevant medical data either from any online resource or from any hospital nearby. We went to PGIMER Chandigarh and talked to professors over there and they told us to first get ethical clearance from their PGIMER ethical committee. We also filled the IEC (Institute Ethical Committee) form for getting clearance from our institute, but it got rejected as there was no clearance from the PGIMER, which will take 2-3 months in the process.

We couldn't test our hypothesis because of the unavailability of medical alarm data. But thanks to Dr. Navin Gopinathan who provided us with chemical adsorption lab data which we used to test at least the noise reduction proposition of our hypothesis.

## Case Study

In order to compensate for the real medical alarm data, we are applying our false alarm reduction approach on the chemical adsorption lab data shared by Dr. Navin Gopinathan, with the sample name:- “DH16-double layer module01 200'C/ 24 hrs(Furnace 200'C) 20000 ppm 1.0L/min adsorption 2bar”, which is basically about the diffusion behavior of $CO_2$ adsorption from a $CO_2/N_2$ gas mixture on Zeolite-13X in a fixed-bed column. In this case, the gas mixture is passed through the packed bed column of the adsorbent material at a specific flow rate, temperature, and pressure. As the gas mixture flows through the column, $CO_2$ molecules are selectively adsorbed onto the surface of the zeolite-13X, while $N_2$ molecules pass through the column without being adsorbed.

If the adsorbent is not saturated, diffusion of $CO_2$ into the zeolite pores continues until the concentration of $CO_2$ inside the pores reaches an equilibrium with the concentration of $CO_2$ in the gas stream. Once the equilibrium is reached, there is no net mass transfer of $CO_2$ from the gas phase to the solid phase. However, if the adsorbent becomes saturated with $CO_2$, desorption or displacement of the adsorbed $CO_2$ by $N_2$ may occur, and so the concentration of the adsorbate in the outlet stream starts to increase.

The adsorption process in the fixed-bed column can be characterized by different parameters, such as bed saturation, and breakthrough time. Bed saturation is the point at which the adsorbent is completely saturated with $CO_2$, and no further adsorption can take place. The term "breakthrough" refers to the point in time when the concentration of $CO_2$ in the outlet gas stream begins to increase, indicating that the adsorbent is becoming saturated with $CO_2$ and the adsorption capacity is being exceeded. [5]

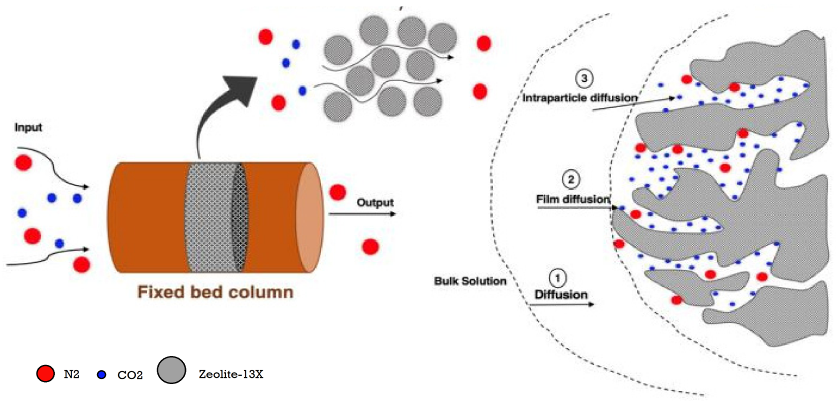

**Fig-5: Adsorption mechanisms of $CO_2$ adsorption from $CO_2/N_2$ gas mixture on Zeolite-13X.**

Now, the concentration data of $CO_2$ is recorded with the time interval of 0.08333 minutes and the experiment ends at 117 minutes where the 100% concentration of the $CO_2$ gas gets recorded by the analyzer. During this time period, total 1382 data points were recorded and on the basis of these data points, a graph is plotted between [**C** ($CO_2$ concentration (in ppm) at any time) **/ $C_0$** (Maximum concentration)] and Time (in mins.) to visualize the $CO_2$ breakthrough curve (Fig. No. ). Although the actual breakthrough point was taken as 2% in this experiment, but as mentioned in the "Principles of Mass Transfer and Separation Process" by B.K Dutta, we are considering the breakthrough level to be 10%, for the sake of generalization. Accordingly, the breakthrough time is recorded as 82.71 minutes.

NOTE: From the data used, it can be observed that some concentration of $CO_2$ has leaked initially directly in the outlet stream without getting adsorbed on the Zeolite-13X adsorbent. So, we can ignore the initial 0.833 minutes concentration readings (before point A in **Fig:-8**) of $CO_2$, which are above 10%.

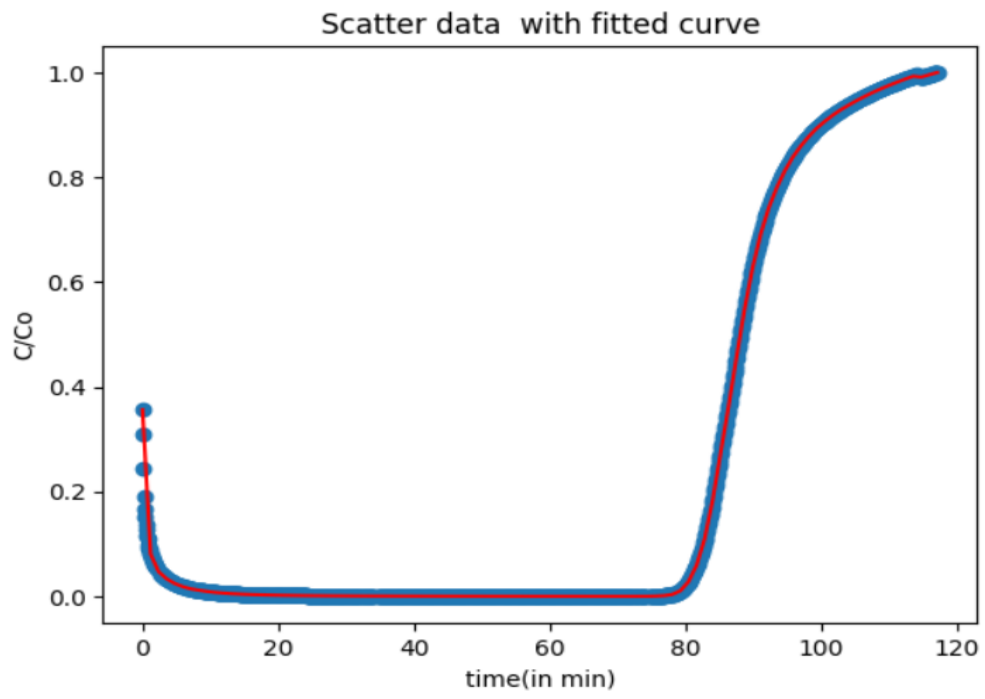

**Fig-6: Curve fitted lab data**

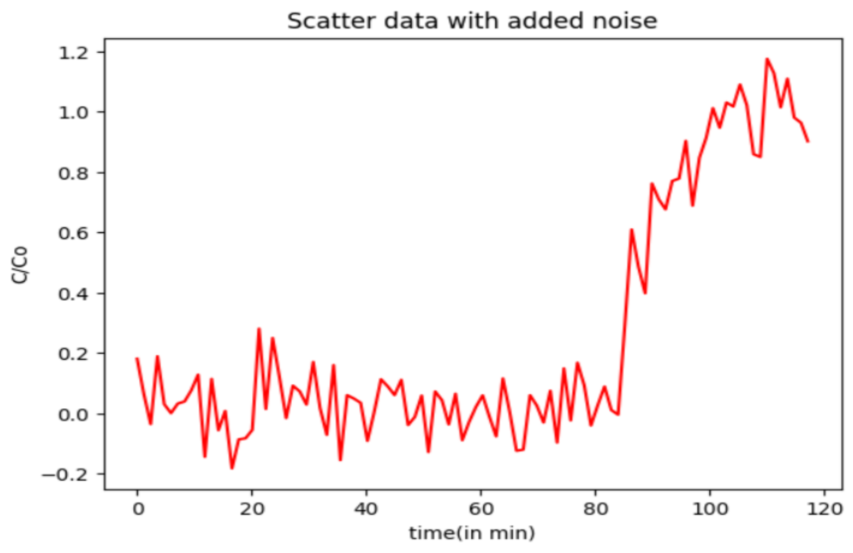

**Fig-7: Curve obtained after adding random noise**

Since our data is taken from lab setup, the curve is very smooth (free from noise) as shown in **Fig-6** . But the industrial data will never be smooth so we added noise to the data via writing python code for random noise generation, as shown in **Fig-7.** .

Here, we have taken the 10% threshold to be the case for beeping of alarms, as shown in **Fig-8**. Now, it can be clearly observed from the graph that on adding the noise, the alarm (for the 10% threshold ) will start buzzing even in the middle region (between point A & B), where ideally, there shouldn't be any alarms. So, we have chosen the domain in which the graph comes down below 0.1 till the point it first time touches that 0.1 range again.

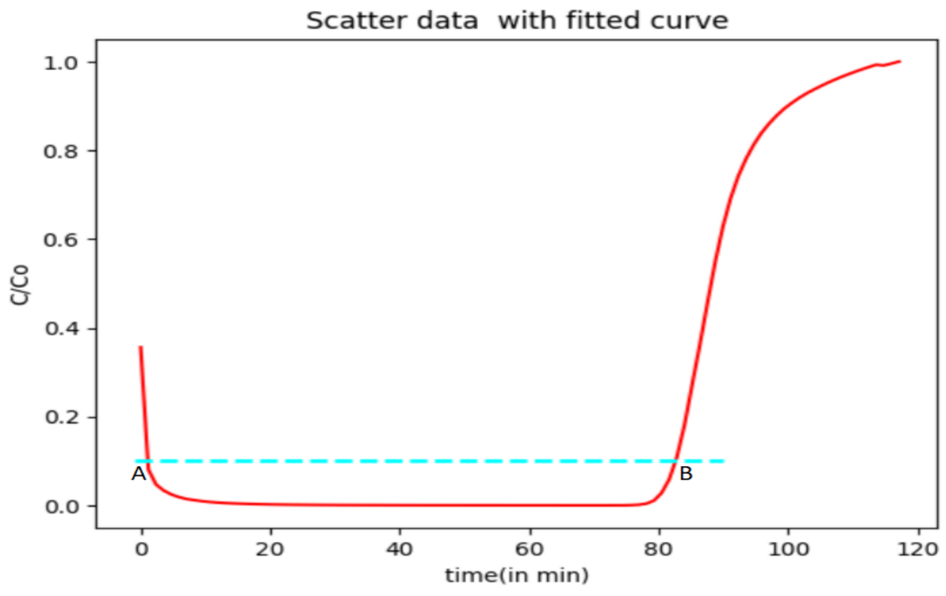

**Fig-8**

In the above figure, the part below the AB line is our region of concern, which represents concentration, y = 0.1 ( or 10% threshold).

With this, we calculated the number of false alarms within this fixed range for noisy data and we found 177 false alarms. We have then tried to reduce this false alarm number with the help of Kalman filter.

## Results & Discussion

**Table-1: False Alarms left after No. of KalmanFilters applied**

| No. of KalmanFilters | No. of  False Alarms |
|---|---|
| 1 | 97 |
| 2 | 36 |
| 3 | 22 |
| 4 | 15 |
| 5 | 11 |
| 6 | 7 |
| 7 | 6 |
| 8 | 6 |

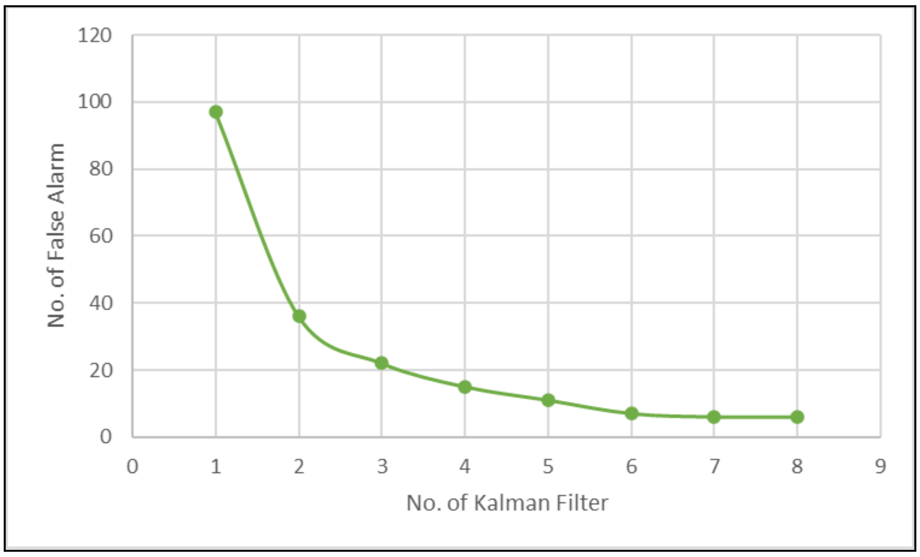

**Fig - 9**

We have used a chain of KalmanFilters to reduce the false alarms till the point it doesn't reduce false alarms any further. The above shown table represents the no. of false alarms encountered after application of each Kalman Filter.

From the table-1 it is clear that false alarms don't get reduced any further after the 7th Kalman Filter. The plot below demonstrates the initial noisy curve and the curve obtained after application of 8 Kalman Filters. Clearly KalmanFilters have reduced the noise quite significantly.

Using this approach we have been successfully able to reduce the False Alarms from 177 to 6.

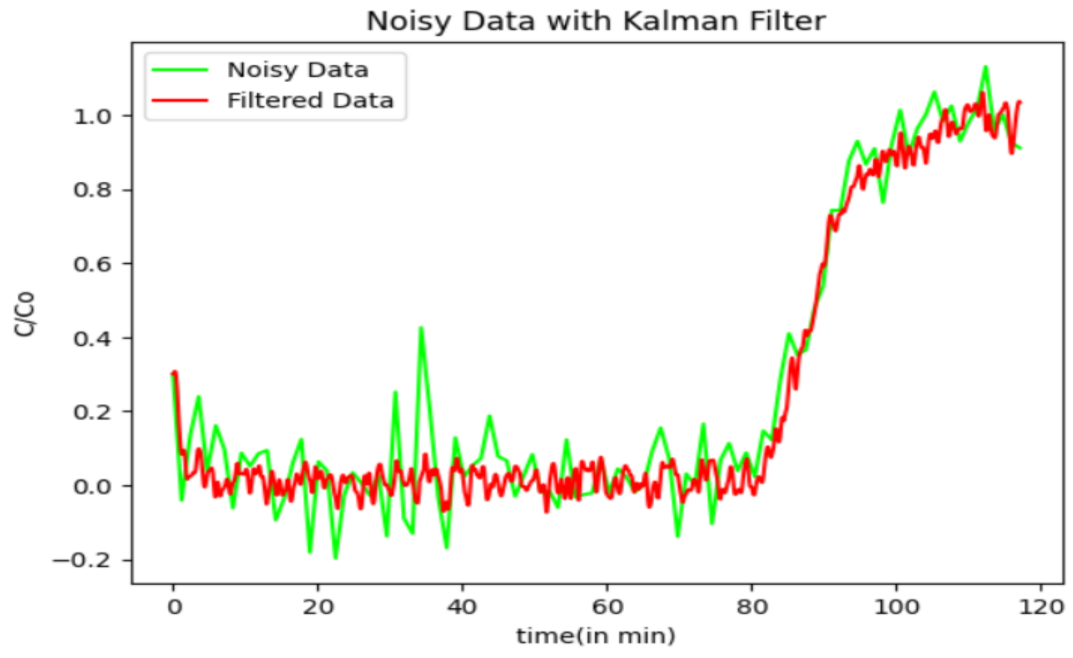

**Fig-10 Noisy data with Filtered data**

We can use similar techniques in medical alarms as well. This will ensure at least noise related errors and false alarms are suppressed.

The close control loop for this case study is shown in Fig: 10, to help in visualization of the actual industrial procedure for the complete process including our approach.

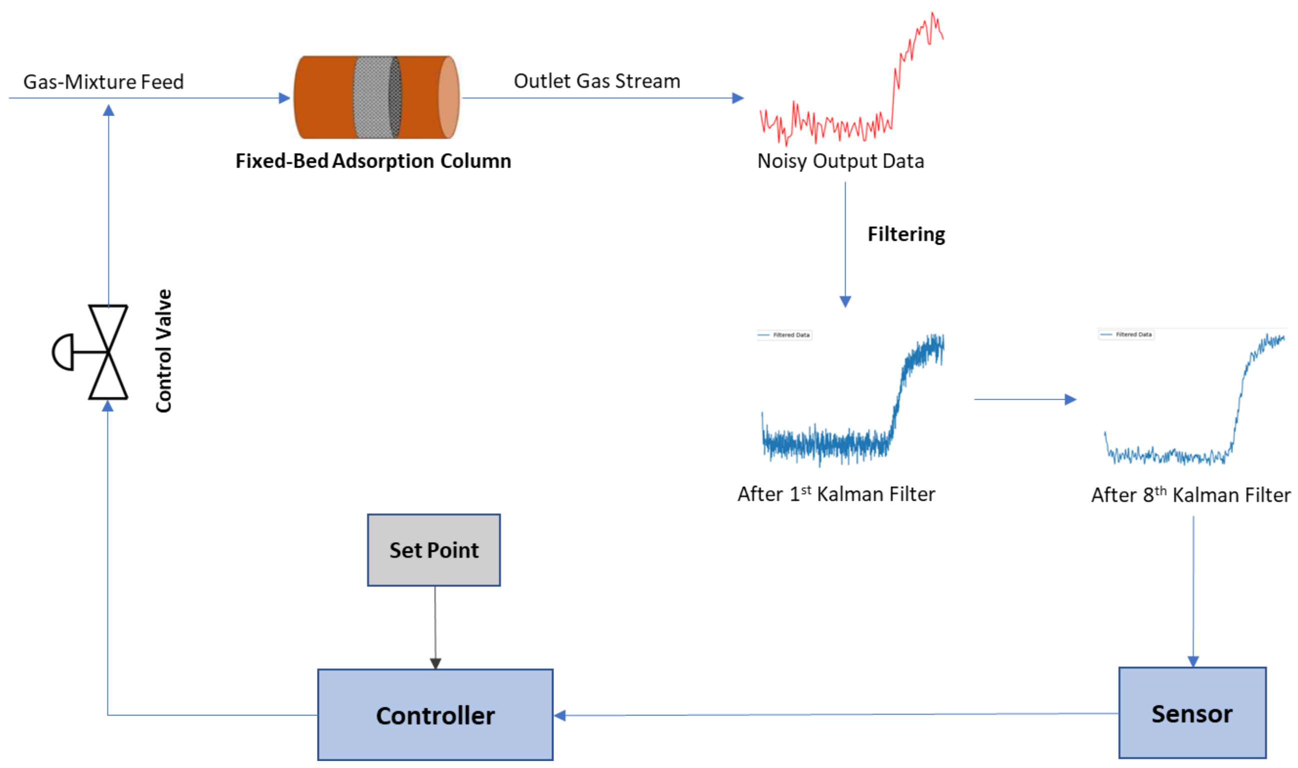

**Fig-11: Closed Control Loop of the Case-Study**

## Conclusion

We tried testing our hypothesis of noise reduction using a series of KalmanFilters on dynamic gas adsorption lab data. In this process, we added random noise to the smooth curve to match it with industrial data. With this approach we have been successful in reducing the number of false alarms from 177 to 6, which is **96.6% reduction**.

Hardware noise in the signal readings is one of the three major reasons for high medical alarm rates. We can apply our KalmanFilter approach in the medical alarms as well. This will result in significant reduction of false medical alarms, which will reduce the problem of alarm fatigueness and subsequently will translate into reduction of suppression of true alarms .

Our algorithmic and data driven approach can also be used in the reduction of false alarms in other domains as per the specifications and requirements.

## Future Scope

Implementing an alarm management strategy in healthcare has the potential to save hundreds of thousands of dollars and hundreds of hours of time for registered nurses. This is because it can reduce the problem of alarm fatigue among medical workers by reducing the number of false alarms. With a decreased rate of false alarms, medical workers can respond more actively to each alarm and reduce the chances of missing actual true alarms.

Reducing the total number of alarm buzzers through predictive maintenance and with our algorithm, can not only increase the efficiency of monitoring systems but also accelerate the recovery of patients in shared hospital rooms. This is because multiple patients in a room can be affected by alarms from other patients, causing discomfort and disrupting their rest.

With the use of Tune standardization and our recommendation systems, patient prioritization can be improved. This can help hospitals to better manage patient care and ensure that those in the most urgent need receive attention first. Overall, these strategies can help to improve patient outcomes, reduce alarm fatigue among medical workers, and save hospitals money through increased efficiency.

In addition to that, our algorithmic and data driven approach can also be used in other sectors of alarm requirements. For example; In the chemical industries, in the fire extinguisher systems, etc. as per the specifications and requirements.

## References

[1] Signal Processing: False Alarm Reduction, MIT Critical Data, Secondary Analysis of Electronic Health Records,DOI 10.1007/978-3-319-43742-2_27

[2] Determining the Impact of an Alarm Management Program on Alarm Fatigue among ICU and Telemetry RNs: An Evidence-Based Research Project, Published online 2022 May 13. doi: 10.1177/23779608221098713

[3] The influence of patient characteristics on the alarming rate in intensive care units: a retrospective cohort study: Published online 2022 Dec 16. doi: 10.1038/s41598-022-26261-4

[4] The effect of interventions made in intensive care units to reduce alarms:A systematic review and meta-analysis study:Gulnur Gul and Seyda Seren Intepeler and Murat Bektas ,https://doi.org/10.1016/j.iccn.2022.103375.

[5] The Diffusion Behavior of CO2 Adsorption from a CO2/N2 Gas Mixture on Zeolite 5A in a Fixed-Bed Column: by Arunaporn Boonchuay and Patcharin Worathanakul, 2022, Volume 13, Issue 4, atmos 13040513, https://doi.org/10.3390/atmos13040513

# Annexure

- Google drive link to the chemical adsorption lab data excel file :
  Original_Labortory_data - Google Drive
  https://drive.google.com/drive/u/0/folders/1fj_fcftlcMEd0JdLbqql4DTinCK6Ke-p

- Google drive link to the modified data file used to run the code:
  Adsorber_data - Google Drive
  https://drive.google.com/drive/u/0/folders/1JATB7MPn2LKWWETq-m5FdXkJHFA9rVCq

- Google Collaboratory link to python code used for carrying out the case study analysis:
  False Alarm - Colaboratory (google.com)
  https://colab.research.google.com/drive/1HVyaGeb2xokDO_wGYb3RE1P-Sb5Rw0i7#scrollTo=3ykBH_8zRQ8o

# Appendix

**Python Code -**

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('adsorber.csv')
X = dataset.iloc[:1382, 2].values
Y = dataset.iloc[:1382, -1].values

plt.scatter(X, Y, color = 'red')
plt.title('Lab Plot')
plt.ylabel('C/Co')
plt.xlabel('time(in min)')
plt.show()

from scipy.interpolate import make_interp_spline
# fit a cubic spline to the data
spline = make_interp_spline(X, Y)
# evaluate the spline over a finer grid
Xfit = np.linspace(X.min(), X.max(), 100)
Yfit = spline(Xfit)

# plot the scatter data and the fitted curve
plt.scatter(X, Y)
plt.plot(Xfit, Yfit, color='red')
```

```
# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Scatter data  with fitted curve')
plt.show()

actual = np.array([])
iter = np.array([])
for i in range(len(Y)):
  if Y[i]<0.1:
    actual = np.append(actual,float(Y[i]))
    iter = np.append(iter,int(i))
print(len(actual))
print(iter
    )

# generate random noise and add it to the data
noise = np.random.normal(scale=0.1, size=len(X))
y_noisy = Y + noise

spline = make_interp_spline(X, y_noisy)

# evaluate the spline over a finer grid
xfit = np.linspace(X.min(), X.max(), 100)
yfit = spline(xfit)

# plot the noisy data
plt.scatter(X, y_noisy)
plt.plot(xfit,yfit,color = 'red')

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Scatter data with added noise')
plt.show()

plt.plot(xfit,yfit,color = 'red')

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Scatter data with added noise')
plt.show()

b_noisy = np.array([])
for i in range(len(actual)):
  if y_noisy[int(iter[i])]>0.1:
    b_noisy = np.append(b_noisy,float(y_noisy[int(iter[i])]))
print(len(b_noisy))

b_noisy2 = np.array([])
for i in range(len(actual)):
  if y_noisy[int(iter[i])]>0.02:

    b_noisy2 = np.append(b_noisy2,float(y_noisy[int(iter[i])]))
print(len(b_noisy2))

!pip install pykalman
!pip install filterpy

from pykalman import KalmanFilter

# apply the Kalman filter
kf = KalmanFilter(initial_state_mean=y_noisy[0], n_dim_obs=1)
filtered_states, _ = kf.filter(y_noisy)

# plot the  filtered data
plt.plot(X, filtered_states, label='Filtered Data')

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Noisy Data with Kalman Filter')

plt.legend()
plt.show()

b_filtered = np.array([])
for i in range(len(actual)):
  if filtered_states[int(iter[i])]>0.1:
    b_filtered = np.append(b_filtered,float(filtered_states[int(iter[i])]))
print(len(b_filtered))

R = (len(b_noisy)-len(b_filtered))/len(b_noisy)*100
print(R)

# apply the Kalman filter for 2nd time
kf = KalmanFilter(initial_state_mean=filtered_states[0], n_dim_obs=1)
filtered_states_1, _ = kf.filter(filtered_states)

# plot the  filtered data
plt.plot(X, filtered_states_1, label='Filtered Data')

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Noisy Data with Kalman Filter')
plt.legend()
plt.show()

b_filtered_cycle_1 = np.array([])
for i in range(len(actual)):
  if filtered_states_1[int(iter[i])]>0.1:
```

```
                                                   b_filtered_cycle_1                  =
np.append(b_filtered_cycle_1,float(filtered_states_1[int(iter[i])]))
print(len(b_filtered_cycle_1))

# apply the Kalman filter for 3rd time
kf = KalmanFilter(initial_state_mean=filtered_states_1[0], n_dim_obs=1)
filtered_states_2, _ = kf.filter(filtered_states_1)

# plot the  filtered data
plt.plot(X, filtered_states_2, label='Filtered Data')

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Noisy Data with Kalman Filter')
plt.legend()
plt.show()

b_filtered_cycle_2 = np.array([])
for i in range(len(actual)):
  if filtered_states_2[int(iter[i])]>0.1:
                                                   b_filtered_cycle_2                  =
np.append(b_filtered_cycle_2,float(filtered_states_2[int(iter[i])]))
print(len(b_filtered_cycle_2))

# apply the Kalman filter for 4th time
kf = KalmanFilter(initial_state_mean=filtered_states_2[0], n_dim_obs=1)
filtered_states_3, _ = kf.filter(filtered_states_2)

# plot the  filtered data
plt.plot(X, filtered_states_3, label='Filtered Data')

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Noisy Data with Kalman Filter')
plt.legend()
plt.show()

b_filtered_cycle_3 = np.array([])
for i in range(len(actual)):
  if filtered_states_3[int(iter[i])]>0.1:
                                                   b_filtered_cycle_3                  =
np.append(b_filtered_cycle_3,float(filtered_states_3[int(iter[i])]))
print(len(b_filtered_cycle_3))

# apply the Kalman filter for 5th time
kf = KalmanFilter(initial_state_mean=filtered_states_3[0], n_dim_obs=1)
filtered_states_4, _ = kf.filter(filtered_states_3)

# plot the  filtered data
plt.plot(X, filtered_states_4, label='Filtered Data')
```

```
# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Noisy Data with Kalman Filter')
plt.legend()
plt.show()

b_filtered_cycle_4 = np.array([])
for i in range(len(actual)):
  if filtered_states_4[int(iter[i])]>0.1:
                                                b_filtered_cycle_4                =
np.append(b_filtered_cycle_4,float(filtered_states_4[int(iter[i])]))
print(len(b_filtered_cycle_4))

# apply the Kalman filter for 6th time
kf = KalmanFilter(initial_state_mean=filtered_states_4[0], n_dim_obs=1)
filtered_states_5, _ = kf.filter(filtered_states_4)

# plot the  filtered data
plt.plot(X, filtered_states_5, label='Filtered Data')

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Noisy Data with Kalman Filter')
plt.legend()
plt.show()

b_filtered_cycle_5 = np.array([])
for i in range(len(actual)):
  if filtered_states_5[int(iter[i])]>0.1:
                                                b_filtered_cycle_5                =
np.append(b_filtered_cycle_5,float(filtered_states_5[int(iter[i])]))
print(len(b_filtered_cycle_5))

# apply the Kalman filter for 7th time

kf = KalmanFilter(initial_state_mean=filtered_states_5[0], n_dim_obs=1)
filtered_states_6, _ = kf.filter(filtered_states_5)

# plot the  filtered data
plt.plot(X, filtered_states_6, label='Filtered Data')

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Noisy Data with Kalman Filter')
plt.legend()
plt.show()

b_filtered_cycle_6 = np.array([])
```

```
for i in range(len(actual)):
  if filtered_states_6[int(iter[i])]>0.1:
                                                b_filtered_cycle_6                =
np.append(b_filtered_cycle_6,float(filtered_states_6[int(iter[i])]))
print(len(b_filtered_cycle_6))

# apply the Kalman filter for 8th time
kf = KalmanFilter(initial_state_mean=filtered_states_6[0], n_dim_obs=1)
filtered_states_7, _ = kf.filter(filtered_states_6)

# plot the  filtered data
# plot the  filtered data
plt.plot(Xfit, Yfit, color='blue')
plt.plot(xfit,yfit,color = 'lime', label= "Noisy Data")
plt.plot(X, filtered_states_7,color= 'red', label='Filtered Data')

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Noisy Data with Kalman Filter')
plt.legend()
plt.show()

b_filtered_cycle_7 = np.array([])
for i in range(len(actual)):
  if filtered_states_7[int(iter[i])]>0.1:
                                                b_filtered_cycle_7                =
np.append(b_filtered_cycle_7,float(filtered_states_7[int(iter[i])]))
print(len(b_filtered_cycle_7))

from scipy.interpolate import make_interp_spline

# fit a cubic spline to the data
spline = make_interp_spline(X, Y)

# evaluate the spline over a finer grid
xfit = np.linspace(X.min(), X.max(), 100)
yfit = spline(xfit)

# plot the scatter data and the fitted curve
plt.plot(xfit, yfit, color='red')
plt.vlines(x=[X[(int(iter[0]))],X[(int(iter[961]))]],ymin    =    0,ymax    =    1,
colors='aqua', ls='--', lw=2, label='vline_multiple - full height')
plt.plot(xfit, yfit, color='red')

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Scatter data  with fitted curve')
plt.show()

print(X[int(iter[0])])
```

```
from scipy.interpolate import make_interp_spline

# fit a cubic spline to the data
spline = make_interp_spline(X, Y)

# evaluate the spline over a finer grid
xfit = np.linspace(X.min(), X.max(), 100)
yfit = spline(xfit)

# plot the scatter data and the fitted curve

plt.plot(xfit, yfit, color='red')
plt.hlines(y=[0.1],  xmin=[-1],  xmax=[90],  colors='aqua',  linestyles='--',  lw=2,
label = "10% threshold")

# add labels and title
plt.xlabel('time(in min)')
plt.ylabel('C/Co')
plt.title('Scatter data  with fitted curve')
plt.show()
```