

Connecting the Dots: A Graph-based Approach for Unsupervised Learning and Adaptive Process Monitoring with LLM-assisted Fault Diagnosis

Kyle Territo^a, Jose Romagnoli^{a*}

^a Louisiana State University, Department of Chemical Engineering, Baton Rouge, LA, United States

* Corresponding Author: jose@lsu.edu.

ABSTRACT

The convergence of artificial intelligence (AI) and chemical process systems engineering is creating unprecedented opportunities to transform current refineries from conventionally operated plants into intelligent, automated, and resilient systems. However, the practical deployment of AI in these complex industrial environments faces several critical challenges. First, most existing process datasets contain minimal labeled data, making it difficult to apply supervised learning techniques that require extensive annotations to generate meaningful insights. Furthermore, refinery data typically consists of high-dimensional, multivariate time series, which pose additional complexities in capturing temporal dynamics and system interactions. Traditional Fault Detection and Diagnosis (FDD) frameworks often struggle to address these complexities, lacking adaptability to evolving process conditions, scalability to large plant networks, and explainability in their diagnostic reasoning. To address these gaps, we propose an integrated framework for unsupervised knowledge discovery, adaptive process monitoring, and AI-assisted fault diagnosis. The approach combines Dynamic Time Warping (DTW) with graph-based layout algorithms to extract and visualize temporal structures and similarities within complex process datasets. The resulting graphs function as an unsupervised model, enabling the identification of operational regimes through process clustering. These models can be deployed online to detect anomalies using adaptive fault thresholding, ensuring robustness to changing process dynamics. When faults are detected, the framework leverages large language models (LLMs) as AI assistants to support fault diagnosis. By bridging domain knowledge—such as piping and instrumentation diagrams (P&IDs) and signal flow diagrams—with data-driven insights, including machine learning metrics and feature contribution scores, the system delivers explainable and accurate diagnostic outputs. This hybrid methodology enhances interpretability and decision-making, offering a scalable and adaptive solution for next-generation AI-enabled refinery operations.

Keywords: Unsupervised Learning, Fault Detection, LLMs, Machine Learning, Graph Networks

INTRODUCTION

Background and Motivation

Chemical refineries are among the most complex and highly automated industrial environments in modern manufacturing. Over several decades, these facilities have adopted advanced process control (APC) technologies—such as proportional–integral–derivative (PID) controllers, multivariable control systems, and model predictive control (MPC)—to maintain operational

efficiency, stability, and safety under tightly constrained process conditions. As a result, today's refineries operate with extensive digitization, high sensor density, and continuous streams of multivariate data originating from interconnected unit operations [1]. Despite this progress, current refinery optimization and monitoring practices still rely heavily on deterministic models, periodic calibrations, and human expertise to interpret process behavior and respond to abnormal conditions. These systems, while robust and well-established, show diminishing

returns as they approach their performance limits, motivating the need for more adaptive and intelligent data-driven solutions [1].

The emergence of Industry 4.0 has accelerated interest in leveraging artificial intelligence (AI), machine learning (ML), and advanced analytics to complement or extend the capabilities of conventional refinery control architectures. AI-driven approaches promise improved performance for nonlinear, high-dimensional, and dynamic systems by uncovering patterns not easily expressed in first-principles or optimization-based models. However, the practical implementation of AI in refinery environments remains challenging. Most downstream oil and gas facilities generate large quantities of operational data but very limited labeled data, restricting the use of supervised learning techniques that dominate the ML landscape [1]. Moreover, the datasets are typically highly-dimensional, multivariate time series, often with irregular sampling, complex cross-correlations, and contain evolving process dynamics such as catalyst degradation, feedstock variability, and equipment aging. These characteristics complicate the training of predictive models and reduce their robustness when deployed in real-world operations.

Prior work in industrial digitalization has shown that while Industry 4.0 technologies—such as IoT sensing, big data infrastructures, and cyber-physical systems—can provide transformative value, they must be carefully adapted for continuous process industries such as refining [1]. Traditional machine learning techniques, often developed for domains with abundant annotations and static data distributions often struggle with poor scalability to plant-wide networks, inadequate handling of non-stationary behavior, difficulty capturing temporal relationships, and a lack of explainability in diagnostic outcomes. These shortcomings contribute to slow adoption rates of AI-based FDD systems in safety-critical industrial environments.

At the same time, broader advancements in artificial intelligence highlight both the urgency and the opportunity for integrating more sophisticated AI techniques into process systems engineering. The 2025 Artificial Intelligence Index Report indicates that AI models are rapidly improving in efficiency, accessibility, and reasoning capabilities, with significant adoption growth across industrial sectors [3]. Industries worldwide are increasingly leveraging AI for predictive maintenance, anomaly detection, and decision support, yet concerns about data quality, transparency, and reliability remain prevalent. These concerns are particularly acute in refinery operations, where process deviations can propagate quickly and incur substantial economic or safety consequences.

Finally, sustainability pressures—including energy efficiency demands, emission reduction targets, and global competitiveness—further amplify the need for

intelligent, adaptive process monitoring solutions. As highlighted by Ghobakhloo, Industry 4.0 technologies can unlock substantial economic and environmental benefits, but only when deployed through architectures that support scalability, robustness, and meaningful knowledge generation [2]. These requirements align closely with the challenges facing modern refineries, where operators must navigate massive data volumes, interconnected process units, and evolving operational regimes.

Together, these factors illustrate a clear need for frameworks that can:

- (1) operate effectively in high-dimensional, minimally labeled data environments
- (2) capture complex temporal structures
- (3) adapt to shifting process conditions, and
- (4) provide explainable and actionable diagnostic insight for the refinery engineers

The framework proposed in this work directly addresses these gaps by integrating unsupervised knowledge discovery, adaptive anomaly monitoring, and AI-assisted diagnostic reasoning within a unified and interpretable architecture. From a process systems engineering perspective, the methodological contribution lies in structuring these components into a cohesive workflow that links data-driven temporal representations with adaptive monitoring and context-aware diagnostic reasoning to support real-time, plant-level decision making.

PROPOSED FRAMEWORK

Unsupervised Graph Construction

The first step towards building the unsupervised models is to shift the focus from individual time steps to windows of data. Before this step, the data were preprocessed through standard cleaning procedures, including handling of missing values and normalization to ensure consistent scaling across variables. Then, by analyzing these windows of data—as opposed to just pointwise data—the model can learn temporal dependencies from the sequences of observations. The parameters for this step are straightforward: a window size, W , and a stride, S . Window size is just the fixed length of data in each window and stride is the step size, or percentage of overlap, between consecutive windows. The choice of W and S determine both the temporal resolution of the data and ultimately the computational cost. The next step is to perform dynamic time warping (DTW) calculations in a pairwise fashion for each window of data. DTW is a widely used approach for assessing similarity between time-series sequences that do not share the same duration or pacing. First proposed by Itakura (1975) and later expanded by Sakoe and Chiba (1978), the technique

aligns two signals by flexibly stretching or compressing the time axis to achieve the minimum alignment cost, also known as the DTW distance [4-5]. The resulting alignment costs are then stored in a matrix as shown in **Figure 1**. These will be the basis for the graph model shown later. For these results, the *tslearn* [6] python package was utilized.

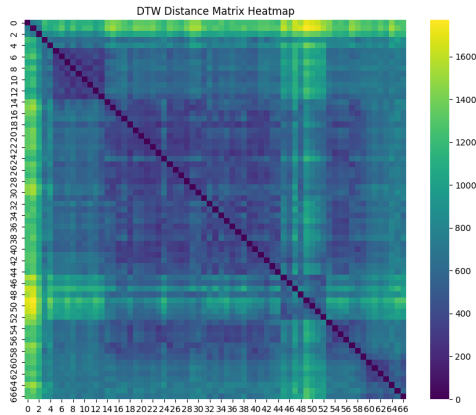


Figure 1. Heatmap of pairwise DTW distances between all windows of an example dataset

Once the DTW matrix is obtained, the next step is to represent the similarity structure from the distances as a graph. Utilizing the well-known k-NN algorithm, each window is treated as a node and edges are created by connecting each node to its k most similar neighbors—determined by the DTW distances. This creates a mapping that connects the temporal relationships within the dataset. These maps can be expressed as graphs by utilizing the *NetworkX* package in python [7]. An arbitrary example is shown in **Figure 2**.

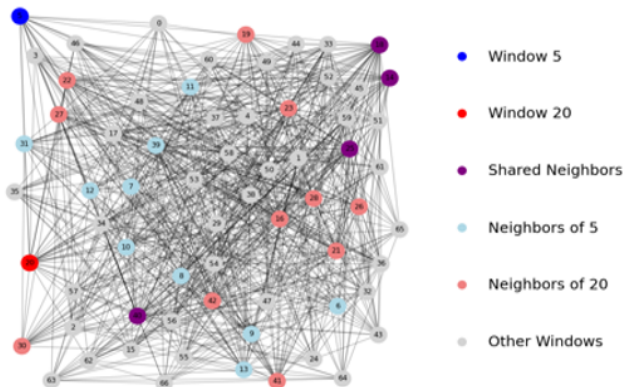


Figure 2. Initial graph highlighting nodes 5 and 20 connected by their k=15 neighbors

The issue now is that we have the mapping—however, it is just a tangled mess. It makes sense mathematically, but it serves no use yet as a visual graph. Since the goal is to ultimately cluster the windows based on

different operational regimes, the nodes need to be organized in such a way that this can be achieved. *NetworkX* has several graph layout algorithms that help with this arrangement. The spring layout, also referred to as the force-directed layout, was chosen as it is a physics inspired algorithm in which nodes are modeled as mutually repelling particles and edges act as attractive springs. The implementation in *NetworkX* utilizes the Fruchterman-Reingold [8] algorithm in which the node positions are iteratively updated until a low-energy, near-equilibrium state is reached or if the set number of iterations is achieved. The resulting graph clusters similar nodes closer together and sparsely connected nodes and outliers are placed towards the periphery. This can be seen in **Figure 3** when 50 iterations of the spring layout algorithm were applied. The most similar nodes to window 5 are mostly grouped near window 5 (blue cluster), the most similar nodes to window 20 are grouped near window 20 (red cluster), and some of the intermediate windows are placed at the boundary between the clusters (purple). At this point a graph-based embedding of the dataset is created, based upon the warped temporal distances. From here, traditional clustering techniques, KMEANS, DBSCAN, etc., could be applied to formally label the data from the unsupervised graphs generated.

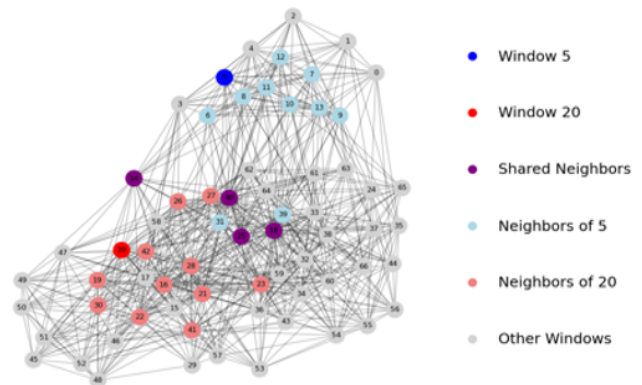


Figure 3. Resulting graph after 50 iterations of the force-directed algorithm was applied

Adaptive Online Monitoring

Since the goal of this framework is to expand beyond just offline modeling to online fault detection, the next step was to see how these graphs could be utilized online. First, the offline models need to be studied and the clusters need to be verified, perhaps by speaking with someone at the plant that is knowledgeable about changes in the process. When a normal operational regime is established, this cluster will be utilized as the baseline for the initial, online graph. Since this is considered normal operations, the natural variation among the distances between these windows will serve great purpose for calculating the initial fault detection threshold. By drawing from the normal cluster, the corresponding

DTW distances for each window can be collected and the distribution can be shown such as in **Figure 4**. From the distribution, statistical calculations can be performed to highlight the mean, standard deviation, percentiles, etc.

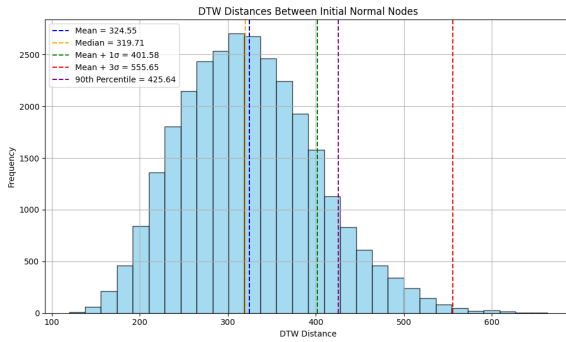


Figure 4: Distribution of pairwise DTW distances between initial normal nodes

Ultimately, the threshold was chosen to be adjustable based on the mean and standard deviations as shown in **Equation 1**. Here μ is the average of the distances, σ is the standard deviation, and X is a multiplier of the standard deviation. These parameters can be tuned based on the dataset at hand and the required fault detection sensitivity.

$$\text{Threshold} = \mu + X\sigma \quad (1)$$

To tackle the issue with evolving process conditions, the fault threshold is updated dynamically using an exponential moving average (EMA) of recent DTW values. This allows for the threshold to gradually adjust to changes in process conditions while still remaining sensitive to sudden anomalies. A parameter α is chosen as a smoothing factor that controls the responsiveness of the threshold to recent changes. Let μ_t and σ_t denote the running mean and standard deviation of current DTW distance d_t :

$$\mu_t = \alpha \cdot d_t + (1 - \alpha) \cdot \mu_{t-1} \quad (2)$$

$$\sigma_t = \alpha \cdot |d_t - \mu_t| + (1 - \alpha) \cdot \sigma_{t-1} \quad (3)$$

Then the adaptive threshold at time t becomes:

$$\text{Threshold}_t = \mu_t + X \cdot \sigma_t \quad (4)$$

If the current distance d_t exceeds the current threshold, the window is flagged as an anomaly. If a window is flagged as normal or if the current threshold is less than the initial threshold, then the threshold is updated. This allows the system to only learn during normal operations while also preventing the threshold from being stuck in a minimum. A third updating condition is applied if a fault persists across a specified number of consecutive windows. This mechanism will serve as an update trigger that will ensure new operating conditions are not persistently flagged as faults. By incorporating them into

the threshold learning, the model is prevented from becoming obsolete and can evolve as plant operations change—tackling one of the main objectives of this framework.

LLM Diagnostic Agents

Up to this point, two stages in the framework are achieved. First, the graph-based embeddings are created and utilized as an unsupervised model. Second, the graph model is prepared and subjected to live data for fault detection. The third stage is to extend to the fault diagnosis portion.

Fault diagnosis is tricky due to complex causal relationships, feedback loops, and misleading indicators. Traditional approaches—physics-based, expert systems, and data-driven ML—each fall short. Models may be rigid, they may be too complicated to create, or they may lack causal grounding. Data-driven methods such as Principal Component Analysis (PCA) contributions, Shapley Additive exPlanations (SHAP) analysis or Subspace Greedy Search (SGS) are often utilized due to their ease of use. By comparing a set of normal conditions to a set of faulty conditions, features with the most contribution can be highlighted. If these always worked, that would be great, however, these methods are purely black-box and often run into the “false contribution” problem. For example, a SHAP analysis may point to a feature that is highly important, but it is just a reactive symptom and not the true root cause of the fault. To overcome this, large language models (LLMs) are suggested as diagnostic agents.

The latest generation of LLMs—such as OpenAI’s ChatGPT models, Anthropic’s Claude models, or Google’s Gemini models, present a significant opportunity for advancing fault diagnosis in process monitoring systems. LLMs are trained on very large text corpora that make them excellent at predicting human like text sequences. In addition, the current stage of LLMs can handle multi-modal inputs such as tables, process diagrams, etc. This capability opens the door for LLMs that act as agents that can perform contextual reasoning that mirrors the diagnostic thought process of a human engineer. Venkatasubramanian and Chakraborty deem these context rich LLMs as “Large Knowledge Models (LKMs)” as opposed to the traditional “LLM” nomenclature [9]. These “LKM” agents can bridge the gap between the domain specific knowledge and data-driven ML outputs to support root cause identification, hypothesis generation, and human-in-the-loop decision making in industrial environments.

Overall Framework

The overall pipeline for this work is shown in **Figure 5**. This summarizes the three stages of the process monitoring workflow including the offline portion where historical data is utilized to build a DTW-based graph, an

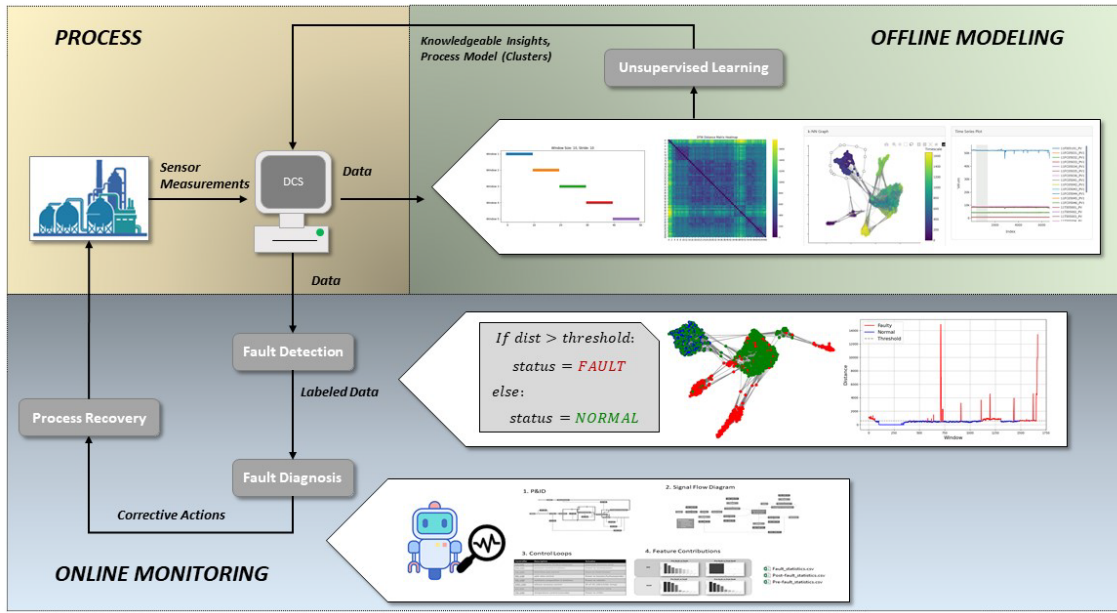


Figure 5. Overall Framework for Process Monitoring including the three stages: (1) Unsupervised Graph-based Learning, (2) Adaptive Fault Detection, and (3) LLM-assisted Fault Diagnosis

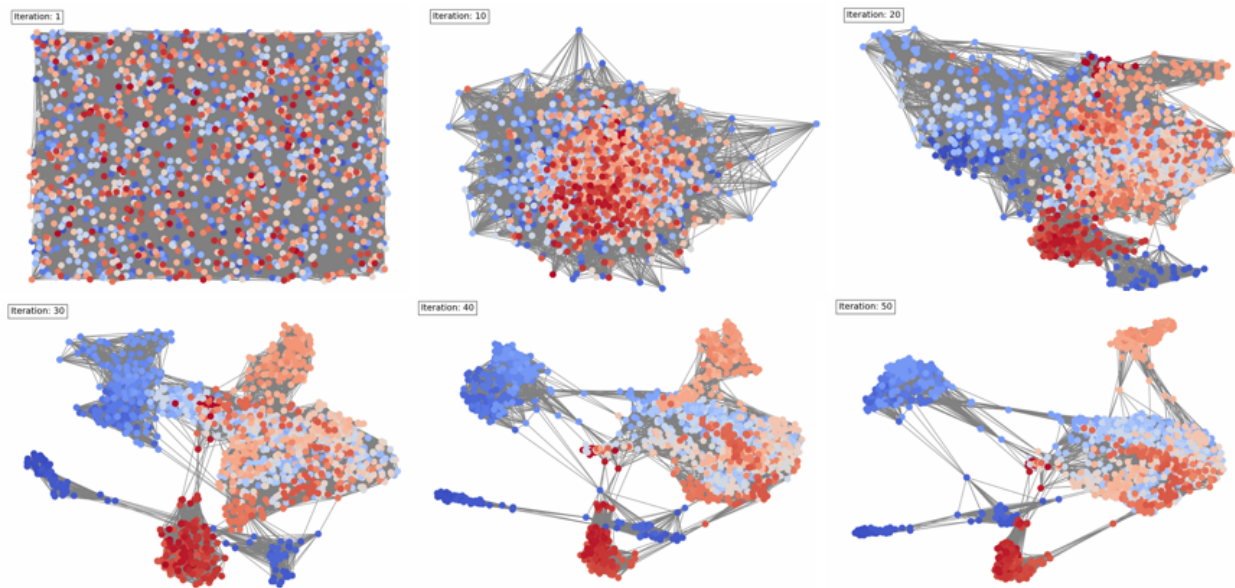


Figure 6. Graph evolution over successive iterations of the physics-based layout (colored by timescale).

online portion where live data can be subjected to the graph to generate fault labels while dynamically updating the fault threshold, and a diagnostic portion where LLM's serve as AI-assistants in real time diagnosis.

RESULTS

The framework was first tested for unsupervised process clustering and process monitoring using data

from an industrial-scale pyrolysis furnace that is involved in the production of ethylene from naphtha [10]. The reactor system is housed within a fired furnace where energy is supplied by burning fuel gas. The system consists of six reactor coils through which a mixture of naphtha and steam flows. The mixture is exposed to temperatures approaching 1000 °C, allowing for thermal cracking to produce ethylene. A total of 27 key process variables were monitored at 20-min intervals over a continuous

90-day operating period. Within the batch, there are known start-up and shutdown regimes, noise from a faulty sensor, and gradual drift due to accumulation of coke on the reactor tubes.

Building Graph Models

The dataset was split into windows of length 5 and a stride of 4, corresponding to window spans of 100 minutes with 20 minutes of overlap. This resulted in 1667 individual windows that were used to create the pairwise DTW matrix. A sensitivity analysis of the k parameter was performed and ultimately a value of $k=20$ was selected as it provided the most well-structured representation of the underlying graph. The evolution of the graph with $W=5$, $S=4$, and $k=20$ is shown in **Figure 6**. At iteration 1, the nodes are all randomly positioned. As iterations of the spring layout algorithm were applied, the similar nodes were slowly pulled toward each other and dissimilar nodes were pushed away from one another—resulting in an embedding with distinct regimes. Furthermore, the two-dimensional coordinates from the produced by the force directed layout were used as inputs to the HDBSCAN clustering algorithm in order to generate real cluster labels. The clustering hyperparameters—`min_cluster_size` and `min_samples`—were again tuned via a sensitivity analysis of the resulting graphs. The values of `min_cluster_size` and `min_samples` were eventually set to 5 and 10 respectively for the best result. Under this configuration, six distinct clusters were found, along with a small subset of nodes labeled as noise as shown in **Figure 7**. This cluster identification remained consistent after repeated trials as well.

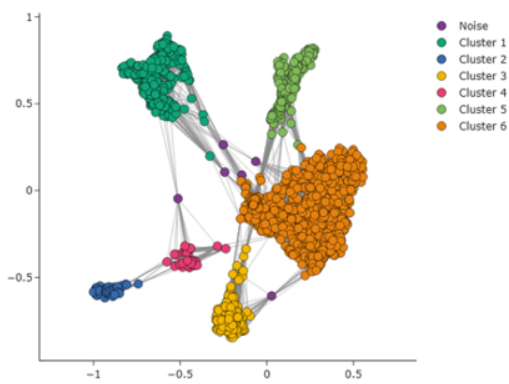


Figure 7. Process Model colored by Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) labels resulting in 6 clusters + noise

Evaluating Graphs for Online Monitoring

To extend the model for online analysis, Cluster 1 was deemed as the normal operational regime. The pairwise DTW distances for this cluster were collected and shown in **Figure 4**. Based on the statistical analysis, the

initial fault threshold was set at the mean plus three standard deviations. The graph was then initialized with only the nodes corresponding to Cluster 1. To monitor for faults, the full dataset was then subjected to this initial model and the results are shown in **Figures 8-9**. **Figure 8** shows the graph evolution and **Figure 9** shows the live run chart with the DTW distances for fault detection.

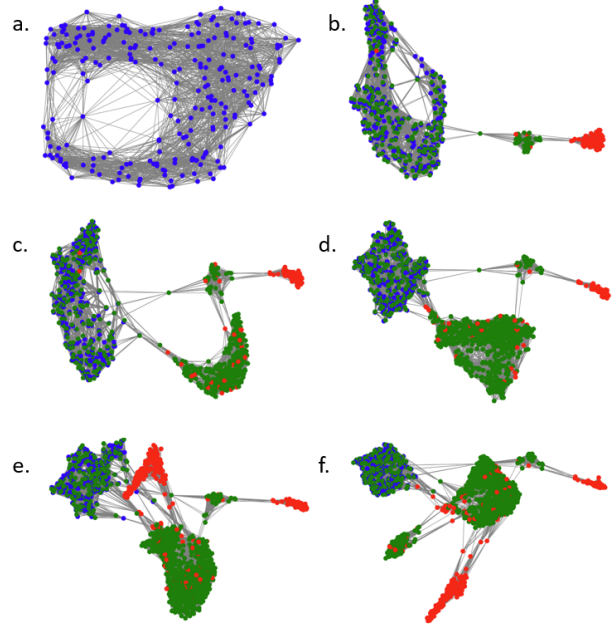


Figure 8. Evolution of the online graph for monitoring (BLUE is initial training nodes, GREEN is live normal nodes, and RED is live faulty nodes)

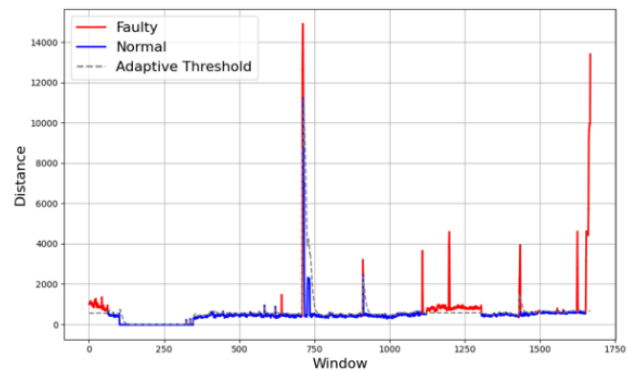


Figure 9. Online Fault Chart with Adaptive Threshold ($\alpha = 0.9, X = 3$)

From these results, several things were accomplished. First, the final resulting graph is similar to the initial graph that was built. This shows that the results are reproducible. Second, the faults that were found were accurate to what we know about the process. It flagged the start-up cluster, the spikes or noise cluster from the sensor noise, a separate operational period cluster, and the final shutdown cluster. Importantly, it did not classify

the process drift as noted in Cluster 6 as faults. This is due to the third point—that the online adaptations were successful. This can be better seen by zooming in on the run chart to see how the dynamic threshold was adapting. This can be observed in **Figure 10**.

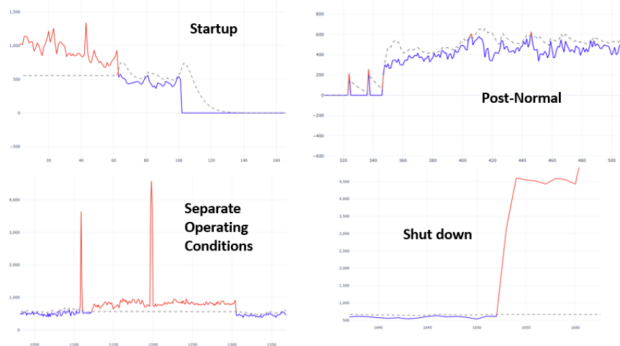


Figure 10. Zooming in on run chart to see the online threshold adaptations

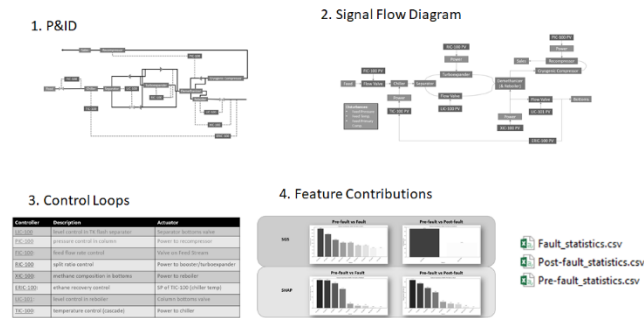


Figure 11. Inputs for Agent Prompting

Utilizing LLMs as Diagnostic Agents

In the final stage, we decided to utilize an ASPEN HYSYS® simulation for an LNG plant for the AI-assisted fault diagnosis. This was chosen since we had the ability to introduce our own faults into the simulation as opposed to relying on only historical datasets. This simulation monitors the operation of a demethanizer column operating under CRR separation conditions. Notably, the simulation has a feed stream defined as natural gas at 4980 kmol/hr supplied at a pressure of 5818 kPa and a temperature of 35 °C. The central unit is a 30-stage distillation tower with an integrated reboiler. A dataset was first created by introducing several faults to the system after running at normal conditions. These faults included a valve malfunction, a sensor bias, a feed disturbance, and a set-point (SP) change. Just like before, the graph model was constructed and the fault threshold was set. Now, when the model was online, the AI-assistant was incorporated. When a set number of faults were labeled, a snapshot of the data was taken and several inputs were collected. These inputs correspond to contextual information such as the P&ID diagram, signal flow diagram, and control loop table as well as the data-driven

metrics—the feature contributions from the fault and statistics from the data as shown in **Figure 11**.

These inputs were then sent to the AI-assistant, via OpenAI's API system and using the GPT-4o model. The agent was instructed to use the domain knowledge and the data-driven knowledge to find the root cause of the fault. Specifically, it was prompted to produce a one-page diagnostic report that identifies the root cause, highlights insights, grounds reasoning in domain knowledge, and recommends operator actions. Using this setup, the agent was tested for several fault scenarios. Here, the results are shown for the valve malfunction in **Figures 12-13**.

GPT Response (VALVE MALFUNCTION):

Fault Type: Control Error

Evidence from inputs:

- The OP monitoring metrics indicate that FIC-100 is a significant candidate for control loop errors detected by OP trends.
- In the SGS analysis, XIC-100_OP has the highest contribution (18.95%), followed by PIC-100_OP (12.98%) and TIC-100_OP (12.19%).
- However, FIC-100_OP has a very low contribution (0.26%), indicating it is not the root cause.
- The SHAP analysis also shows that FIC-100_OP has a contribution of 10.07%, which is not significantly larger than the others.

Root Cause Explanation:

- The control error is likely due to a malfunction in the FIC-100 control loop, which is responsible for maintaining a stable feed flow rate into the process.
- Given that no significant OP metrics were detected, it suggests that the feed valve may not be responding correctly to the control signals, leading to instability in the feed flow.
- This could be due to a valve malfunction, such as sticking or improper calibration, or a sensor bias affecting the flow measurement.

Recommended Next Steps:

- Operators should inspect the FIC-100 control loop, focusing on the feed valve and its associated flow sensor.
- Calibration checks should be performed, and the valve should be tested for proper operation.
- If necessary, maintenance should be conducted to ensure the valve is functioning correctly and that the sensor readings are accurate.
- Additionally, monitoring should continue to assess the impact of any adjustments made.

Figure 12. Diagnostic report generated by LLM for Valve Malfunction

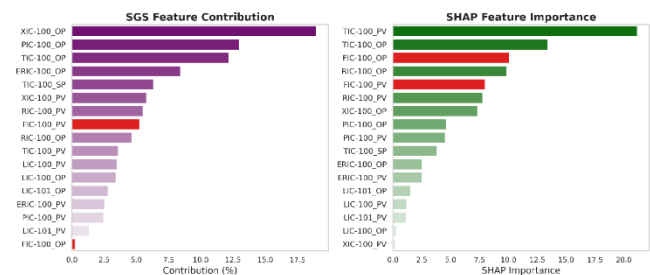


Figure 13. Feature Contributions for SGS and SHAP (true fault features in RED)

In this fault, the feed control valve (FIC-100) was

plugged by decreasing the valve coefficient by 10%. The agent analyzed the inputs that were collected and, in the report, accurately described the fault. It began by first discussing evidence from the inputs. For example, it said that “FIC-100 is a significant candidate for control loop errors detected by OP trends”. It then went on to say that FIC-100 was not highly ranked in the feature contributions for both SHAP and SGS, causing a discrepancy. It then went on to the root cause hypothesis. Here it ultimately decided that there was most likely a “malfunction in the FIC-100 control loop.” Following this, it suggested corrective actions such as the valve should be “tested for proper operation” and specifically “focus on the feed valve and its associated flow sensor.”

Overall, this was a pretty good report as the agent only had knowledge based on its initial knowledge and the knowledge learned from the inputs. The prompt included detailed instructions, but no actual information of what faults occurred. In addition, it is important to note that just the standard SGS and SHAP plots would not have captured the true root cause of the fault as it highlighted the FIC-100 control loop first only as the 8th most important in the SGS plot and 3rd most important in the SHAP plot.

CONCLUSION

In summary, the proposed framework “connects the dots” across key challenges in process monitoring by addressing limited labeled data, capturing temporal dynamics in multivariate time series, adapting to changing process conditions through a dynamic fault threshold, and autonomously generating root-cause hypotheses using LLMs. Together, these elements form a unified, interpretable, and scalable approach for modern refinery operations. The results show that DTW-based graph embeddings effectively reveal operational regimes, the adaptive thresholding reliably separates true faults from natural drift, and the LLM-assisted diagnosis produces context-aware explanations grounded in both domain knowledge and data-driven evidence. Future work will aim at addressing scalability concerns, prompting strategies, and collecting more quantitative results for both the detection and diagnosis portions of the framework. Overall, this work demonstrates the value of integrating unsupervised learning, adaptive monitoring, and AI-assisted diagnosis into a cohesive workflow. Future work will refine LLM prompting strategies, expand multimodal inputs, and evaluate the framework’s scalability for broader industrial deployments.

AUTHOR IDENTIFIERS

Author ORCIDs:

Territo K: [0009-0002-1133-915X](https://orcid.org/0009-0002-1133-915X)

Romagnoli JA: [0000-0003-3682-1305](https://orcid.org/0000-0003-3682-1305)

REFERENCES

1. Olaizola I, Quartulli M, Unzueta E, Goicolea J, Flórez J. Refinery 4.0, a review of the main challenges of the industry 4.0 paradigm in oil & gas downstream. *Sensors* 22:9164 (2022). <https://doi.org/10.3390/s22239164>
2. Ghobakhloo M. Industry 4.0, digitization, and opportunities for sustainability. *Journal of Cleaner Production* 252:119869 (2020). <https://doi.org/10.1016/j.jclepro.2019.119869>
3. Maslej, N., Fattorini, L., Perrault, R., Gil, Y., Parli, V., Kariuki, N., Capstick, E., Reuel, A., Brynjolfsson, E., Etchemendy, J. and Ligett, K., 2025. Artificial intelligence index report 2025. arXiv preprint arXiv:2504.07139.
4. Itakura F. Minimum prediction residual principle applied to speech recognition. *IEEE Trans. Acoust., Speech, Signal Process.* 23:67-72 (1975). <https://doi.org/10.1109/tassp.1975.1162641>
5. Sakoe H, Chiba S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust., Speech, Signal Process.* 26:43-49 (1978). <https://doi.org/10.1109/tassp.1978.1163055>
6. Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K. and Woods, E., 2020. Tslern, a machine learning toolkit for time series data. *Journal of machine learning research*, 21(118).
7. Hagberg AA, Schult DA, Swart PJ. Exploring network structure, dynamics, and function using networkx. *Proceedings of the Python in Science Conference* :11-15 (2008). <https://doi.org/10.25080/tcww9851>
8. Fruchterman T. M. J., Reingold E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11), 1129–1164.
9. Venkatasubramanian V, Chakraborty A. Quo vadis chatgpt? from large language models to large knowledge models. *Computers & Chemical Engineering* 192:108895 (2025). <https://doi.org/10.1016/j.compchemeng.2024.108895>
10. Romagnoli JA, Briceño-Mena L, Manee V. AI in chemical engineering. CRC Press (2024). <https://doi.org/10.1201/9781003455905>

© 2026 by the authors. Licensed to PSEcommunity.org and PSE Press. This is an open access article under the creative commons CC-BY-SA licensing terms. Credit must be given to creator and adaptations must be shared under the same terms. See <https://creativecommons.org/licenses/by-sa/4.0/>

