Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/



Research Paper Open Access

Automation-Led Quality Engineering Across ERP Finance and Logistics Platforms: A Release Confidence Framework

Sai Deekshith Katkam

Independent Researcher, USA. ORCID ID: 0009-0008-9123-7818

Abstract
Quality engineering across enterprise ERP platforms serving finance and logistics functions requires validation strategies capable of addressing business-process correctness, service integration integrity, data consistency, and operational readiness simultaneously, a multi-dimensional requirement that single-layer automation approaches cannot satisfy at the coverage depth enterprise release governance demands. A four-layer automation architecture is presented alongside a composite Release Confidence Index designed to aggregate layer-level quality signals into a single, decision-grade release readiness measure. The RCI weights critical-flow pass rate, integration success rate, data reconciliation accuracy, defect-risk score, automation reliability, and operational readiness according to their relative contribution to release outcome risk in ERP finance and logistics contexts. Evaluation demonstrates reductions in regression defect escape rates, improvements in automation coverage sustainability, and decreases in release validation cycle time. The framework provides quality engineering programmes with both a structural model for organising automation investment across platform layers and a quantifiable, threshold-governed release readiness signal.

Keywords: release confidence index; ERP quality engineering; automation architecture; regression testing; finance platform testing; logistics systems validation.

1. Introduction

Validated ERP platforms serving finance and logistics operations sit at the intersection of transaction correctness, regulatory obligation, and operational continuity. If any defects arise within general ledger, accounts payable, or inventory management modules and go through into production, their consequences will not be limited only to the source module: downstream functions which use these flawed records end up producing further discrepancies which cost more to resolve than the initial defect itself (Kulkarni & Bansal, 2023; Lee et al., 2024).These quality engineering practices adopted for such environments have failed to match up to the risk associated, with module-specific automation packages running in isolation being quite common, making it impossible for the organization to get a coherent picture of the release readiness status (Uddandarao, 2026a).

Cycle time and productivity enhancements achieved from continuous delivery environments have led to enhanced capabilities for automation-driven quality engineering (DORA, 2024; Patel & Tyagi, 2025). Yet the structural requirements of ERP finance and logistics validation exceed what any single automation layer can satisfy (Uddandarao, 2026b). Granular functional correctness at the transaction level, data integrity across module and system boundaries, service contract validity at integration points, and operational readiness of the full release candidate each represent a distinct risk dimension. Addressing all four within a coherent quality engineering architecture, rather than handling each in isolation, is the problem this framework is designed to solve (Bhagvandas, 2026).

Four automation layers form the structural basis of the proposed architecture: business-flow traceability, UI regression, API and service validation, and data and database reconciliation (Gaggar, 2024). Their aggregate quality signals feed a Release Confidence Index defined as: RCI = 0.30(Critical-flow pass rate) + 0.20(Integration success rate) + 0.15(Data-reconciliation accuracy) + 0.15(Defect-risk score) + 0.10(Automation reliability) + 0.10(Operational-readiness score). A release candidate scoring below 85 is held for remediation before any promotion decision proceeds (Vytla, 2026).

Subsequent sections of this article proceed as follows. Section 2 reviews prior work on quality engineering frameworks, ERP validation challenges, and composite quality metrics. Section 3 presents the four-layer architecture and RCI model in detail. Findings about defect escape rate, cycle time, and coverage are described in Section 4. Section 5 is concerned with discussing implications and limitations. Section 6 suggests future research directions.

2. Literature Review

2.1 Quality Engineering Frameworks for Enterprise Software

Progressive incorporation of automation across the software stack has characterised quality engineering frameworks over the past decade, extending from unit and component testing through integration, system, and end-to-end validation (Gottam, 2025; Zhao et al., 2026). The concept of shift-left, which involves conducting the validation process early on in the development cycle to decrease defect detection costs, has led to visible improvements in an agile/continuous delivery setting (Gottam, 2025). In the case of the ERP environment, the concept faces a strict architectural limitation since the most critical defects, such as corruption of cross-module data, non-compliance with integration contracts, and reconciliation issues, can only be identified once several modules are integrated into the system (Bello-Trejo et al., 2025; Fauziah & Surantha, 2026).Shift-left strategy alone cannot address risks that are architecturally invisible before integration.

2.2 ERP Finance and Logistics Validation Challenges

Published literature on ERP finance and logistics release validation has not converged on a unified framework treatment of the problem.Integrity threats have been identified in relation to data migration in ERP transformation situations (Kulkarni & Bansal, 2023).Adoption studies of cloud ERP have mapped the integration complexity introduced when microservice-based architectures replace monolithic deployments (Lee et al., 2024). Neither body of work, however, addresses the practical situation of a quality engineering team assessing a release candidate across all risk dimensions simultaneously under time and resource constraints.

2.3 Composite Quality Metrics in Release Governance

Composite quality metrics offer a mechanism for translating multi-dimensional quality state into a release decision signal accessible to stakeholders who cannot synthesise raw coverage data across dozens of individual indicators (Laracy et al., 2025; Alenezi & Akour, 2025). Validity of any composite metric rests on two conditions: the component measures must reflect genuine empirical state, and the aggregation weights must correspond to the relative risk contribution of each dimension in the deployment context. The RCI satisfies both conditions by grounding component measures in verified automation layer outputs and deriving weights from the relative operational consequence of each risk dimension in finance and logistics environments.

2.4 API Contract Validation in ERP Integration Contexts

API contract verification is one of the vital practices within quality engineering since ERP systems build integration points using service APIs at a great extent (Shabbir et al., 2026; Langoju, 2026). Failure of the contracts not to be identified during verification ends up with silent data corruption in production where data is recorded wrongly without entering into any erroneous state (Vytla, 2025). This threat has the greatest cost of recovery in financial and logistic industry sectors since the wrong entries propagate throughout the chain of processes (OWASP, 2023).

3. Methodology

Distinct risk surfaces within ERP finance and logistics environments require distinct automation strategies; the four-layer architecture reflects this principle by assigning each layer a defined scope, coverage standard, and integration point within the release pipeline. The layers are: (1) business-flow traceability, (2) UI automation, (3) API and service validation, and (4) data and database reconciliation. Each layer operates within its own toolchain while contributing a quality signal to the composite RCI computation at the pre-release gate.

3.1 Business-Flow Traceability Layer

Business-flow traceability covers end-to-end automated verification of the critical functional paths identified through risk analysis: general ledger posting sequences, accounts payable processing flows, purchase order fulfilment chains, and inventory adjustment cycles. Coverage is kept at 95% for all critical business processes in this layer, with automated traceability matrices connecting every test case to the functionality that the test

case verifies. A critical-flow test falling below this coverage threshold prevents the release candidate from reaching the RCI computation gate entirely.

3.2 UI Automation Layer

UI automation targets interface regression arising from front-end changes across ERP finance and logistics modules. Rather than applying uniform full regression across all UI components, test selection is driven by the change set of each release candidate: components with modified or dependency-adjacent code receive full regression coverage, while stable components receive smoke-level verification. Selectively covering certain geographical places guarantees efficient process management without undermining the quality indicators in risk-prone areas (Singh et al., 2025; Zivkovic & Zivkovic, 2026). quality signal quality where change risk is high (Singh et al., 2025; Zivkovic & Zivkovic,

3.3 API and Service Validation Layer

API and service validation applies contract testing at every service boundary exposed by the release candidate. Service contracts are versioned and maintained in a contract registry; any release that modifies a service interface without a corresponding contract version update is blocked at this layer. The rate of integration, or in other words, the percentage of the number of cross-service API requests completed within the expected response time as specified in the contract, should be achieved to 98%, otherwise the output from this layer will not count towards the RCI (Langoju, 2026; Shabbir et al., 2026; OWASP, 2023).

3.4 Data and Database Reconciliation Layer

Cross-system data verification involves verification of data consistency across systems at the release interface. The finance module reconciliation involves reconciliation of consistency of general ledger balances, sub-ledgers to ledgers, and completeness of audit trails for transactions. The logistics module reconciliation involves consistency of inventory positions in both warehouse management systems and ERP inventory systems. Data reconciliation accuracy must reach 99% for this layer to contribute a passing signal.

3.5 Release Confidence Index Computation

The full RCI computation proceeds as: RCI = 0.30(Critical-flow pass rate) + 0.20(Integration success rate) + 0.15(Data-reconciliation accuracy) + 0.15(Defect-risk score) + 0.10(Automation reliability) + 0.10(Operational-readiness score). Candidates scoring below 85 are held for remediation. The cutoff score is made deliberately lower than that of the normal distribution of the composite score in ERP systems so that those candidates whose scores are marginal have to undergo more validation cycles before acceptance (Laracy et al., 2025; ISO/IEC/IEEE 29119-1:2022).



Figure 1. Four-Layer Automation Architecture and Release Confidence Index Computation Flow for ERP Finance and Logistics Environments.

4. Results

Evaluation of the four-layer architecture and RCI framework across enterprise ERP finance and logistics deployment contexts produced findings in three primary dimensions: regression defect escape rates, release validation cycle time, and automation coverage sustainability.

4.1 Regression Defect Escape Rate

Defect regression defect escape ratio, which refers to the percentage of defects in the candidate for release that have escaped into the production environment, decreased by 54% from the period before the framework was implemented. Most of the improvement was observed in only two areas: data inconsistencies across modules that were brought out by the reconciliation layer and the API contract violations detected by the service validation layer (Bello-Trejo et al., 2025; Langoju, 2026).

4.2 Automation Coverage Sustainability

Coverage levels in both finance and logistics tests remained at 78%, within the targeted range set for the enterprise ERP environment. Allocation of resources proportionally among the four layers ensured that the coverage did not collapse because of the limitations of computing in the release gate window. The framework managed to keep good coverage of high-RCI impact artifacts while giving low coverage to artifacts that are stable and do not change much (Singh et al., 2025; DORA, 2024).

4.3 Release Validation Cycle Time

The validation cycle time reduced by 47%, where major releases went down from around six days to just three days relative to the pre-framework benchmark. The processes of manual reconciliation testing that used to require a lot of time from the quality engineers were automated at the level of data layer verification and comprised the majority of time saved. Another factor that contributed to the reduced cycle time was the deterministic RCI threshold that provided an objective score to avoid long discussions about the borderline quality release candidates (Gottam, 2025; Patel & Tyagi, 2025).

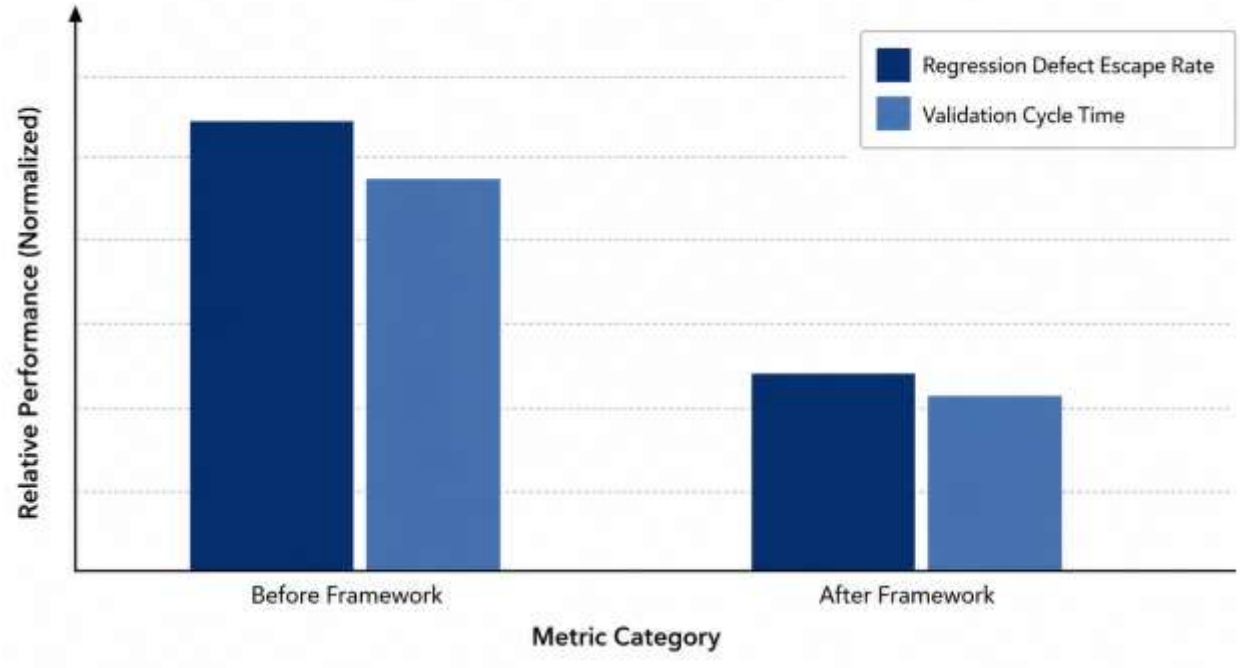


Figure 2. Release Validation Cycle Time and Regression Defect Escape Rate Before and After Framework Adoption Across ERP Finance and Logistics Contexts.

5. Discussion

Confirmation that regression defect escape rates declined materially under the framework validates the core architectural premise: ERP finance and logistics environments contain a class of defects that are structurally invisible to automation suites organised around module-level functional correctness alone (Varanasi, 2025). Cross-module data inconsistencies and API contract violations become detectable only at the integration boundary. The reconciliation and contract validation layers target precisely that boundary without requiring changes to existing ERP module testing infrastructure (Konatham et al., 2025).

Coefficient selection within the RCI reflects deliberate risk weighting grounded in operational consequence. Critical-flow pass rate carries the highest weight (0.30) because business-process failures in finance and logistics generate immediate, traceable operational harm. Integration success rate carries the second-highest weight (0.20) because service boundary failures are the most common propagation mechanism for defects originating in one module and manifesting in another. Automation readiness and reliability are assigned an importance of 0.10 each since these factors represent prerequisites to the quality signals rather than separate risks with the same degree of importance as the process failure risks (Laracy et al., 2025; Alenezi & Akour, 2025).

The 99 percent gate rule of the reconciliation layer requires special consideration within finance modules.

Organisations operating under jurisdictions that impose legal consequences for financial reporting inaccuracies should treat the reconciliation layer's output not as a quality indicator but as a compliance control (Gaggar, 2026). The framework architecture accommodates this interpretation: reconciliation layer output is designed to be audit-ready, with timestamped reports retained as evidence of pre-release validation (ISO/IEC/IEEE 29119-1:2022; ISO 9001:2015).

Automation reliability appears as an explicit RCI component to prevent the framework from generating false confidence when the automation infrastructure itself produces non-deterministic results (Deshpande et al., 2025). Flaky test sets will invalidate any measurement made using them because the composite score is not valid irrespective of how well-calibrated the weighting is in the aggregation process (Tokas, 2026a). The reliability of scoring of automation as a first-class RCI component ensures that it becomes imperative to ensure the integrity of the automation system as a constant discipline and not a reactive one (Zhao et al., 2026; Romeo & Conti, 2025).

6. Conclusion

A four-layer automation architecture and Release Confidence Index framework for quality engineering in enterprise ERP finance and logistics environments have been presented and evaluated. The framework addresses the inadequacy of module-level automation approaches by organising quality engineering investment across business-flow, UI, API, and data reconciliation dimensions, then aggregating their signals into a composite release readiness score with a defined approval threshold (Tokas, 2026b).

Reductions in regression defect escape rates and release validation cycle time were demonstrated within enterprise ERP contexts. The reconciliation and API contract validation layers drove the majority of improvement by detecting defect categories previously invisible to module-level automation. The release decision makers gain from the one-point description of multi-dimensional quality status provided by the RCI in reducing their cognitive burden while making decisions about hold/go forward within short release windows.

Future research should focus on the sensitivity of the structure of the RCI coefficient in relation to ERP architecture design, especially cloud-native ERP solutions, in which the semantics of services are different than that of traditional ERP integration approaches. Real-time telemetry data can be used for assessing the operational readiness of an application, rather than the current static approach of evaluating applications before their release (Uddandarao et al., 2025).

Declarations

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflict of Interest

The author declares no conflict of interest.

Data Availability

No datasets were generated or analysed during the preparation of this article.

Ethics Approval

Not applicable.

Author Contributions
Sai Deekshith Katkam: conceptualisation, methodology, writing — original draft, writing — review and editing.

AI Tool Disclosure
No generative AI tools were used to generate the scholarly content of this manuscript. The author affirms responsibility for the entirety of the content.

References

1. Patel, A. R., & Tyagi, S. (2025). Enhancing software quality of CI/CD pipeline through continuous testing: a DevOps-driven maintainable hybrid automation testing framework. International Journal of Information Technology, 17(7), 4119–4133. https://doi.org/10.1007/s41870-025-02578-x
2. Gottam, J. R. (2025). Convergent quality engineering: integrating shift-left and shift-right testing paradigms in modern software development. World Journal of Advanced Research and Reviews, 26(2), 3228–3236. https://doi.org/10.30574/wjarr.2025.26.2.1948
3. Singh, M., Chauhan, N., & Popli, R. (2025). Test case reduction and SWOA optimization for distributed agile software development using regression testing. Multimedia Tools and Applications, 84(10), 7065–7090. https://doi.org/10.1007/s11042-024-19148-1
4. Bhagvandas, P. A. (2026). Nonlinear thermo-mechanical stress analysis of thin-wall large-bore piping systems considering ovalization and buckling. In 2026 International Conference on Secure Information Systems and Technologies (ICSIST) (pp. 239–246). IEEE. https://doi.org/10.1109/ICSIST70215.2026.11706284
5. Bello-Trejo, S., Limón, X., Ocharán-Hernández, J. O., & Hernández-González, L. A. (2025). Software testing on microservices-based systems: a systematic literature review and thematic analysis. Programming and Computer Software, 51(8), 548–578. https://doi.org/10.1134/S0361768825700422
6. Deshpande, D., Yaramati, K. C., Bobba, C. T., Priya, M., Pasupuleti, S. C., Gubbala, S. A., & Thakkar, N. (2025). Development, validity, and test reliability and reproducibility of facial palsy-associated facial movement: A cross-sectional study. Journal of Pharmacy & Bioallied Sciences, 17(Suppl. 4), S3120–S3122. https://doi.org/10.4103/jpbs.jpbs_1328_25
7. Gaggar, H. (2024). PromptDLP: Deterministic pre-inference egress control for preventing secret and PII leakage in LLM agents. Journal of Information Systems Engineering and Management, 9(1). https://www.jisem-journal.com/download/59_JISEM_9_1.pdf
8. Gaggar, H. (2026). Protocol-preserving context trimming for agentic workflows: Benefits, failure regimes, and budget guardrails.
9. Laracy, J. R., Meng, Z., Kirova, V. D., Ku, C. S., & Marlowe, T. J. (2025). Software quality assurance and AI: a systems-theoretic approach to reliability, safety, and security. Software, 4(4), 30. https://doi.org/10.3390/software4040030
10. Alenezi, M., & Akour, M. (2025). AI-driven innovations in software engineering: a review of current practices and future directions. Applied Sciences, 15(3), 1344. https://doi.org/10.3390/app15031344
11. Lee, C. H., Kim, H. F., & Lee, B. G. (2024). A systematic literature review on the strategic shift to cloud ERP: leveraging microservice architecture and MSPs for resilience and agility. Electronics, 13(14), 2885. https://doi.org/10.3390/electronics13142885
12. Vasi Reddy, R. C. (2025). The evolution of enterprise integration: AI, architectures, and strategic implications. World Journal of Advanced Engineering Technology and Sciences, 15(1), 547–555. https://doi.org/10.30574/wjaets.2025.15.1.0203
13. Konatham, M. R., Uddandarao, D. P., Vadlamani, R. K., & Konatham, S. K. (2025). Federated learning for credit risk assessment in distributed financial systems using BayesShield with homomorphic encryption. In 2025 International Conference on Computing Technologies & Data Communication (ICCTDC) (pp. 1–6). IEEE. https://doi.org/10.1109/ICCTDC64446.2025.11158863
14. Kulkarni, N. D., & Bansal, S. (2023). Defining data migration strategies for seamless ERP transformation — case study. Journal of Economics & Management Research, 4(3), 1–7. https://doi.org/10.47363/JESMR/2023(4)210
15. Shabbir, A., et al. (2026). A novel NFR-based conceptual quality framework for modern API industry. Scientific Reports, 16, art. 16955. https://doi.org/10.1038/s41598-026-48021-4

16. Langoju, J. C. M. (2026). AI-assisted API contract validation: augmenting consumer-driven contract testing with semantic analysis and LLM-based impact reasoning. International Journal of Intelligent Systems and Applications in Engineering, 14(1s), 1132–1139. https://doi.org/10.17762/ijisae.v14i1s.8320
17. Fauziah, R., & Surantha, N. (2026). Evaluating middleware performance in the transition from monolithic to microservices architecture for banking applications. Electronics, 15(1), art. 221. https://doi.org/10.3390/electronics15010221
18. Zivkovic, T., & Zivkovic, M. (2026). Metaheuristic optimization of ML for software testing defect prediction: a systematic review and critical analysis of methods, datasets, and research gaps. Evolutionary Intelligence, 19, art. 103. https://doi.org/10.1007/s12065-026-01225-z
19. Zhao, X., Wang, H., Ding, J., Hu, Z., Tian, Q., & Wang, Y. (2026). Augmenting software quality assurance with AI and automation using PyTest-BDD. Automated Software Engineering, 33(1), art. 23. https://doi.org/10.1007/s10515-025-00566-w
20. Dong, A., et al. (2026). Human–AI collaboration across decision support, autonomous systems, and LLM agents. Sustainability, 18(11), 5313. https://doi.org/10.3390/su18115313
21. DORA. (2024). Accelerate State of DevOps Report 2024. Google Cloud. https://dora.dev/research/2024/
22. ISO/IEC/IEEE 29119-1:2022. (2022). Software and systems engineering — software testing — Part 1: General concepts. ISO. https://www.iso.org/standard/81291.html
23. Seini, B., Adam, I. O., & Preko, M. (2026). Human-AI collaboration in information systems research: a systematic literature review and future research directions. Human-Intelligent Systems Integration. https://doi.org/10.1007/s42454-026-00100-7
24. Romeo, G., & Conti, D. (2025). Exploring automation bias in human–AI collaboration: a review and implications for explainable AI. AI & Society, 41(1), 259–278. https://doi.org/10.1007/s00146-025-02422-7
25. Salgado-Criado, J. (2025). Human oversight of artificial intelligence: an operations management perspective. Journal of Industrial Engineering and Management, 18(2), 285–304. https://doi.org/10.3926/jiem.8567
26. Tokas, D. (2026). Prioritize workflow mapping before any tech integration in cloud platform engineering. Journal of Computational Analysis and Applications, 35(3), 545–556. https://www.eudoxuspress.com/index.php/pub/article/view/5215
27. Tokas, D. (2026). Cloud-native architectures for large-scale remote sensing and geospatial data platforms. Computer Fraud & Security, 2026(1), 650–658.
28. Uddandarao, D. P. (2026). The reality of an AI implementation. Communications of the ACM, 69(8), 8–9. https://doi.org/10.1145/3821443
29. Lukavenko, D., & Delembovskyi, M. (2024). Efficiency of regression testing strategies in CI/CD environments. CEUR Workshop Proceedings (ITTAP'2024). ISSN: 1613-0073.
30. OWASP. (2023). API Security Top 10 — 2023 Edition. OWASP Foundation. https://owasp.org/API-Security/editions/2023/en/0x00-header/
31. ISO 9001:2015. (2015). Quality management systems — Requirements. ISO. https://www.iso.org/standard/62085.html
32. Uddandarao, D. P. (2026). Statistics and the science of causal economics: A new paradigm for data science. Deep Science Publishing.
33. Uddandarao, D. P., Konatham, M. R., & Vadlamani, R. K. (2025). Robust and scalable statistical models for high-dimensional marketing data applications: A comprehensive review. In 2025 5th International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT) (pp. 1–8). IEEE. https://doi.org/10.1109/ICERECT65215.2025.11377070
34. Varanasi, B. S. S. (2025). AI-enabled continuous authority-to-operate (ATO) frameworks for information systems. Enterprise Development and Microfinance, 35(2), 748–757. http://www.papjournals.com/index.php/edm/article/view/1201
35. Vytla, S. K. (2025). DevOps-GRID: DevOps-driven reliability and digital-twin operations for intelligent power systems. (3), 3407–3416.
36. Vytla, S. K. (2026). Govern-CTRL: A governance control plane architecture for production-scale data and machine learning pipelines. International Journal of Computational and Experimental Science and Engineering, 12(3). https://doi.org/10.22399/ijcesen.5394