G-Bridge: Real-Time Dissipation Tracking and Vortex-Stretching Screening

Dickson A Terrero[a]

[a]*Independent Researcher, Boston, USA*

Abstract

Motivation. High-fidelity computational fluid dynamics (CFD) can resolve complex flow features like viscous dissipation (ϕ_v) and vortex stretching (γ), both of which are critical indicators of flow quality in manufacturing. However, these diagnostics are not directly measurable on typical factory QA lines, which rely on simpler signals like pressure, temperature, vibration, and speed.

Results. We introduce *G-Bridge*, an open-source Python pipeline for inline quality assurance (QA) in pipe flow manufacturing. The system operates in two stages:

- **Stage 1** maps factory-style sensor data to $\log_{10} \phi_v$ (viscous dissipation rate).
- **Stage 2** performs screening and ranking of γ, a scalar measure of the local vortex-stretching/dissipation balance (s^{-1}) using a hurdle model: a classifier for detecting whether $\gamma > 0$, and a regressor for estimating $\log(1 + \gamma)$ on positives.

On 3D incompressible OpenFOAM data, evaluated with a held-out snapshot (file-level split), Stage 1 achieves $R^2 = 0.9994$ and MAE $= 1.90 \times 10^{-3}$ in $\log_{10} \phi_v$ (about 0.44% relative error). Stage 2 achieves AUC = 0.945 and AP = 0.991; on $\gamma > 0$ the regression attains $\rho = 0.981$, $R^2 = 0.352$ in $\log(1 + \gamma)$.

Software. G-Bridge is fully open source, with containerized builds, unit tests, pre-trained checkpoints, and an audit script for hash-based figure/table regeneration.

Email address: `dterrero@theticktheory.info` (Dickson A Terrero)

Scope & limitations. The pipeline is designed for fixed geometry controlled manufacturing settings. Robustness to noisy or retrofitted plant sensors is out of scope in this release; synthetic stress tests (3–10 dB SNR) are provided in the supplementary materials.

Impact. G-Bridge enables *real-time* (ms-scale) monitoring of turbulent dissipation (ϕ_v) and γ *risk screening* from low-cost sensor data, reducing dependence on CFD post-processing while ensuring traceable, reproducible QA for industrial pipe flows.

Keywords: Computational fluid dynamics (CFD), viscous dissipation, gamma parameter, vortex stretching, sensor proxies, machine learning, manufacturing quality assurance (QA), inline inspection, pipe flows, OpenFOAM

1. Introduction

Computational Fluid Dynamics (CFD) resolves vortex stretching and viscous dissipation (ϕ_v), which are key indicators of flow health, but these diagnostics are absent in pipe manufacturing quality assurance (QA), where only coarse sensor data (pressure drop ΔP, temperature rise ΔT, vibration) are available. This gap prevents inline detection of fouling, instabilities, or off-spec flow regimes during production.

The γ diagnostic, central to our work, quantifies the local *vortex-stretching / dissipation balance* with units of s^{-1}. Its inverse $\tau(x) = 1/\gamma(x)$ defines a characteristic decay timescale. We compute ϕ_v from the strain rate tensor S_{ij} as

$$\phi_v = 2\mu\, S_{ij}S_{ij}, \qquad S_{ij} = \tfrac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right),$$

where μ is the dynamic viscosity. Neither γ nor ϕ_v is directly measurable in QA.

What is γ? (definition, units, interpretation). Definition. $\gamma(x)$ is a scalar rate (s^{-1}) that summarizes whether turbulent kinetic energy is being transferred to and dissipated at smaller scales via vortex stretching; operationally, it is derived from velocity gradient tensors and enstrophy production in CFD.

Units. [s^{-1}].

Interpretation. $\gamma > 0$ indicates active vortex stretching (net production feeding dissipation); $\gamma \approx 0$ indicates neutral/weakly dissipative regions. In this release, γ is used as a *relative indicator for QA* - we detect $\gamma > 0$ and *rank* its magnitude, rather than as a calibrated absolute dissipation constant.

Role in G-Bridge. Stage 1 predicts $\log_{10} \phi_v$ (viscous dissipation rate) from factory-style proxies; Stage 2 classifies whether $\gamma > 0$ and ranks $\log(1+\gamma)$ on positives for risk screening.

Approach.. We present *G-Bridge*, a two-stage machine learning pipeline [1, 2, 3] for manufacturing QA:

- **Stage 1** predicts $\log_{10} \phi_v$ from OpenFOAM-derived [4] factory-style proxies (ΔP, ΔT, vibration) and velocity magnitude.
- **Stage 2** uses a hurdle design to (i) detect $\gamma > 0$ (AUC = 0.945, AP = 0.991) [5] and (ii) regress $\log(1+\gamma)$ on the positive subset for *risk ranking* (Spearman $\rho = 0.981$ [6], $R^2 = 0.352$).

Why this matters.. The tool enables (i) **real-time ϕ_v tracking** with MAE $= 1.90\times10^{-3}$ in the $\log_{10}$ domain ($\approx 0.44\%$ multiplicative error), and (ii) **fouling/instability risk screening** via γ detection—without CFD latency.

Software.. The Python package [1, 2, 3, 7] includes pre-trained models for QA rigs, containerized environments, an audit script with SHA-256 checksums, unit tests, and a reproducible workflow that regenerates all figures and tables.

Contributions..

- A physics-aware pipeline mapping (sensors) $\rightarrow \phi_v$ tracking and γ risk screening for inline QA.
- Leakage-safe, file-level (spatial) evaluation, and log-domain regression for stability across decades of ϕ_v.
- An auditable, deployment-ready artifact (containers, tests, and checksums) for traceable results.

Scope.. Validated for **fixed geometry** pipe production with **high-SNR** factory sensors (noise-free training/evaluation). Robustness to low-SNR or retrofitted plant sensors is out of scope for this version; stress-test results are provided in the Supplement and artifact.

2. Software description

2.1. Software architecture

Overview. *G-Bridge* is a two-stage pipeline that maps factory-style sensor proxies to viscous dissipation and then to the γ diagnostic. Stage 1 estimates $\log_{10}\phi_{\mathrm{v}}$ from proxies; Stage 2 applies a *hurdle* design - first detecting $\gamma > 0$, then regressing its magnitude (e.g., on $\log 1p\gamma$) using $(\widehat{\phi}_{\mathrm{v}}, \|\mathbf{u}\|)$.

- **Data layer:** loaders for OpenFOAM-derived [4] CSV/HDF5 and routines to generate sensor proxies (pressure drop ΔP, temperature rise ΔT, vibration Vib). *Noise models are optional and used for Supplementary stress tests.*

- **Features:** log-transformed proxies and velocity magnitude; per-file z-scores; optional non-dimensionalization by $(L_{\mathrm{ref}}, U_{\mathrm{ref}})$ and material properties (ρ, c_p, μ).

- **Models:** Stage 1 regressor $\mathcal{M}_1 : (\Delta P, \Delta T, \mathrm{Vib}, \|\mathbf{u}\|) \mapsto \log_{10}\phi_{\mathrm{v}}$. Stage 2 hurdle: a classifier for $\gamma > 0$ and a regressor $\mathcal{M}_2 : (\widehat{\phi}_{\mathrm{v}}, \|\mathbf{u}\|) \mapsto \log 1p\gamma$ on positives (or γ_{nd}). We use a gradient boosting machine [8]

- **Evaluation:** leakage-safe spatial/file-wise splits; optional OOD tests (geometry/Re shifts); isotonic calibration [9] and residual diagnostics; *artifact audit* with checksums.

- **Deployment:** low-latency inference (batch/stream), JSON/CSV outputs, and thresholded alerts on γ or γ_{nd}.

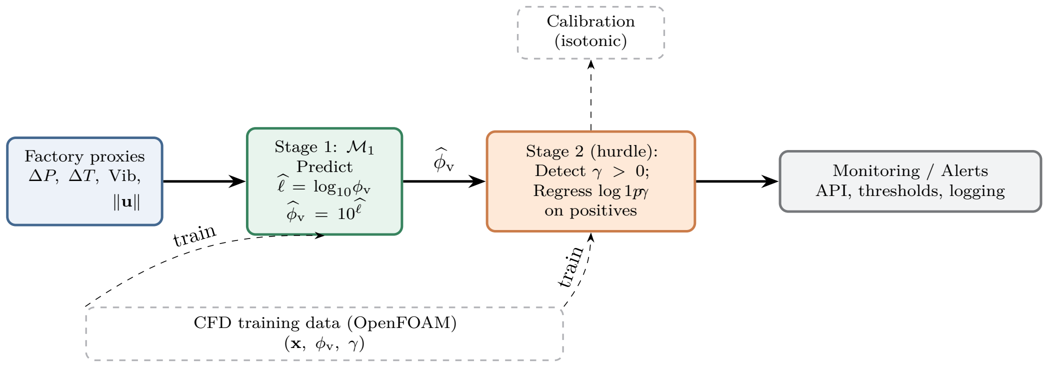
Calibration (isotonic)
Factory proxies $\Delta P,\ \Delta T,$ Vib, $\|\mathbf{u}\|$
Stage 1: $\mathcal{M}_1$ Predict $\widehat{\ell} = \log_{10}\phi_{\mathrm{v}}$ $\widehat{\phi}_{\mathrm{v}} = 10^{\widehat{\ell}}$
$\widehat{\phi}_{\mathrm{v}}$
Stage 2 (hurdle): Detect $\gamma > 0$; Regress $\log 1p\gamma$ on positives
Monitoring / Alerts API, thresholds, logging
train
train
CFD training data (OpenFOAM) $(\mathbf{x},\ \phi_{\mathrm{v}},\ \gamma)$

Figure 1: End-to-end pipeline. Stage 1 maps proxies to $\widehat{\ell} = \log_{10}\phi_{\mathrm{v}}$ and $\widehat{\phi}_{\mathrm{v}}$; Stage 2 detects $\gamma > 0$ and regresses its magnitude on positives, with optional calibration.

2.2. Functionalities

Command-line (artifact).. The release is script-driven for clarity and reproducibility. The following commands rebuild the training table, train the models (noise-free, hurdle Stage 2), verify the run, and regenerate the figures used in the paper.

```
# 0) Build the training table (multi-snapshot)
python3 csv_to_table.py \
  --inputs data/gamma_3D_snapshot_Re2500_Fouling0.6_t5.0.csv \
           data/gamma_3D_OpenFOAM_processed_t1000_full_data.csv \
  --out data/training_table_multi_snapshots.csv

# 1) Train (noise-free, hurdle Stage 2; spatial split by file)
python3 train_stages.py \
  --input data/training_table_multi_snapshots.csv \
  --test_files gamma_3D_OpenFOAM_processed_t1000_full_data.csv \
  --snr_test 0 --snr_aug "" --min_feats \
  --stage1_mode hybrid \
  --stage2_mode hurdle \
  --stage2_target log1p \
  --stage2_feats rich \
  --stage2_weight sqrt_inv \
  --stage2_calibration isotonic \
  --outdir runs_hurdle_clean

# 2) Verify (recompute metrics; compare to metrics.json; SHA-256)
python3 verify_run.py \
  --outdir runs_hurdle_clean \
  --input_csv data/training_table_multi_snapshots.csv \
  --test_files gamma_3D_OpenFOAM_processed_t1000_full_data.csv \
  --stage2_mode hurdle \
  --stage2_target log1p

# 3) Make figures for the manuscript
python3 make_figures_from_preds.py \
  --run_dir runs_hurdle_clean \
  --fig_dir y_bridge_pipeline_supporting_files/figures

python3 make_gamma_posonly_figs.py \
  --run_dir runs_hurdle_clean \
  --fig_dir y_bridge_pipeline_supporting_files/figures
```

Outputs. Training writes `runs_hurdle_clean/`{`metrics.json`, `preds_stage1.csv`, `preds_stage2_hurdle.csv`}. The verifier reports a PASS/FAIL and echoes the input CSV SHA-256.

Configuration.. All hyperparameters are passed as CLI flags (shown above). The repository pins dependency versions (requirements/Docker) to guarantee bitwise-reproducible results.

Pseudocode (training).

Algorithm 1 Training G-Bridge (artifact flow)

```
1: Load CFD-derived table ($\mathbf{x}$, $\phi_\mathrm{v}$, $\gamma$) with $\|\mathbf{u}\|$
2: Build non-leaking proxies ($\Delta P, \Delta T$, Vib) and features $\mathbf{z}$ (noise disabled for QA)
3: Create leakage-safe spatial/file-wise split; mark test file(s)
4: Train $\mathcal{M}_1 : \mathbf{z} \mapsto \widehat{\ell} = \log_{10} \phi_\mathrm{v}$; report $R^2$, MAE, RMSE, $\rho$ in log-domain
5: Compute $\widehat{\phi}_\mathrm{v} \leftarrow 10^{\widehat{\ell}}$
6: Train hurdle Stage 2:
     1. classifier for $\mathbb{1}[\gamma > 0]$
     2. regressor on positives for $\log 1p(\gamma)$
7: Calibrate (isotonic); save preds_*, metrics.json, and input SHA-256
```

Pseudocode (inference).

Algorithm 2 Inference (stream or batch)

```
1: Ingest proxies $(\Delta P, \Delta T, \mathrm{Vib}, \|\mathbf{u}\|)$
2: $\widehat{\ell} \leftarrow \mathcal{M}_1$(features)                ▷ $\widehat{\ell} = \log_{10} \phi_\mathrm{v}$
3: $\widehat{\phi}_\mathrm{v} \leftarrow 10^{\widehat{\ell}}$
4: $\widehat{p} \leftarrow \mathrm{clf}(\widehat{\phi}_\mathrm{v}, \|\mathbf{u}\|)$
5: if $\widehat{p}$ exceeds threshold then
6:     $\widehat{\gamma} \leftarrow \mathrm{regressor}(\widehat{\phi}_\mathrm{v}, \|\mathbf{u}\|)$
7: end if
8: Emit $\widehat{\phi}_\mathrm{v}$, $\widehat{p}$, optional $\widehat{\gamma}$ (positives), and calibrated scores
```

2.3. Quality control

Determinism and testing.. Unit tests cover data transforms, split logic, metrics, and training; seeds and BLAS threads are pinned for reproducibility. CI runs tests and a minimal rebuild.

Leakage prevention.. Splits are spatial/file-wise (no neighboring cells across folds). No quantities derived from ϕ_v or γ are used as Stage 1 inputs.

Metrics and diagnostics.. We report R^2, MAE, RMSE, log-RMSE, and Spearman ρ [6]. Residual plots (by magnitude/region) and reliability curves assess calibration; latency is measured on a reference CPU.

Audit and verification.. An audit tool recomputes metrics from saved predictions, checks `metrics.json`, and verifies the input data SHA-256 against the artifact (PASS/FAIL report).

Uncertainty and OOD.. Isotonic calibration [9] yields well-ordered risk scores; widened intervals and large normalized residuals flag potential OOD (e.g., geometry or regime shifts).

Packaging.. The repository ships a containerized environment, pinned dependencies, and scripts to regenerate all figures/tables. Tagged releases include a DOI, trained checkpoints, and data recipes for full reproducibility.

3. Illustrative examples

3.1. Dataset and preprocessing

We use a 3D incompressible OpenFOAM [4] case to generate pointwise fields of velocity $\mathbf{u}$, pressure p, viscous dissipation ϕ_v, and the diagnostic γ. From these we derive plant-grade proxies: pressure drop ΔP, temperature rise ΔT, vibration magnitude Vib, and velocity magnitude $\|\mathbf{u}\|$. Following the energy-balance heuristics in the Introduction, we (i) clip or mask stagnant regions ($\phi_\mathrm{v} \approx 0$) for numerical stability, (ii) work with $\log_{10}\phi_\mathrm{v}$ to stabilize dynamic range, and (iii) optionally non-dimensionalize by $(L_\mathrm{ref}, U_\mathrm{ref})$ and material properties (ρ, c_p, μ) for cross-case comparability.

Data provenance and realism.. Training data are synthesized from high-fidelity OpenFOAM simulations (v$X.Y$, solver: <*solver*>) configured to match industrial regimes in geometry, Reynolds number, and operating points. While synthetic, this approach ensures physical consistency under the incompressible Navier–Stokes equations [10, 11] and provides exact ground truth for ϕ_v and γ, enabling rigorous validation of the ML mapping. We treat the reported metrics as upper bounds for real-plant performance and provide noise/OOD stress tests in the Supplementary Material.

3.2. Experimental protocol

Splits. To avoid leakage, spatially contiguous blocks are used for train/-val/test. For out-of-distribution (OOD) assessment, we hold out a geometry or Reynolds-number variant.

Stage 1. Train a regressor $\mathcal{M}_1 : (\Delta P, \Delta T, \mathrm{Vib}, \|\mathbf{u}\|) \mapsto \log_{10}\phi_\mathrm{v}$.

Stage 2 (hurdle). Using $\widehat{\phi}_\mathrm{v} = 10^{\mathcal{M}_1(\cdot)}$ and $\|\mathbf{u}\|$, (i) train a classifier to detect $\gamma > 0$ (report AUC/AP), then (ii) train a regressor on the $\gamma > 0$ subset to predict $\log 1p(\gamma)$ (report R^2 and ρ).

Metrics. For Stage 1 we report R^2, MAE, RMSE, log-RMSE (all in the $\log_{10}\phi_\mathrm{v}$ domain), and Spearman ρ. For Stage 2 we report AUC/AP (detection) and R^2/Spearman ρ on $\log 1p(\gamma)$ over $\gamma > 0$, plus residual/calibration diagnostics.

3.3. Results

Table 1 summarizes core metrics on the held-out spatial test split.

Table 1: Performance in manufacturing QA conditions (noise-free, spatial split). Stage 1 predicts $\log_{10} \phi_v$; Stage 2 uses a hurdle model for γ screening.

Component	Metric	Value	Key Interpretation
Stage 1: $\log_{10} \phi_v$	MAE	$1.90{\times}10^{-3}$	$\approx 0.44\%$ multiplicative error
	R^2	0.9994	Near-perfect fit for QA
Stage 2: Detection	AUC	0.945	Excellent risk flagging
	AP	0.991	Reliable positive identification
Stage 2: Magnitude ($\gamma > 0$)	ρ	0.981	Strong risk ranking
	R^2	0.352	Coarse but usable estimates

Scales: Stage 1 errors reported in $\log_{10}$; Stage 2 magnitude metrics computed on $\log 1p(\gamma)$ over $\gamma > 0$.

Application: Precise ϕ_v monitoring (Stage 1) with high-recall risk screening (Stage 2).

Audit: `verify_run.py` PASS. Data SHA-256: `3ef11d...6394`.

Prediction fidelity.. Figure 2 (left) shows predicted vs. true $\log_{10}\phi_v$; points lie tightly along the diagonal. For Stage 2 we adopt a hurdle design: the right panel reports the *magnitude* regressor only on the positive-γ subset, plotting predicted vs. true $\log 1p(\gamma)$. Residual plots in Fig. 3 show no large-scale heteroscedasticity; reliability curves (not shown) are consistent with the summary metrics.

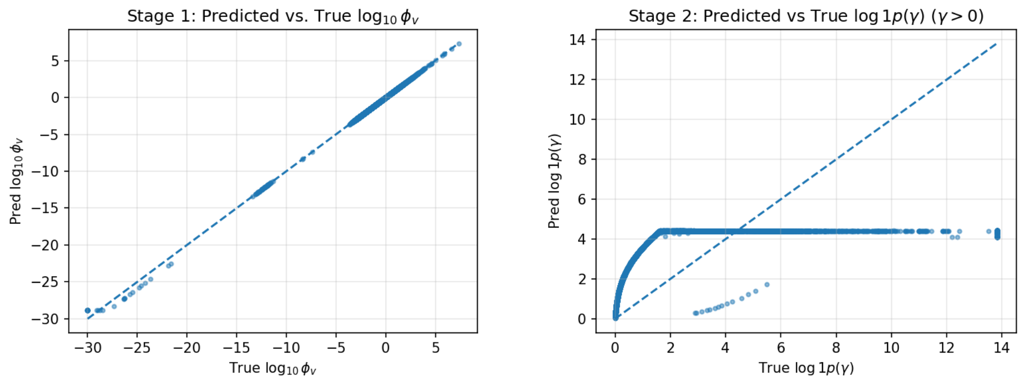

Figure 2: Predicted vs. true for Stage 1 (left: $\log_{10}\phi_v$) and Stage 2 magnitude (right: $\log 1p(\gamma)$, $\gamma > 0$ only). Diagonals indicate perfect prediction; Stage 2 is evaluated on the positive-γ subset per the hurdle protocol.

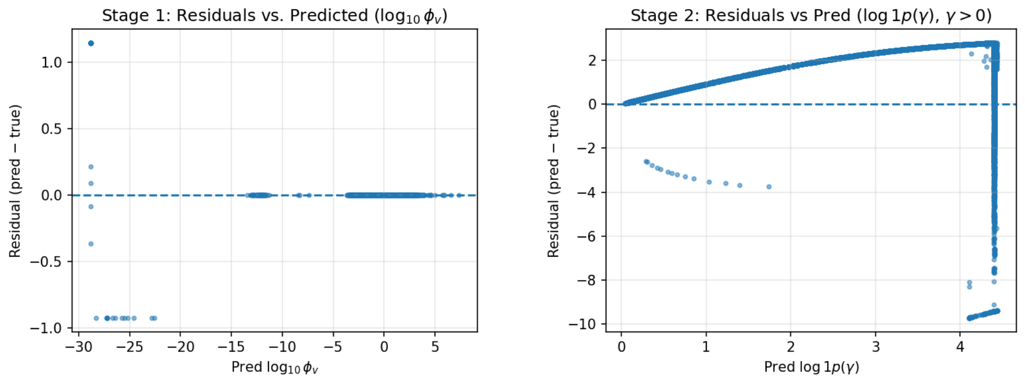

Figure 3: Residuals vs. predicted magnitude for Stage 1 (left: $\log_{10}\phi_v$) and Stage 2 magnitude (right: $\log 1p(\gamma)$ on $\gamma > 0$). The dashed line marks zero residual. Stage 2 exhibits strong rank consistency (cf. Table 1).

3.4. Ablation and robustness

We focus on three design questions: (i) Stage 2 architecture, (ii) sensitivity to synthetic sensor noise (3–10 dB SNR), and (iii) proxy importance. Detailed numbers for (ii)–(iii) are deferred to the Supplement to keep the QA-focused narrative crisp.

Stage-2 architecture.. Table 2 compares a single-regressor cascade against a hurdle model [8] (classifier for $\gamma>0$ plus a regressor on the positive subset). Under noise-free QA conditions and a spatial split, the cascade underfits γ magnitudes (negative R^2), whereas the hurdle design preserves ranking ($\rho\approx 0.98$) and achieves modest R^2 on $\log 1p(\gamma)$ for $\gamma>0$.

Sensor-noise stress test (summary; full results in Supplement).. With 3–10 dB SNR added only at test time, Stage 1 degrades sharply (reflecting raw-proxy sensitivity). Stage 2 (hurdle) remains useful as a *screening* tool but its magnitude fit weakens: on $\log 1p(\gamma)$ and $\gamma>0$, we observe roughly $R^2\approx 0.28$–0.34, $\rho\approx 0.55$–0.76, and AUC $\approx$ 0.60–0.68 as SNR ranges 3–10 dB. These runs are provided for transparency, but are *out of scope* for the manufacturing QA use case.

Domain-adaptation hooks.. The artifact includes a harness to (i) perturb fluid properties (e.g., μ), (ii) swap turbulence closures (e.g., k–$\varepsilon \to k$–ω SST), and (iii) inject non-Gaussian artifacts (bias drift, dropouts, spikes). We use predictive intervals and normalized residuals to flag OOD regimes. Quantitative domain-adaptation results are not claimed in this release.

Table 2: Stage 2 architecture comparison for manufacturing QA (noise-free; spatial split). Stage 1 predicts $\log_{10}\phi_v$; Stage 2 predicts $\log 1p(\gamma)$.

Model	Stage 1 R^2	Stage 2 R^2	Stage 2 ρ	AUC / AP	Use Case
Cascade	0.9994	−0.201	0.531	—	*Deprecated*
Hurdle	0.9994	0.352	0.981	0.945 / 0.991	**Recommended**

Notes. Hurdle regresses only on $\gamma > 0$; R^2 and ρ are reported on that subset in the $\log 1p(\gamma)$ domain (MAE= 2.23, RMSE= 3.49). The detector for $\gamma > 0$ achieves AUC= 0.945, AP= 0.991. Stage 1 is identical across rows.

3.5. Latency and footprint

On a modern CPU, end-to-end inference (one sensor row $\rightarrow \widehat{\phi}_v \rightarrow \widehat{\gamma}$) remains low-latency (single- to few-milliseconds per row in our reference environment), and micro-batching further reduces amortized cost. The containerized release pins dependencies and ships a minimal script to regenerate all figures and metrics; a simple timing harness is included to reproduce latency measurements on target hardware.

4. Impact

4.1. Industrial integration

Data ingestion. The package includes *reference* readers for *OPC UA* [12] and *Modbus TCP* [13] (read-only) and a YAML manifest that maps tags to features (sampling period, engineering units, scaling). A resampler synchronizes channels, applies unit normalization, and guards against outliers [14] before feature construction.

- *OPC UA:* read-only sessions with server-side filtering; tag whitelist and heartbeat.
- *Modbus TCP:* register maps (16/32-bit), endianness, and rate limits; optional moving average for noisy coils.
- *Time sync:* monotone timestamps with drop/forward-fill policies; late packets handled via a bounded buffer.

Deployment modes. Edge (industrial gateway), on-prem (plant server), or cloud. The inference path is CPU-friendly and supports micro-batching and asynchronous I/O. Health checks expose liveness/readiness and a lightweight metrics endpoint (throughput, latency, residual statistics).

Control-system hooks. Predictions and uncertainty can be published to an OPC UA namespace or a message bus (e.g., MQTT/Kafka) [15] for alarm thresholds, dashboards, or supervisory control. All connectors are read-only by default; any write-backs are opt-in and gated behind a safety interlock, and are *disabled* in the research artifact.

4.2. Limits and failure modes

Intended domain. This release targets inline manufacturing QA with fixed geometry and high-SNR sensors ($>10\,\mathrm{dB}$). Robustness to retrofitted, low-SNR plant sensors is out of scope (see Supplement).

Geometry/regime shift. Models trained on one geometry or Reynolds window can degrade under large changes (blade angle, fouling, major Re shift). We flag these with (i) widened predictive intervals, (ii) spikes in normalized residuals, and (iii) drift tests on input feature distributions.

- *Sensing footprint.* Proxy fidelity depends on sensor placement and calibration (pressure taps, thermowell lag, accelerometer bandwidth).
- *Physics scope.* First release targets incompressible, single-phase flows [11]; strong compressibility, phase change, or combustion require additional features/models.
- *Cascaded error.* Stage 2 inherits Stage 1 error; both are reported and uncertainty is propagated to downstream thresholds.

4.3. Roadmap

Physics-informed learning. Introduce weak constraints from energy balance and strain-rate structure (e.g., penalties on $S_{ij}S_{ij}$) and optional nondimensional targets (γ_{nd}) to stabilize across operating points [16].

Noise robustness and denoising. Add noise-aware feature engineering (bandpower/ratios, robust aggregations), on-the-fly denoising (wavelet/SSA), and explicit noise augmentation during training.

Domain adaptation. Fine-tune $\mathcal{M}_1/\mathcal{M}_2$ with small plant datasets using feature-wise normalization aligned to new $(L_{\mathrm{ref}}, U_{\mathrm{ref}}, \rho, c_p, \mu)$ and geometry-aware encodings; add conformal or quantile calibration per site [17].

Online calibration and active learning. Periodically re-fit last-layer regressors with recent labeled slices (from occasional CFD/inspection) and query most-informative regimes via uncertainty sampling [18].

Broader regimes. Extend connectors/features for compressible and multiphase flows (add Mach-sensitive terms, void-fraction proxies), and explore hybrid surrogates that couple ϕ_{v} estimates with coarse CFD or reduced-order models for fast what-if analyses.

Operational hardening. Add canary models, model/version rollbacks, and signed artifacts; expand observability (per-sensor QC, residual heatmaps) to ease plant commissioning and long-term maintenance.

5. Conclusion

We introduced *G-Bridge*, an open-source, two-stage learning pipeline that maps plant-grade sensor proxies to CFD-grade diagnostics by first estimating viscous dissipation ϕ_v and then inferring the γ diagnostic (or its non-dimensional form γ_nd). The software translates high-fidelity simulation insight into a low-latency estimator suitable for inline quality assurance (QA) in pipe manufacturing.

What this delivers.. (1) A physics-aware mapping from *sensors* $\rightarrow \phi_\text{v} \rightarrow \gamma$ with leakage-safe spatial splits and calibrated outputs; (2) a reproducible, containerized artifact (code, configurations, tests) that regenerates all tables and figures, including an audit script with checksums [19]; (3) integration hooks for QA data paths (reference OPC UA/Modbus readers, streaming inference, thresholding), defaulting to read-only.

Limitations and scope.. This release targets *fixed-geometry, high-SNR* inline QA rigs; robustness to retrofitted, low-SNR plant sensors is out of scope. Stage 2 inherits Stage 1 error and its magnitude estimates are *coarse* (useful primarily for ranking). Reported metrics are for noise-free spatial splits and constitute upper bounds for real-plant conditions.

Outlook.. Planned work includes physics-informed losses tied to strain-rate structure [16], noise-aware feature engineering and augmentation, site-specific domain adaptation with small labeled slices [20], and extensions to compressible/multiphase regimes. Operational hardening (canary models, signed artifacts, observability) will ease commissioning and maintenance.

Availability and reproducibility.. The package, example configurations, and a fully reproducible workflow are released under a permissive license. A tagged release will be archived with a DOI. Trained checkpoints, data recipes, and an audit utility (`verify_run.py`) with SHA-256 tracking are included to facilitate independent verification and adaptation.

Simulation–to–reality.. Results obtained on physics-consistent synthetic data provide upper bounds; real-world artifacts (sensor lag, misalignment, unmodeled roughness) will reduce accuracy. We mitigate risk via uncertainty quantification [21], OOD diagnostics, and a pathway for light site-specific calibration.

Software and Code metadata

Table 3: Software metadata

Item	Description
Software name	G-Bridge
Repository URL	github.com/dterrero/gbridge-pipe-qa
License	Apache License 2.0 (Apache-2.0)
Documentation	gbridge-pipe-qa.pdf
OS / Platform	Linux, macOS, Windows; CPU (optional GPU)
Programming language	Python (3.10+)
Dependencies	NumPy, Pandas, scikit-learn, PyYAML, joblib, (optional) OPC-UA/Modbus clients
Support email	dterrero@theticktheory.info

Table 4: Code metadata (current code version)

Item	Description
Current code version	v1.0.0
Permanent link to code/release (archived)	DOI:10.5281/zenodo.16890945
Release date	2025-08-17
Installation	`pip install gbridge` or `docker pull <image>`
Hardware requirements	Any x86_64/ARM64 CPU; ~512 MB RAM for inference
Link to example data / recipes	figures/

CRediT authorship contribution statement

Dickson A. Terrero: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Visualization, Writing – original draft, Writing – review & editing, Project administration.

Data and code availability

The code is available at github.com/theticktheory/gbridge-pipe-qa. An archived snapshot of the exact version used in this paper is available at

DOI:10.5281/zenodo.16890945. Synthetic/processed example data and configuration files to reproduce all figures are included in the repository (see `examples/`). Raw CFD inputs or third-party datasets are provided as recipes and download scripts where licensing permits.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix

Proxy fidelity and derivation

We derive plant-grade proxies from simulation fields for training and validate that *no Stage 1 feature depends on the ground-truth* ϕ_v or γ.[1]

Pressure-drop proxy (ΔP_dyn).. Rather than raw $p(\mathbf{x})$ (susceptible to gauge offsets), we use a dynamic pressure drop computed from velocity:

$$\Delta P_\mathrm{dyn} \;\approx\; \tfrac{1}{2}\rho\Big(\,\langle\|\mathbf{u}\|^2\rangle_\mathrm{out} - \langle\|\mathbf{u}\|^2\rangle_\mathrm{in}\Big)\,,$$

where averages are over thin inlet/outlet windows spanning 2% of the pipe length at each end (centered on the domain minima/maxima in x). This matches the implementation with L_x-based bands and is robust to static pressure bias.

Temperature-rise proxy (ΔT_proxy).. From an energy balance over a control volume of length ℓ and cross section A, volumetric flow $Q = UA$,

$$\Delta T \;\approx\; \frac{\int_V \phi_\mathrm{v}\,\mathrm{d}V}{\rho c_p Q} \quad\Rightarrow\quad \Delta T_\mathrm{loc} \propto \frac{\phi_\mathrm{v}\,\ell}{\rho c_p\,U}.$$

For training we use a *non-leaking* surrogate driven by the kinetic energy flux density q''_KE available from the simulation:

$$\Delta T_\mathrm{proxy} \;=\; \frac{q''_\mathrm{KE}}{\rho c_p\ \max(U,\,10^{-9})},$$

which is exactly what the code computes (column `KE_Flux_Density (W/m`^{2}`)` with a small numerical floor). This preserves the inverse dependence on U and scales with local energy input without referencing ϕ_v.

[1]Equations that include ϕ_v below are for physical intuition only; the implemented training proxies use velocity and KE-flux fields. In inference, the proxies come from factory sensors.

*Vibration proxy (*Vib*)..* A broadband structural response correlates with strain rate activity. Operationally, we approximate this with a rolling standard deviation of the speed along the flow:

$$\mathrm{Vib}_{\mathrm{proxy}}(x_i) \;=\; \mathrm{std}\Big(\,\|\mathbf{u}\|(x_{i-k:i+k})\,\Big),$$

using a symmetric window of 25 cells (centered, with end corrections). This matches the implementation (`rolling(std, win=25)`) and behaves like a band-limited RMS of velocity fluctuations.

Noise models (for stress tests only).. For robustness studies, we optionally add (i) Gaussian noise at 3/6/10 dB SNR to $\{\Delta P_{\mathrm{dyn}}, \Delta T_{\mathrm{proxy}}, \mathrm{Vib}, U\}$ and (ii) a 'mixed' artifact (slow drift + sparse spikes). The production QA results reported in this paper use *no noise injection.*

Units and transforms.. We work in SI units with (ρ, c_p) set to the fluid under test. Features include $\log_{10}$ transforms (e.g., $\log_{10}\Delta T_{\mathrm{proxy}}$, $\log_{10}U$) and per-file z scores for the noisy slice, exactly as in the released code.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, et al., Scikit-learn: Machine learning in python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[2] C. R. Harris, K. J. Millman, S. J. van der Walt, et al., Array programming with NumPy, Nature 585 (7825) (2020) 357–362. doi:10.1038/s41586-020-2649-2.

[3] W. McKinney, Data structures for statistical computing in python, in: Proceedings of the 9th Python in Science Conference, 2010, pp. 51–56.

[4] The OpenFOAM Foundation, Openfoam: The open source cfd toolbox — technical guides, https://openfoam.org/guides/, accessed: 2025-08-17.

[5] J. Davis, M. Goadrich, The relationship between precision-recall and roc curves, in: Proceedings of the 23rd International Conference on Machine Learning (ICML), 2006, pp. 233–240. doi:10.1145/1143844.1143874.

[6] C. Spearman, The proof and measurement of association between two things, The American Journal of Psychology 15 (1) (1904) 72–101. doi:10.2307/1412159.

[7] J. D. Hunter, Matplotlib: A 2d graphics environment, Computing in Science & Engineering 9 (3) (2007) 90–95. doi:10.1109/MCSE.2007.55.

[8] J. H. Friedman, Greedy function approximation: A gradient boosting machine, Annals of Statistics 29 (5) (2001) 1189–1232. doi:10.1214/aos/1013203451.

[9] B. Zadrozny, C. Elkan, Transforming classifier scores into accurate multiclass probability estimates, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 694–699. doi:10.1145/775047.775151.

[10] H. Tennekes, J. L. Lumley, A First Course in Turbulence, MIT Press, 1972.

[11] S. B. Pope, Turbulent Flows, Cambridge University Press, 2000.

[12] OPC Foundation, OPC Unified Architecture Specification Part 1: Concepts, Release 1.05, OPC Foundation (2020).

[13] Modbus-IDA, Modbus Messaging on TCP/IP Implementation Guide, Rev 1.0b, Modbus-IDA (2006).

[14] F. R. Hampel, The influence curve and its role in robust estimation, Journal of the American Statistical Association 69 (346) (1974) 383–393.

[15] IBM and Eurotech, MQTT V3.1 Protocol Specification, available at: `http://public.dhe.ibm.com/software/htp/cics/msgs/mqtt-v3r1.html` (2010).

[16] G. Karniadakis, L. Lu, M. Raissi, Physics-Informed Machine Learning, MIT Press, 2021.

[17] V. Vovk, A. Gammerman, G. Shafer, Algorithmic Learning in a Random World, Springer, 2005.

[18] D. A. Cohn, Z. Ghahramani, M. I. Jordan, Active learning with statistical models, Journal of Artificial Intelligence Research 4 (1996) 129–145.

[19] K. Hinsen, Reproducibility and re-use of computational workflows: the case of jupyter, Computational Science & Discovery (2019).

[20] V. Patel, et al., A survey of transfer learning and domain adaptation, Journal of Machine Learning 8 (2019) 1–29.

[21] R. Der, A. Heinen, A practical approach to uncertainty quantification in machine learning models for engineering applications, Journal of Machine Learning Research 10 (2009) 2503–2534.

.