Investigating Machine Learning Models for the Development of K-Solutions

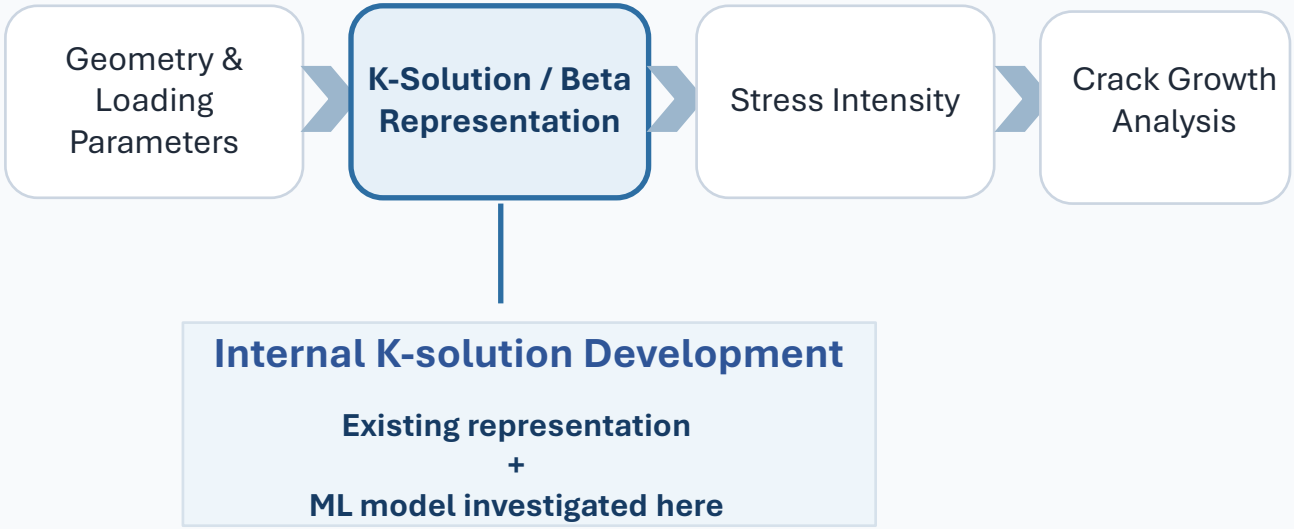
An exploratory R&D study of multidimensional Beta modeling for AFGROW

Sai Sampan Patrudu Lanka | James Lambert

LexTech, Inc. • AFGROW Workshop • September 16, 2026

Where Machine Learning Fits in the AFGROW Workflow

- This work focuses on the internal development and review of K-solution and Beta data used by AFGROW.
- In the current phase, the AFGROW user workflow remains unchanged while the ML capability is developed and evaluated behind the scenes.
- The goal is to investigate whether ML can help us develop, check, and maintain increasingly complex multidimensional K-solutions more effectively.
- If the approach continues to validate well, it could support future improvements to AFGROW, with the exact form of that capability still open.



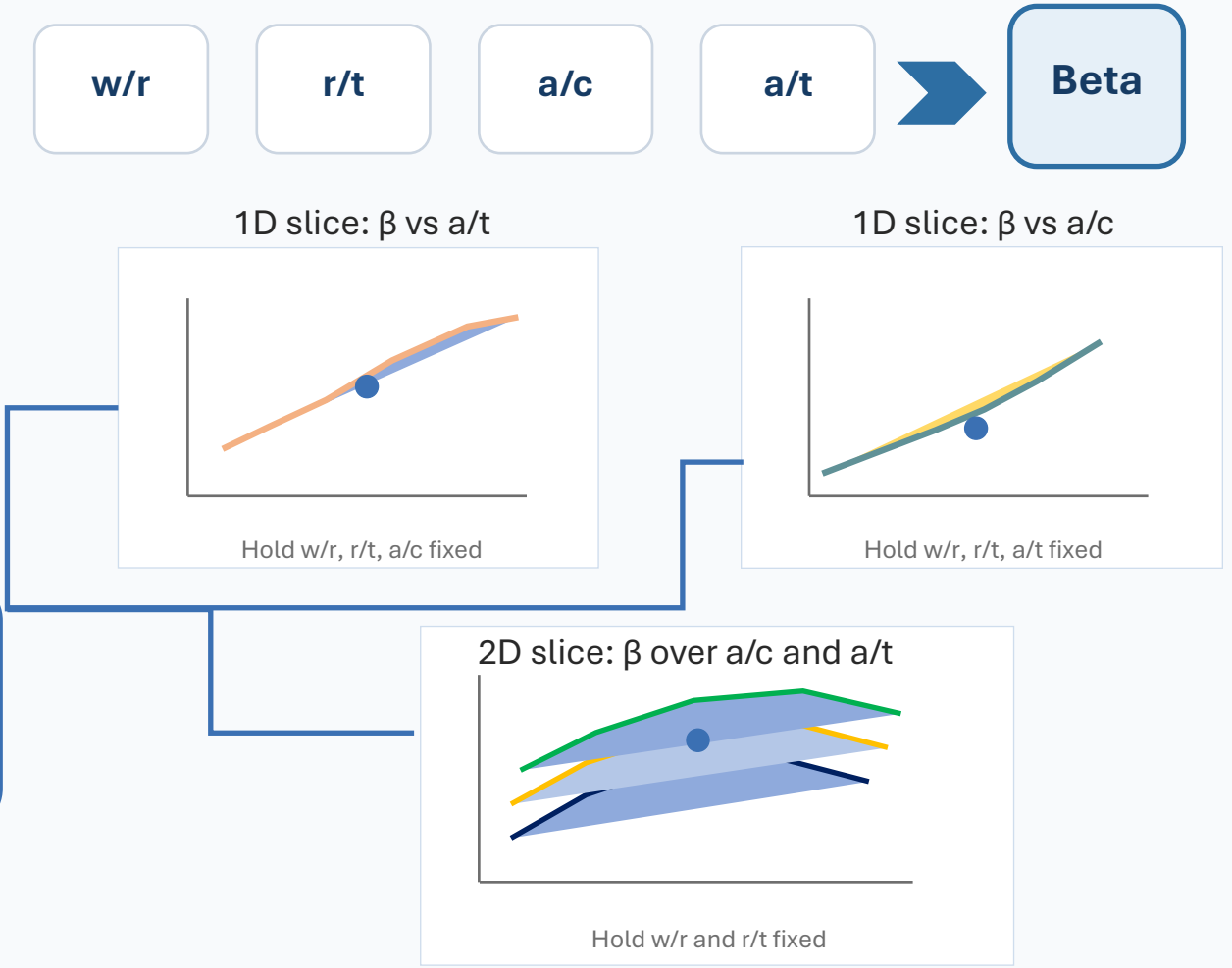
Why Explore ML for K-Solution Development?

- K-solution datasets can involve several interacting geometry and loading parameters, creating an increasingly multidimensional solution space.
- Engineers still inspect and refine these solutions through lower-dimensional curve slices, even though those slices are connected to the same overall relationship.
- A fit that looks acceptable in one direction can still produce undesirable behavior when the same data are viewed along another dimension.
- As solution spaces become larger and more complex, the internal effort required to fit, inspect, and maintain all of those connected views also grows.

Higher-dimensional Beta relationship
β = f(w/r, r/t, a/c, a/t)

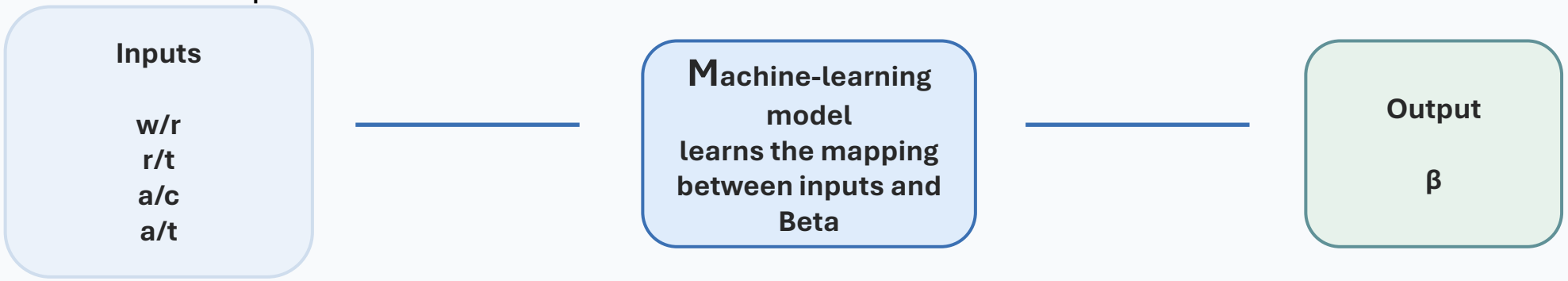
Different lower-dimensional views of the same underlying relationship.

Current finite-width example



Framing the Machine-Learning Problem

- The available K-solution data provide crack-geometry inputs together with the corresponding Beta values.
- Earlier cases used r/t, a/c, and a/t as inputs, while the finite-width case added w/r.
- Beta is treated as the prediction target for the machine-learning model.
- The model learns how Beta changes as the geometry parameters vary together.
- The relationship is nonlinear and multidimensional, so the inputs cannot be treated as completely independent.
- Separate loading and crack-location cases are modeled from their own datasets rather than assuming one universal relationship.
- The model must reproduce both accurate Beta values and useful curve behavior across the solution space.



Earlier datasets used three inputs (r/t, a/c, a/t). The finite-width case extends the problem to four physical inputs.

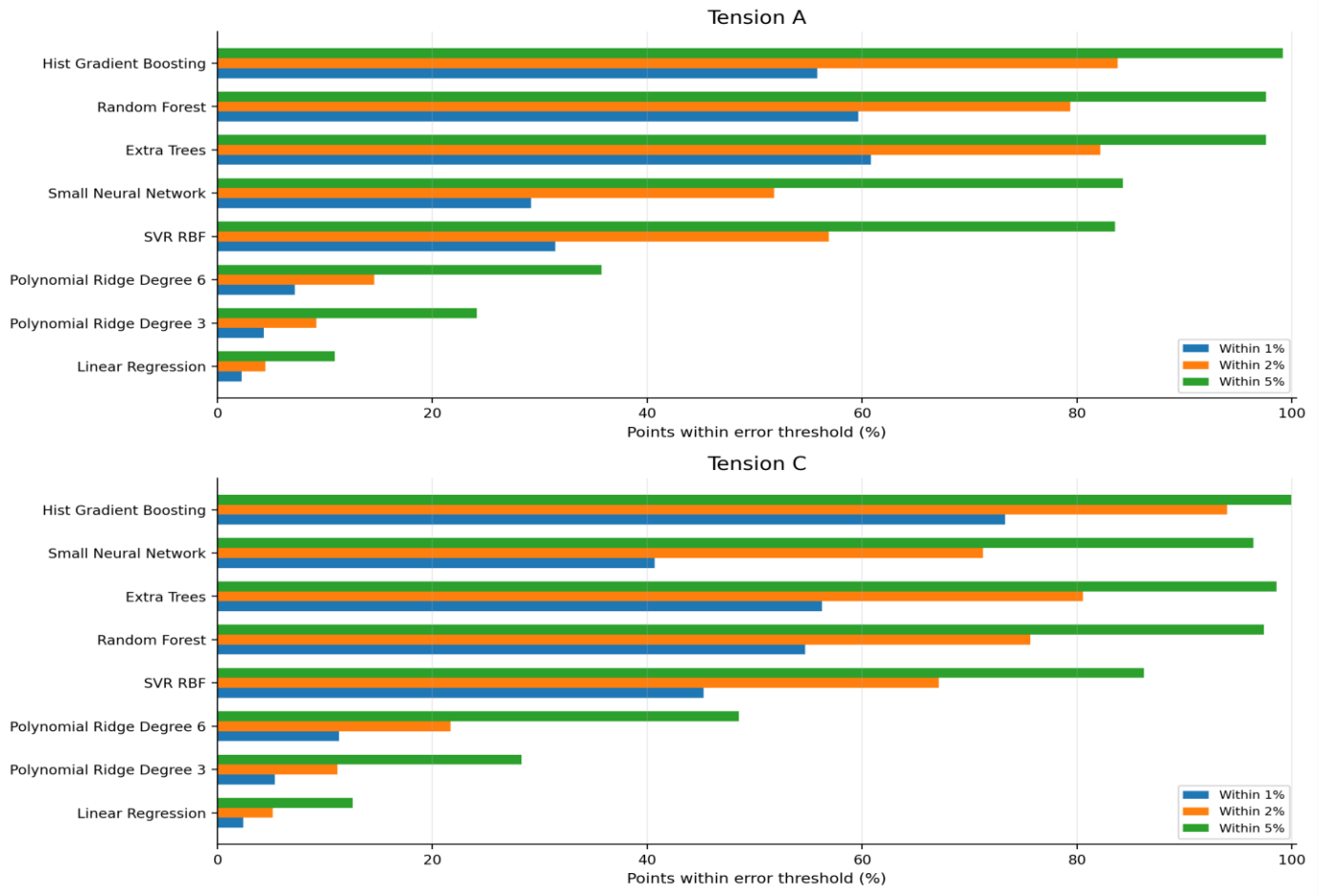
Questions That Guided the Investigation

1. Can ML reproduce the Beta relationships accurately?
2. How reliable is random-split testing for structured K-solution data?
3. How should curve behavior be validated?
4. Can ML capture the multidimensional relationship with smooth curve behavior?
5. Can the model become portable?

Initial Benchmark: ML Models Captured the Beta Relationship

- Linear regression and polynomial ridge provided the initial baselines, while the tree-based models captured the Beta relationship much more effectively in the random-split benchmark.
- Hist Gradient Boosting became the strongest early random-split baseline for both Tension A and Tension C.
- Random-split coverage within 5% reached 99.15% for Tension A and 99.95% for Tension C.
- At this stage, the obvious conclusion was that point prediction looked promising.

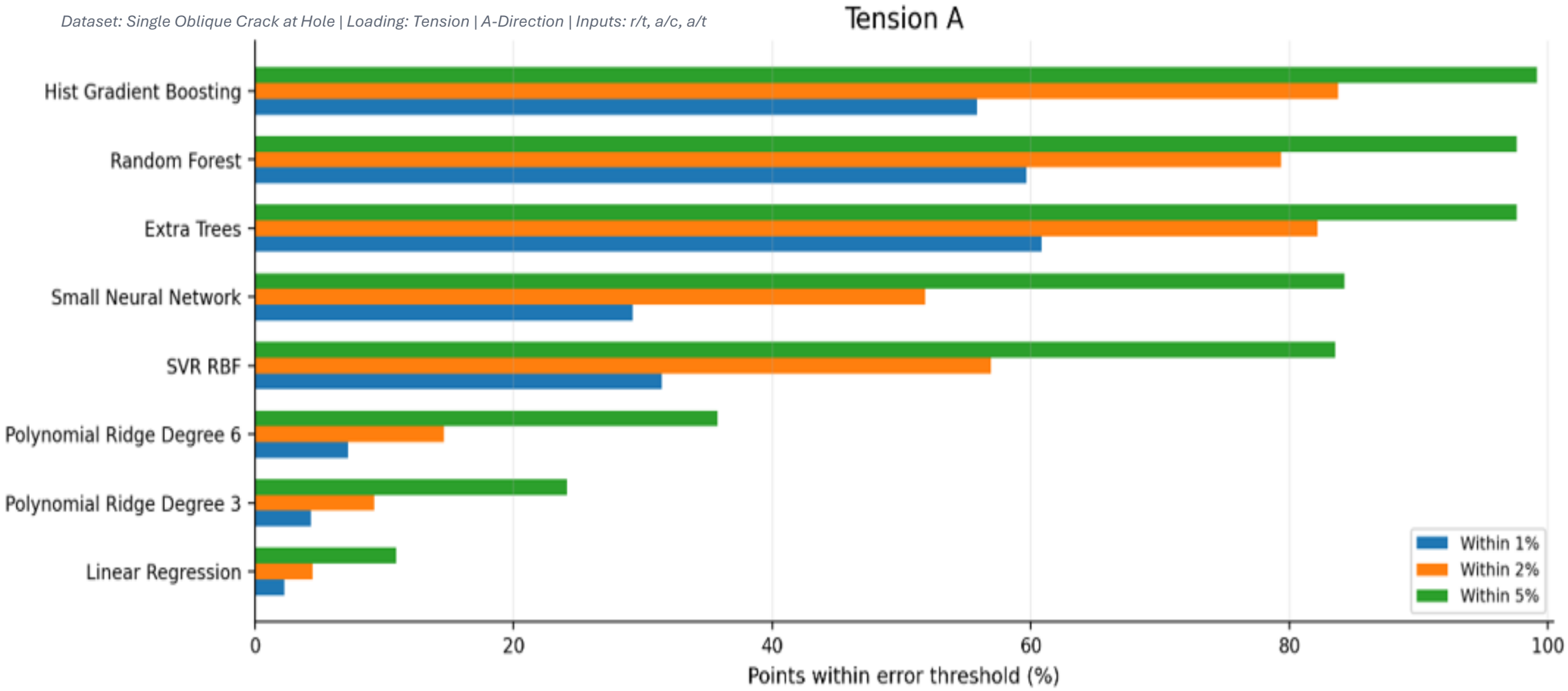
Dataset: Single Oblique Crack at Hole | Loading: Tension | A- and C-Directions | Inputs: r/t, a/c, a/t



But this was only a random-row test.

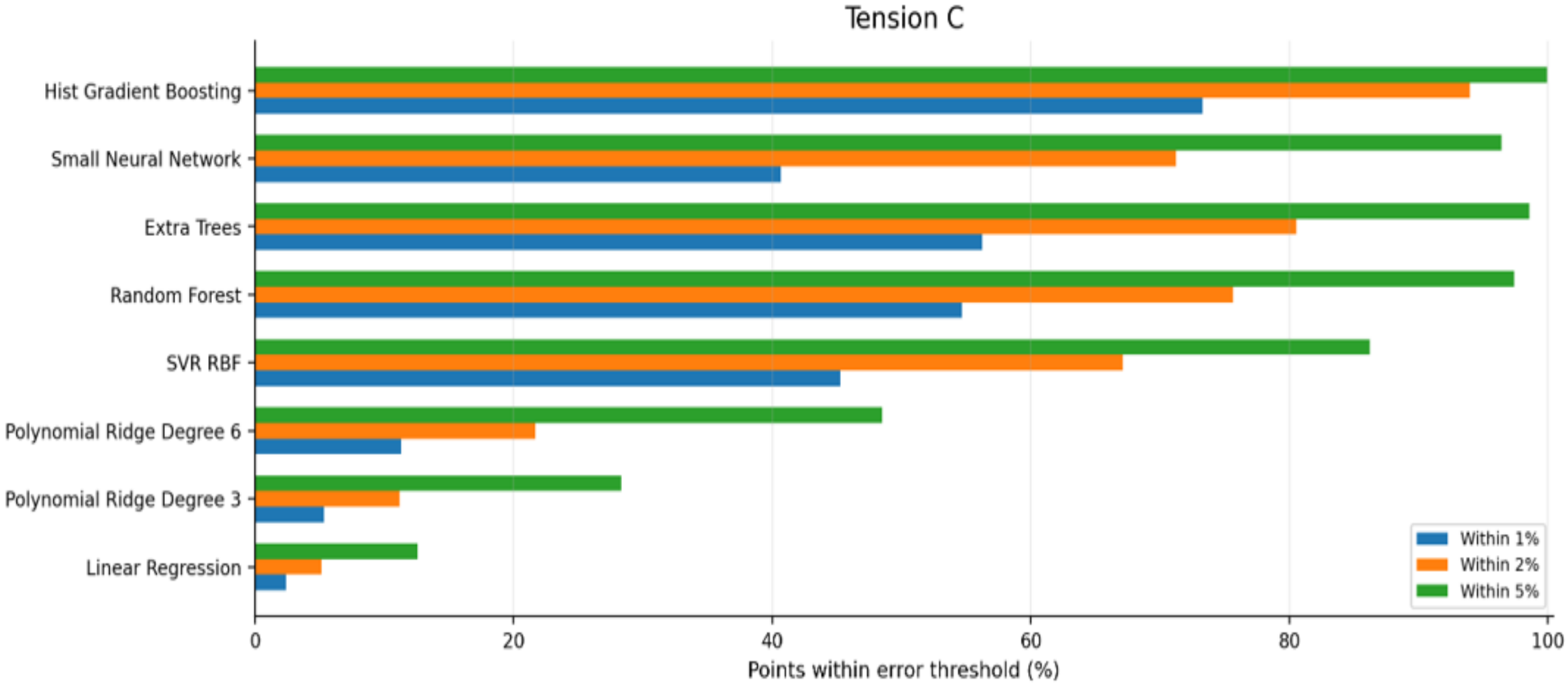
Expanded View: Initial Benchmark Across Model Families

Dataset: Single Oblique Crack at Hole | Loading: Tension | A-Direction | Inputs: r/t, a/c, a/t



Expanded View: Initial Benchmark Across Model Families

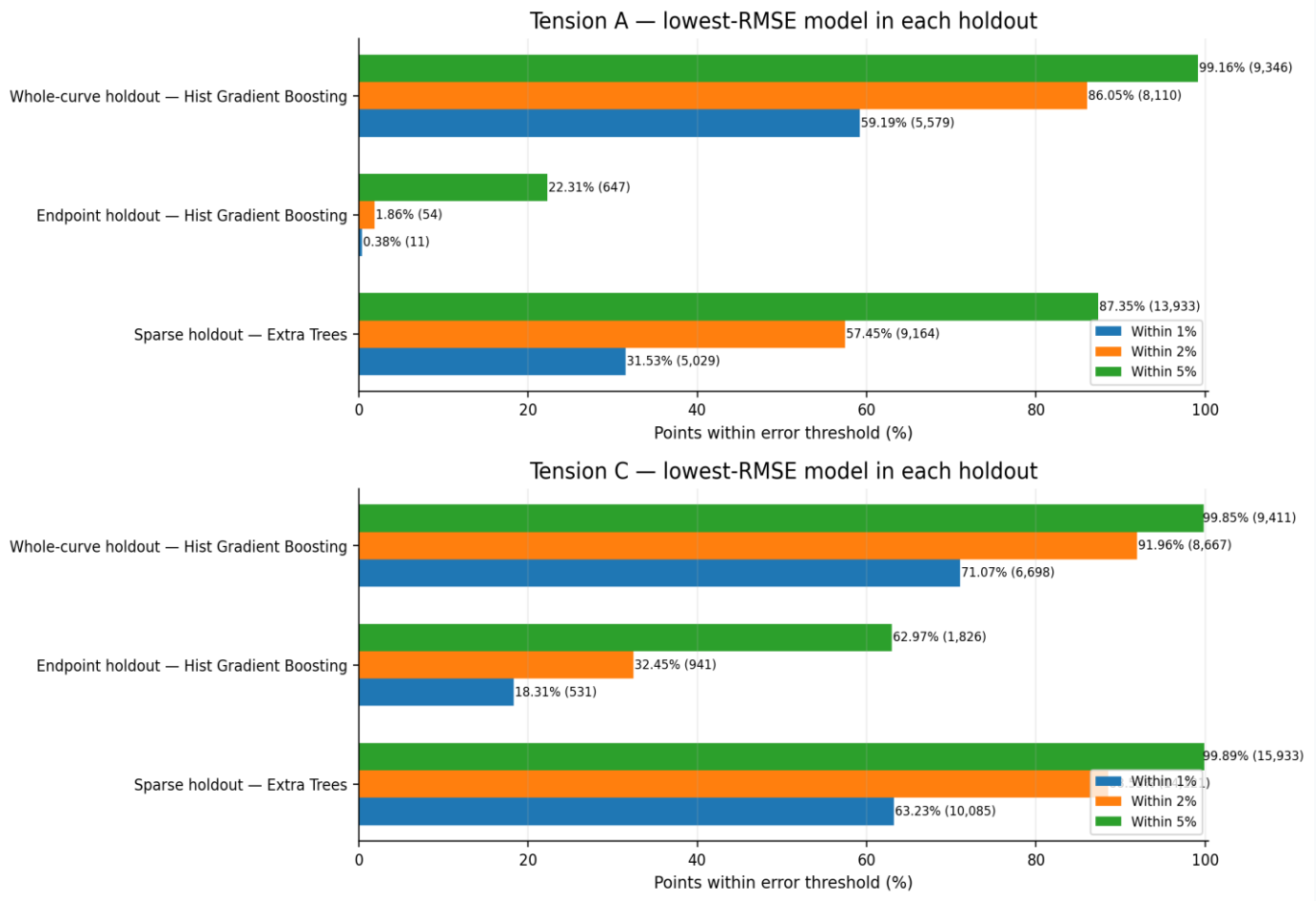
Dataset: Single Oblique Crack at Hole | Loading: Tension | C-Direction | Inputs: r/t, a/c, a/t



Random Split Accuracy Was Not Enough

- A random row split can place points from the same or very similar curves in both training and test data.
- This makes the test useful as an initial benchmark, but it can make prediction look easier than the real engineering problem.
- Structured validation was introduced to remove data in more meaningful ways, including sparse a/t sampling, complete curve holdout, and endpoint holdout.
- Performance remained strong for some interpolation-type tests but became noticeably harder when entire curves or endpoint regions were withheld.
- The endpoint tests were especially important because they showed where the model was being asked to predict farther from the available training information.
- Example: HGB within 5% fell from 99.15% to 22.31% for Tension A and from 99.95% to 62.97% for Tension C.

Dataset: Single Oblique Crack at Hole | Loading: Tension | A- and C-Directions | Inputs: r/t, a/c, a/t

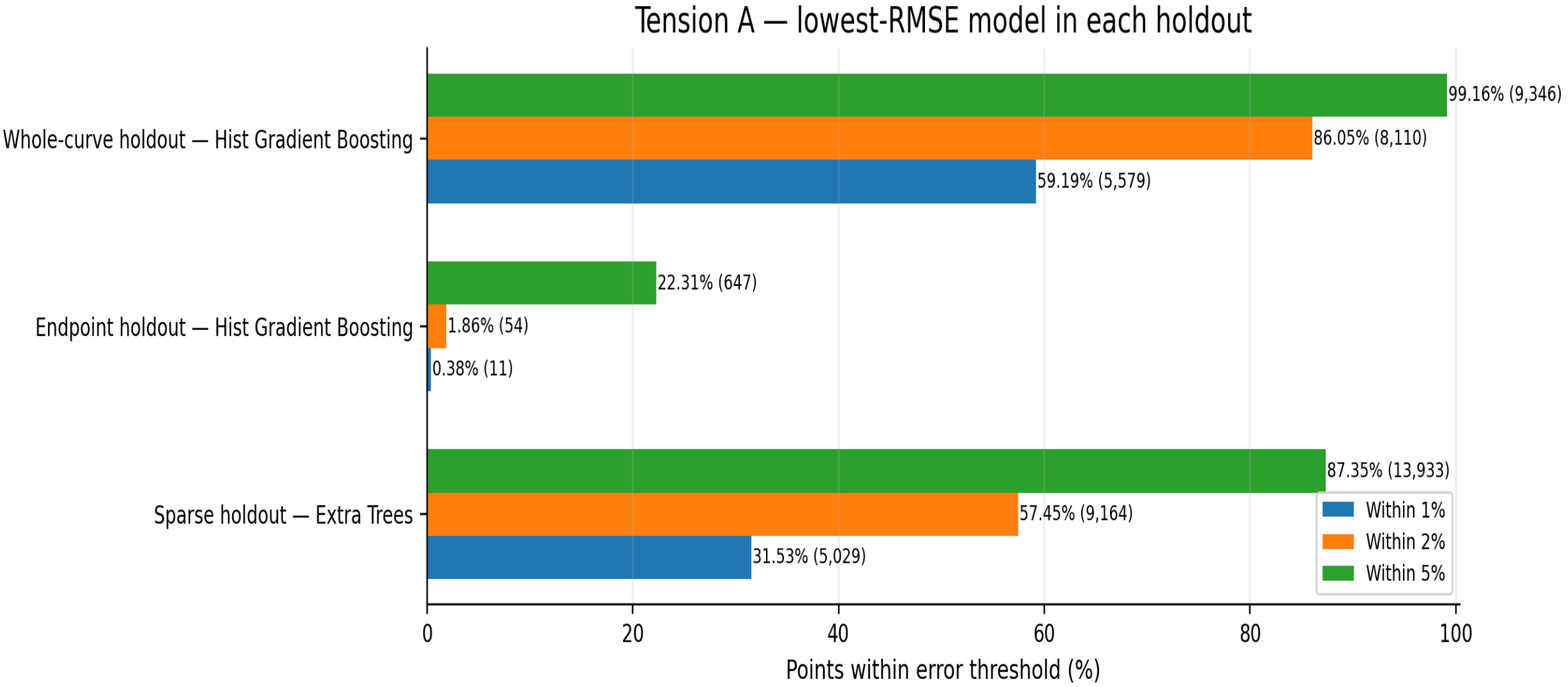


Sparse holdout uses Extra Trees because it was the lowest-RMSE model in that holdout; the other two shown here use Hist Gradient Boosting.

Structured holdouts better matched how an ML model may be asked to fill missing solution regions.

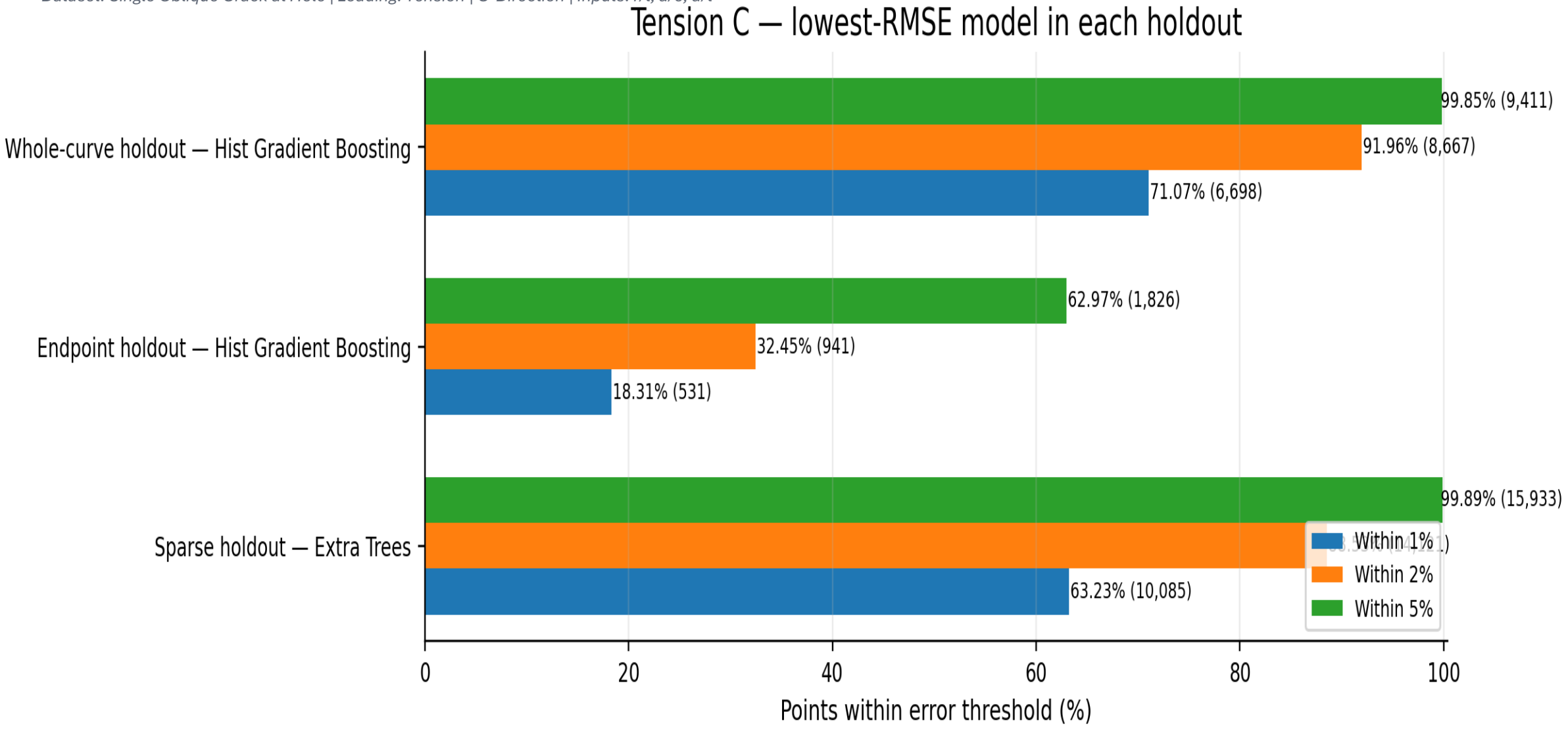
Expanded View: Structured Holdout Results for Tension A

Dataset: Single Oblique Crack at Hole | Loading: Tension | A-Direction | Inputs: r/t, a/c, a/t



Expanded View: Structured Holdout Results for Tension C

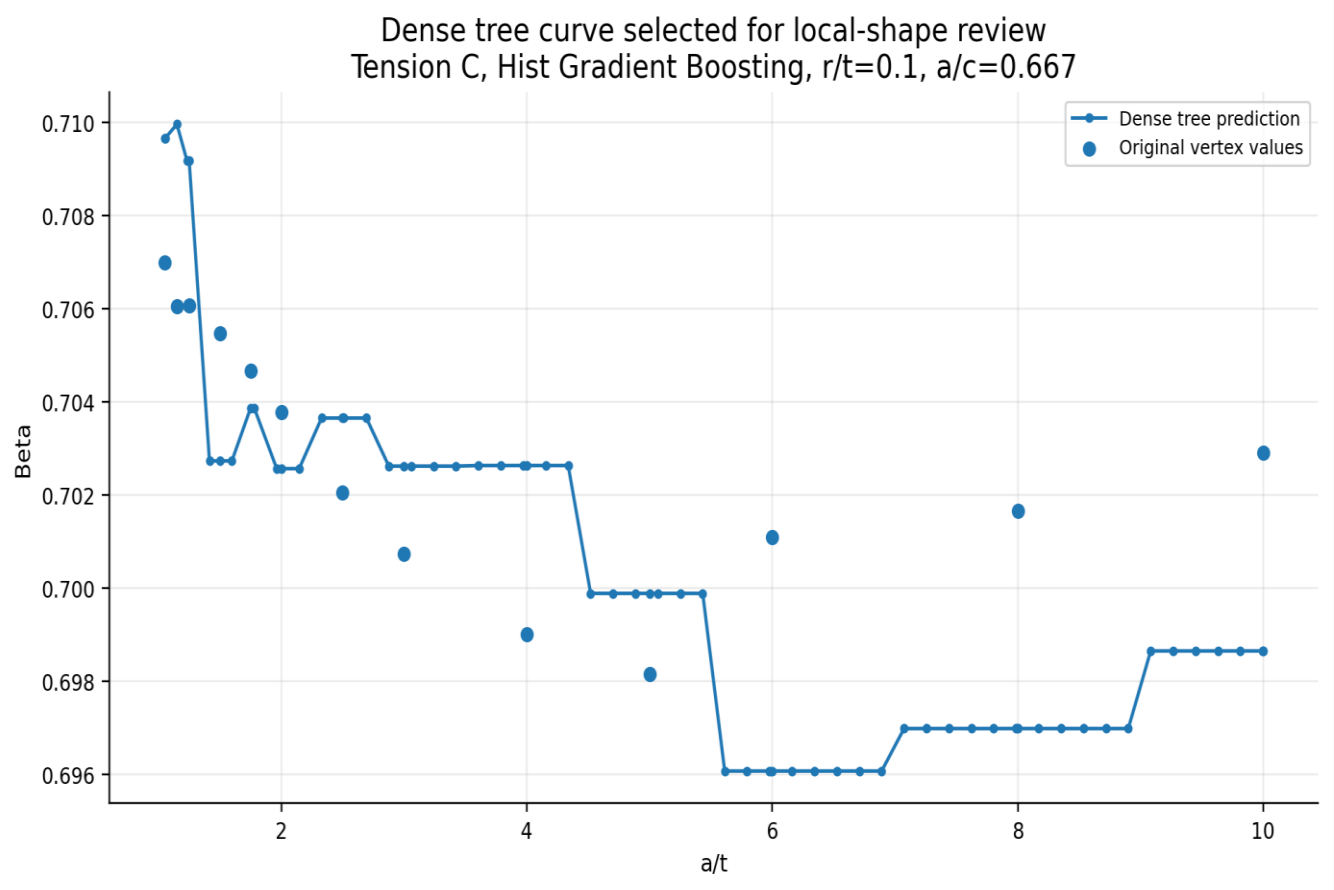
Dataset: Single Oblique Crack at Hole | Loading: Tension | C-Direction | Inputs: r/t, a/c, a/t



Low Pointwise Error Did Not Guarantee Good Curve Behavior

- Additional predictions were generated between the original grid locations (dense sampling) to make the model's local behavior visible.
- Tree models could remain numerically accurate at known points while still producing piecewise or locally irregular behavior between them.
- The Tension C Hist Gradient Boosting example shown was selected by a shape audit as a clear case of this local behavior.
- The example shown is a Tension C HGB curve selected by a shape audit at r/t = 0.1 and a/c = 0.667.
- This shifted the goal from minimizing pointwise error alone to achieving both numerical accuracy and reliable curve behavior.

Dataset: Single Oblique Crack at Hole | Loading: Tension | C-Direction | Slice: r/t = 0.1, a/c = 0.667



Accuracy at evaluated points is necessary—but not sufficient.

Validation Expanded from Point Error to Curve Behavior

Midpoint Interpolation

Checks local behavior between neighboring known points.

Midpoint bounds

Flags midpoint predictions leaving the neighboring Beta range.

Direction mismatch

Checks whether predicted local rise/fall direction agrees with the reference trend.

Visual review

Uses an internal visual review tool to inspect smoothness and local curve behavior.

Slope-range flags

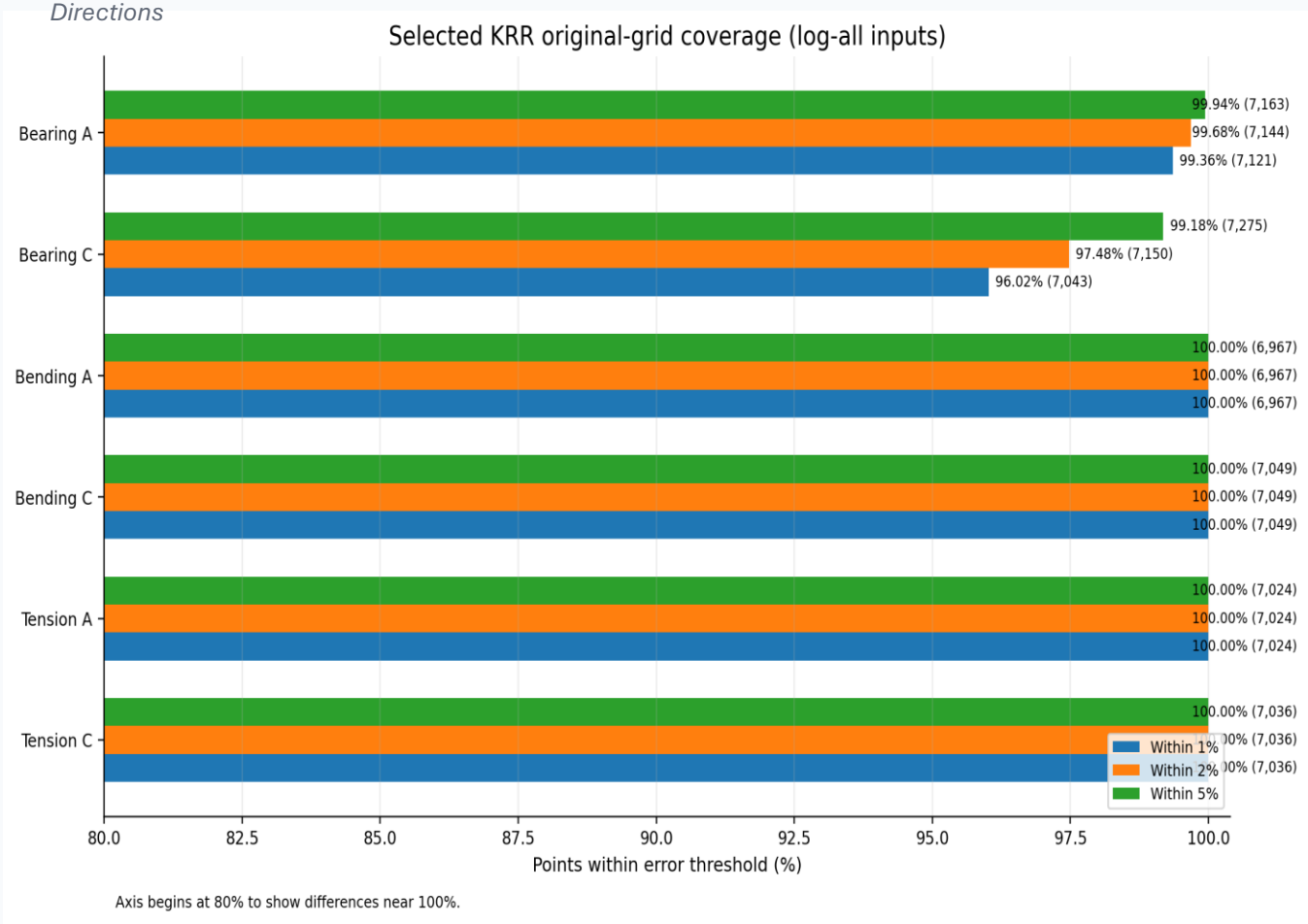
Flags unusually steep or weak local changes.

Important: midpoint reference values are a local consistency diagnostic, not independent solver ground truth.

KRR Showed That Smooth Curve Behavior Was Achievable

- RBF Kernel Ridge Regression produced a smooth nonlinear response surface instead of the piecewise behavior seen with tree-based models.
- Input transformations and parameter tuning were important for obtaining both strong numerical fit and usable curve behavior.
- Selected KRR candidates reconstructed more than 99% of the original-grid points within 5% across all six tested datasets.
- These results are full-grid reconstruction diagnostics and should not be interpreted as independent holdout validation.

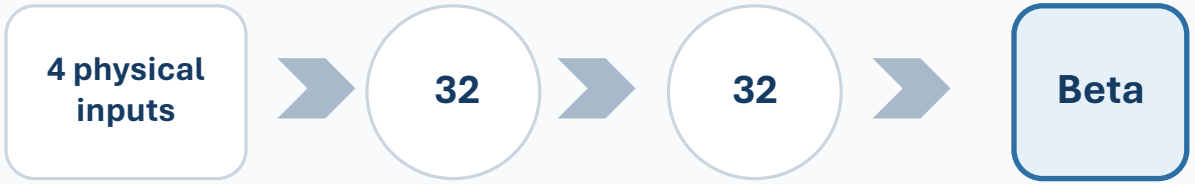
Dataset family: Single Oblique Crack at Hole | Loadings: Bearing, Bending, and Tension | A- and C-Directions



KRR showed that smooth nonlinear ML-based representation was feasible.

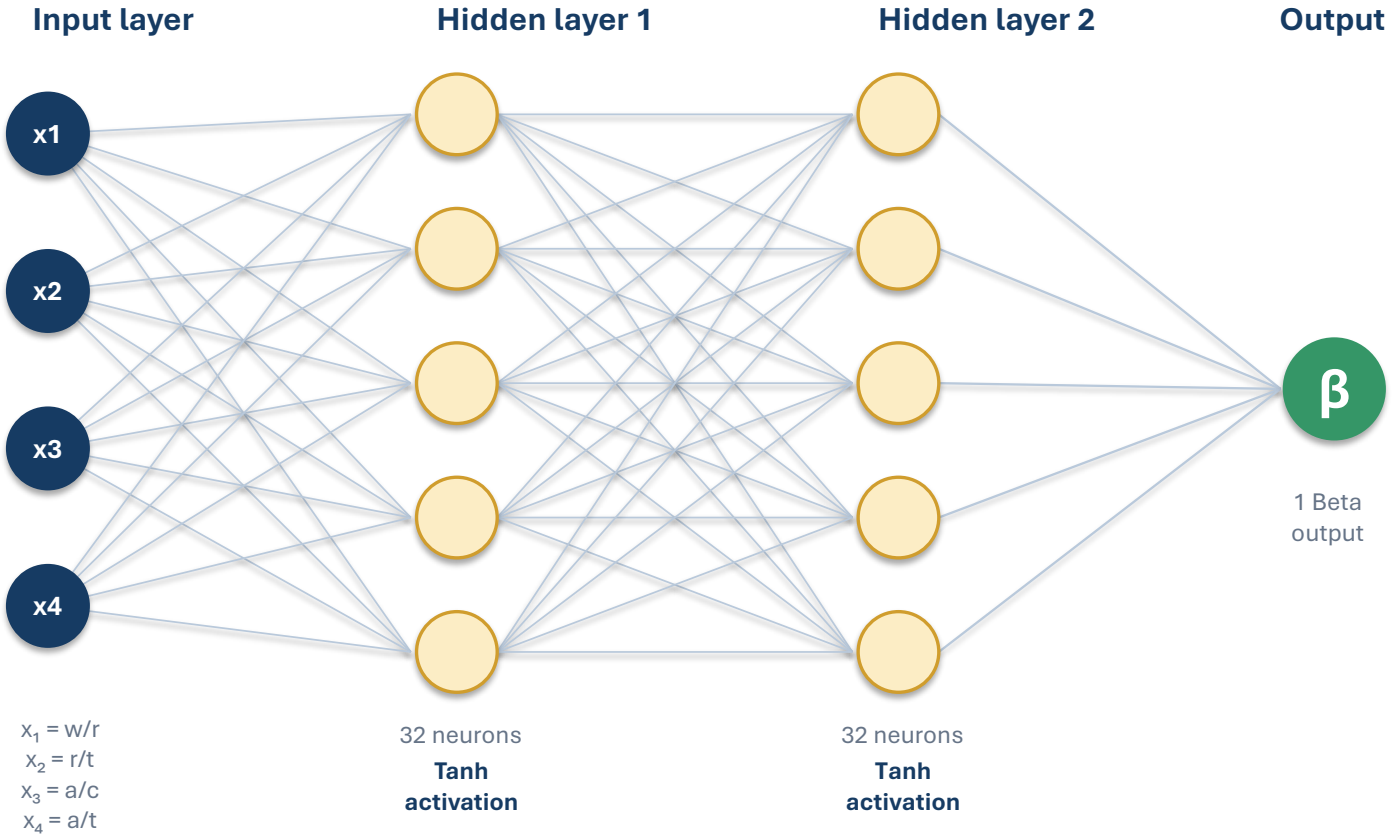
Why Move to a Multilayer Perceptron?

- KRR produced several promising α-γ combinations with strong accuracy and smooth curve behavior, but extending exact kernel methods to the much larger finite-width dataset introduced increasing computational cost.
- Kernel approximations such as Nyström could have been investigated further, but the project instead expanded toward neural networks as another continuous modeling approach for the larger multidimensional problem.
- The MLP learns one continuous relationship across all interacting inputs and can incorporate additional parameters, such as w/r, without changing the overall modeling framework.
- The current finite-width network is intentionally small, with two hidden layers of 32 neurons using Tanh activation and only 1,249 trainable parameters, so the investigation remains focused on whether a compact model can provide accurate and well-behaved predictions.



Inside the MLP: From Weighted Sums to Nonlinear Features

Each hidden neuron forms a linear weighted sum first; the Tanh activation function then transforms it nonlinearly.



What one hidden neuron does

Input vector

$x = [w/r,\ r/t,\ a/c,\ a/t]^T$

Learned weights

$\theta = [\theta_1, \theta_2, \theta_3, \theta_4]^T$

Linear step

$z = \theta^T x + b$

Activation function

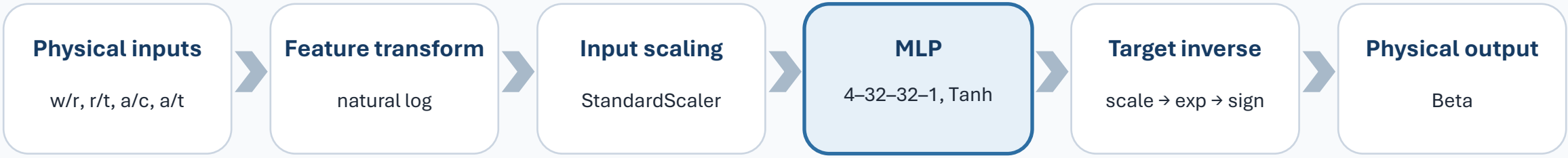
$h = \tanh(z)$

linear weighted sum → nonlinear hidden output

Layer-by-layer view

$z_1 = W_1 x + b_1 \rightarrow h_1 = \tanh(z_1)$ $\quad z_2 = W_2 h_1 + b_2 \rightarrow h_2 = \tanh(z_2)$ $\quad \beta \hat{=} W_3 h_2 + b_3$

Current MLP Inference Pipeline



Two-fit design

Evaluation model

80% training split → 20% independent test
Used only for train/test performance metrics

Full-data model

Refit independently on all original rows
Used for final model output, Visual files, and curve diagnostics

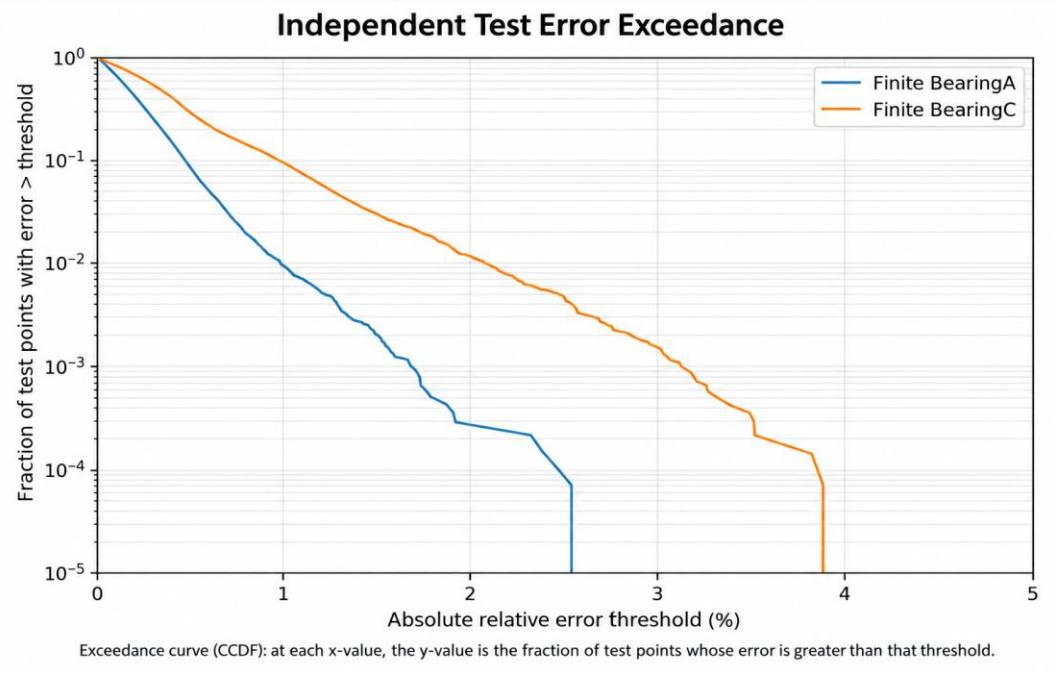
This separation keeps test evidence distinct from the final trained model used for output generation.

Current MLP: Independent Test Performance

Dataset	Within 1%	Within 5%	P95 error	Max error
Finite Bearing A	99.06%	100.00%	0.61%	2.62%
Finite Bearing C	90.27%	100.00%	1.27%	4.58%

13,837 held-out test rows per dataset

- Both finite-width MLPs predicted every held-out random-split test point within 5% error.
- Finite Bearing A was especially strong, with 99.06% of test points within 1% and a P95 error of 0.61%.
- Finite Bearing C was more difficult, with 90.27% within 1% and a P95 error of 1.27%, showing a noticeably heavier error tail.
- These results come from the independent 20% test split and are kept separate from the full-data reconstruction results used later for curve diagnostics.



Dataset: Finite-Width Single Corner Crack at Hole | Loading: Bearing | A- and C-Directions | Inputs: w/r, r/t, a/c, a/t

Error = absolute relative error in Beta.

The MLP Is Also Checked Across Every Input Direction

Dataset: Finite-Width Single Corner Crack at Hole | Loading: Bearing | A- and C-Directions | Inputs: w/r, r/t, a/c, a/t

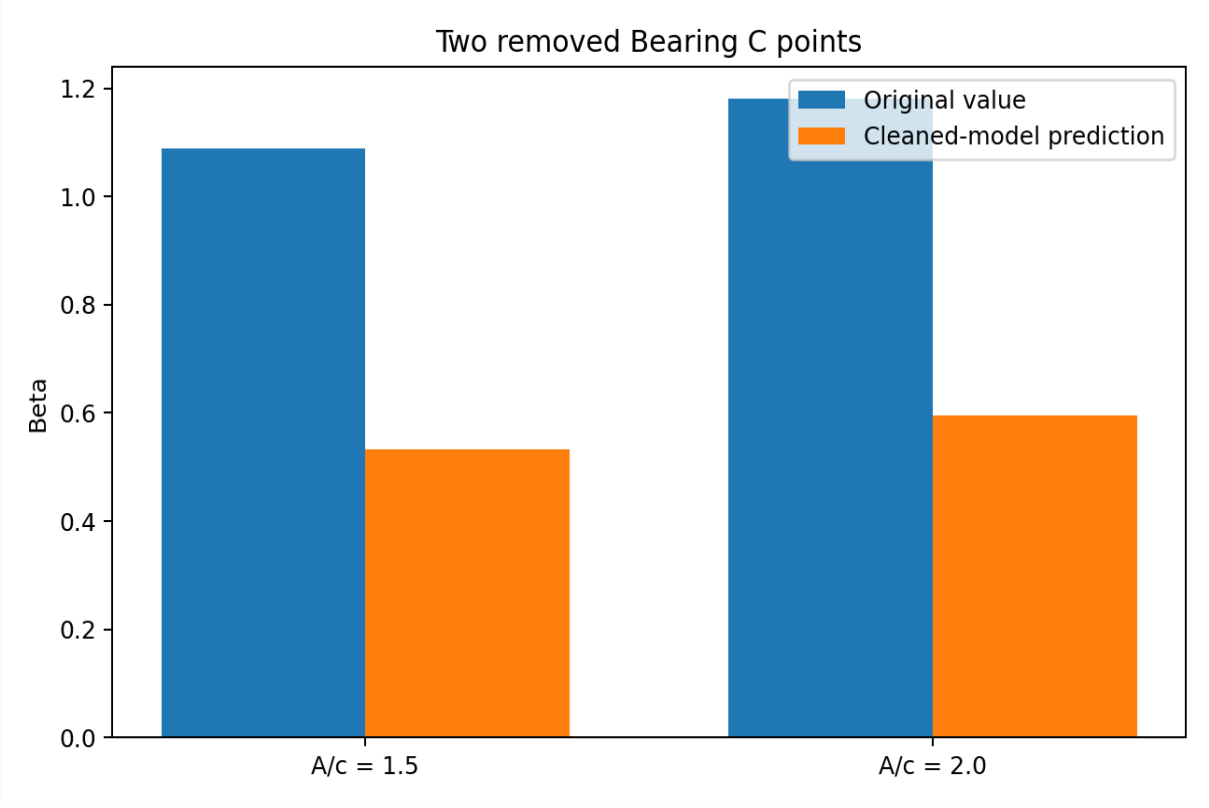
- The current diagnostic workflow evaluates w/r, r/t, a/c, and a/t, rather than checking only the primary a/t curves.
- Opposite-direction mismatches were low across all axes, at approximately **0.24% for Finite Bearing A** and **0.13% for Finite Bearing C**.
- Padded slope-flag rates were approximately **2.87% and 1.78%**, showing where local curve behavior still deserves closer inspection.
- These are full-data shape diagnostics, so they complement rather than replace the independent holdout results shown earlier.

Goal: one multidimensional model that remains usable when viewed from different directions.

A Second Use Case: Flagging Data That Warrants Review

- Two Finite Bearing C points were removed when preparing the cleaned dataset.
- A model trained on the cleaned surrounding data predicted Beta values roughly 50% away from the original supplied values at those same coordinates.
- The large residuals show that these points are strongly inconsistent with the surrounding multidimensional relationship learned by the model.
- Large residuals can therefore be used as a practical screening signal to identify data points that deserve closer engineering review.

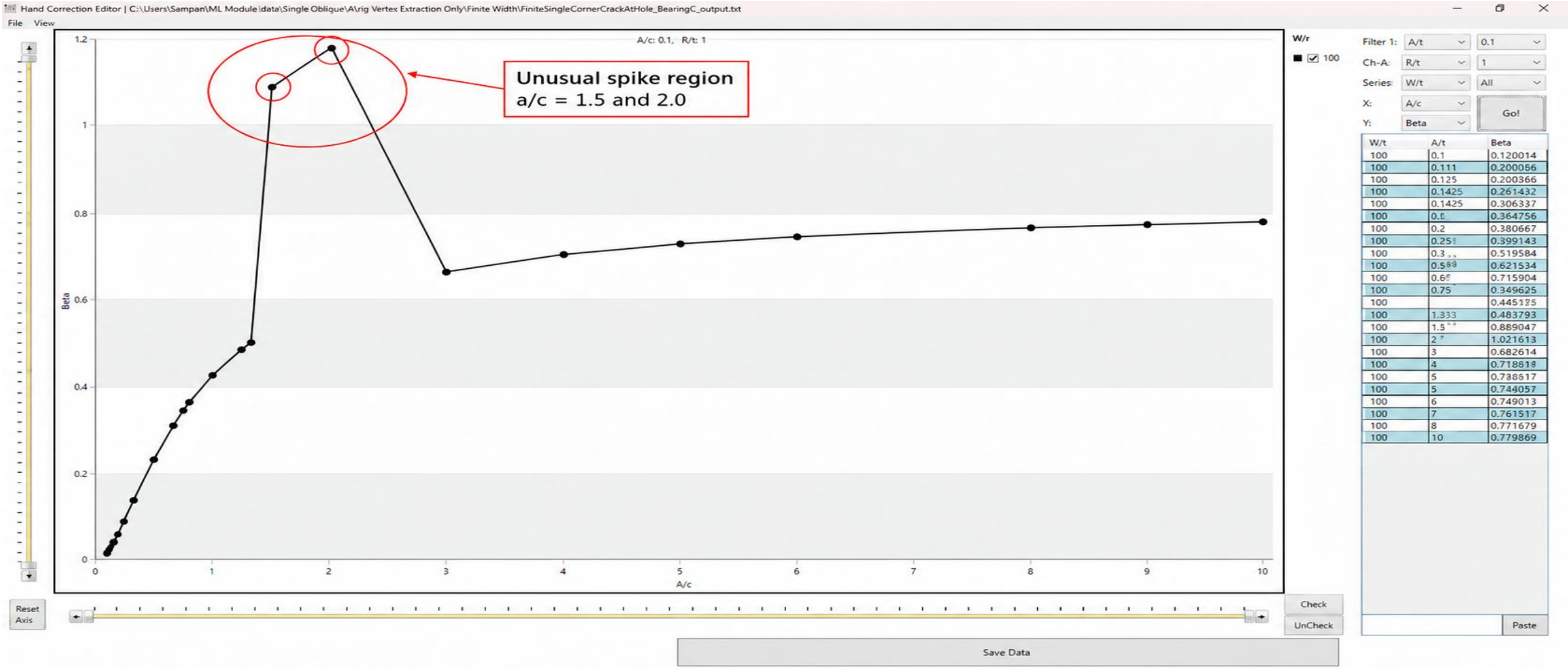
Dataset: Finite-Width Single Corner Crack at Hole | Loading: Bearing | C-Direction | Removed points at a/c = 1.5 and 2.0



Removed locations: w/r = 100, r/t = 1, a/t = 0.1, with a/c = 1.5 and 2.0

Residual screening → prioritize where an engineer looks first

Original Finite Bearing C Curve Highlighting the Outlier Region



Dataset: Finite-Width Single Corner Crack at Hole | Loading: Bearing | C-Direction | Slice: w/r = 100, r/t = 1, a/t = 0.1

What Could This Add to the AFGROW Workflow?

Multidimensional representation

Learn interacting inputs together instead of treating each visible slice as a separate fitting problem.

Reduced manual curve work

Significantly reduce repeated fitting, smoothing, and cross-dimensional checking as solution spaces become larger.

Targeted data review

Use large residuals or unusual local behavior to direct engineer attention toward the points most worth reviewing.

Compact scalable ML model

Once properly validated, a trained model can evaluate dense combinations of inputs without repeating the original fitting process.

The current work is exploratory R&D: these are potential workflow benefits, not production claims.

From Research Model to Portable Inference Prototype

- The full-data MLP was exported to ONNX so it could be evaluated outside the original Python training environment.
- The exported model includes the required input transformations, scaling, network prediction, and inverse target transformation needed to return physical Beta.
- ONNX Runtime predictions were checked against the original PyTorch pipeline to verify that the exported model reproduced the same results.
- The prototype was successfully tested on a separate computer, showing that the trained model could be transferred and executed independently of the development setup.
- This is a first portability demonstration, not an AFGROW integration; additional interface, validity-range, and software-integration work would still be required.

Conclusions and Next Steps

What this investigation currently supports

- Machine-learning models can represent the tested Beta relationships with strong numerical accuracy.
- Engineering validation needs to follow the structure of the K-solution rather than rely on random-row accuracy alone.
- KRR and the current MLP both show that smooth multidimensional ML models are technically promising.
- The ONNX prototype demonstrates a practical path from a trained research model to portable inference.

What still needs to be demonstrated

- Run stronger structured MLP holdouts on the finite-width datasets.
- Compare the ML approach directly with the current interpolation workflow using common reference cases and tolerances.
- Define model validity limits and expected behavior near or outside the trained solution domain.
- Evaluate larger and more varied solution sets before considering operational use.

ML is being investigated as an additional K-solution development and review capability within the existing AFGROW workflow.

Questions?