Real-Time Surrogate Modeling of Tracked Vehicle Terramechanics via Run-Level Learning and Deterministic Verification

Hüseyin Umutcan Şanlı[1] and Mehmet Önder Efe[2]

***Abstract*— Real-time simulation of tracked skid-steer vehicles is frequently bottlenecked by the computational cost of track–terrain interaction models. To address this, we replace the physics-based terramechanics function in a Simulink model with a neural network surrogate tailored for hardware-in-the-loop deployment. We train separate Multi-Layer Perceptrons (MLPs) for rigid (friction-limited) and soil (strength-limited) regimes to predict body-frame forces and yaw moment (Q_x, Q_y, M_z). Crucially, we deviate from standard offline training by enforcing *input truthfulness*: dataset generation relies strictly on logged block interfaces—including signal delays and solver artifacts—rather than idealized command profiles. To prevent temporal leakage, we utilize run-level data splitting and a custom weighted loss function that penalizes errors in the sensitive yaw moment channel. Finally, we demonstrate that a manual MEX deployment eliminates runtime overhead, reducing latency below the physics baseline to enable real-time execution.**

***Index Terms*— tracked vehicles, terramechanics, surrogate modeling, neural networks, system identification, real-time simulation**

I. INTRODUCTION

Autonomous tracked vehicle simulation often forces a choice: high-fidelity terramechanics that break real-time constraints, or simplified kinematics that fail to capture skid-slip dynamics. For tracked skid-steer vehicles, the dominant computational cost frequently originates from track–terrain interaction models that integrate shear stresses over the contact patch and apply geometry-dependent moment arms [1]. While such terramechanics models provide physically meaningful predictions, their step-time cost can become a bottleneck in closed-loop simulation, hardware-in-the-loop, or game-engine co-simulation [2].

This paper addresses the problem of approximating the terramechanics interaction block with a fast black-box surrogate suitable for real-time deployment. Recent approaches have demonstrated the effectiveness of neural networks for off-road mobility prediction [3] and high-fidelity vehicle dynamics modeling [4]; building on these concepts, we target the specific replacement of the internal force computation block. The target block computes body-frame forces and yaw moment

$$[Q_x,\ Q_y,\ M_z] = f(V_x, V_y, \omega, a_x, a_y, V_{tL}, V_{tR}, \theta)\,, \quad (1)$$

where, (V_x, V_y, ω) are vehicle planar velocities and yaw rate, (a_x, a_y) are planar accelerations, (V_{tL}, V_{tR}) are left/right sprocket (belt) speeds, and θ is a terrain parameter vector. Two operating regimes are considered: (i) *rigid* terrain, where shear is friction-limited, and (ii) *soil* terrain, where shear is strength-limited.

A. Contributions

We log the exact Simulink block inputs (including the one-step acceleration delay) and train separate rigid/soil MLPs with run-level splits and a yaw-weighted Huber loss. On held-out runs, the models match the physics block closely across all three outputs, with the largest errors appearing in M_z under aggressive PRBS maneuvers. A manual MEX forward pass cuts per-step latency below the physics baseline, which is the key requirement for real-time deployment.

B. Paper Organization

Section II summarizes the terramechanics interaction model being replaced; Section III describes dataset generation, signal alignment, and leakage prevention; Section IV presents the surrogate model, train-only normalization, and the training procedure; Section V reports test-split performance and tail-error characterization; Section VI evaluates generalization under unseen maneuvers and held-out terrain regimes; Section VII analyzes the accuracy–latency trade space via an architecture sweep and latency-aware operating points; and Section VIII reports runtime benchmarks and discusses deployment implications for real-time simulation.

II. TERRAMECHANICS INTERACTION MODEL OVERVIEW

The surrogate replaces a terramechanics interaction function that computes net track forces and yaw moment by distributing normal load across discrete track contact elements and integrating shear stresses over a discretized contact patch, following the general theory for skid steering [5].

A. Normal Load Distribution

For each track (left/right), the normal load on a set of contact elements is computed from a static component plus load transfer terms driven by vehicle accelerations and geometry [5]. The resulting per-element normal force is nonnegative and defines the mean normal pressure over that element's contact patch.

B. Shear Stress Law and Patch Integration

For each contact element, the contact patch is discretized into a grid of cells [6]. At each cell, the local relative slip velocity $\mathbf{v}_s = [V_{sx}, V_{sy}]^\top$ between track and ground is computed from the vehicle planar motion, yaw rate, and

[1]Hüseyin Umutcan Şanlı is with Land Platform Technologies, HAVELSAN, Ankara, Türkiye husanli@havelsan.com.tr

[2]Mehmet Önder Efe is with the Department of Computer Engineering, Hacettepe University, Ankara, Türkiye onderefe@ieee.org

belt speed. A slip distance measure j is formed using a soft contact-time approximation, and a Janosi–Hanamoto-style saturation [7] is applied:

$$\tau(j) = \tau_{\max}\left(1 - e^{-j/K}\right), \tag{2}$$

where, K is a shear deformation modulus. For rigid terrain, $\tau_{\max} = \mu p$ (friction-limited), with normal pressure p and friction coefficient μ. For soil, $\tau_{\max} = c + p\tan(\phi)$ (strength-limited) [8], with cohesion c and internal friction angle ϕ. The cell forces are then integrated over area A_{cell} using the slip direction to obtain contributions to Q_x and Q_y. This formulation remains widely used in modern fast dynamic modeling [9].

C. Yaw Moment and Resistive Terms

The yaw moment M_z is formed by summing moment contributions from all cell forces using their lever arms relative to the body frame. In addition to shear forces, rolling resistance is modeled as a resistive longitudinal term whose sign follows belt rolling direction and is added consistently to the track longitudinal force and moment balance. Because M_z is the difference of large left/right force distributions applied at finite lever arms, it is typically more sensitive to modeling error than Q_x and Q_y; this motivates explicit emphasis on M_z during training.

III. Dataset Generation and Integrity Controls

The dataset is generated directly from simulation to ensure the surrogate learns the same mapping as the original block. Each run corresponds to a maneuver executed under a sampled terrain parameter set, producing time-series samples that are stored as run-indexed arrays to preserve trajectory boundaries.

A. Simulation Settings and Run Structure

A fixed-step solver is used with sample time $T_s = 0.01\,\text{s}$ and a finite run duration, discarding an initial transient window to remove start-up artifacts. Each run is stored as a cell entry $\mathbf{X}^{(i)} \in \mathbb{R}^{N_i \times d}$, $\mathbf{Y}^{(i)} \in \mathbb{R}^{N_i \times 3}$, where $d = 10$ (rigid) or $d = 11$ (soil), allowing small variations in logged run length while preserving run-level boundaries.

B. Inputs, Outputs, and Feature Order

Both surrogates predict the same targets $[Q_x, Q_y, M_z]$. The complete set of input features and output targets for both rigid and soil regimes is detailed in Table I.

TABLE I
SURROGATE INPUTS AND OUTPUTS (PER TIME STEP).

Mode	Features
Rigid ($d = 10$)	$[V_x, V_y, \omega, a_x, a_y, V_{tL}, V_{tR}, \mu, f_r, K]$
Soil ($d = 11$)	$[V_x, V_y, \omega, a_x, a_y, V_{tL}, V_{tR}, c, \phi, f_r, K]$
Targets (3)	$[Q_x, Q_y, M_z]$

C. Input Truthfulness and Alignment Fixes

Standard open-loop data collection often yields poor surrogates because it relies on idealized command profiles rather than the actual signals processed by the solver. Our pipeline enforces *input truthfulness* by logging the exact block interface during execution. This is particularly critical for acceleration inputs (a_x, a_y), which are frequently routed through memory blocks to break Simulink algebraic loops. Neglecting the resulting single-step delay (z^{-1}) causes a temporal misalignment between the state and the force response. By training on the logged, delayed signals rather than the upstream commands, the surrogate naturally learns to compensate for these solver-induced latencies and eliminates potential left/right mapping errors inherent in manual signal reconstruction.

D. Terrain Sampling and Maneuvers

Terrain parameters are sampled per run from bounded ranges for both modes, and each run selects a maneuver profile from a finite set (e.g., straight, turn, skid reversal, pivot, reversal, and PRBS-like excitation) to excite a broad operating envelope. Per-run metadata (terrain parameters, maneuver type, and excitation scalars) is stored alongside $\mathbf{X}^{(i)}, \mathbf{Y}^{(i)}$ to enable downstream stress testing and held-out evaluations.

IV. Surrogate Model and Training Procedure

Two separate multi-output regression surrogates are trained for rigid and soil modes, both using a feedforward MLP due to its favorable inference latency and ease of deployment.

A. Normalization and Run-Level Splitting

To prevent leakage and enable deployment, splitting and normalization follow strict rules:

- **Run-level split:** runs are partitioned into train, validation, and test sets, avoiding time-step mixing across splits.
- **Train-only normalization:** inputs and outputs are standardized using z-score statistics computed on the training set only; the resulting normalization statistics are saved in the model artifact and reused for validation, test, and deployment.

B. Network Architecture and Loss

The baseline surrogate is a configurable MLP with dropout and L_2 regularization. Training uses a robust Huber loss for regression with output weights to emphasize yaw moment accuracy. This loss function down-weights extreme outliers (common in transient simulation spikes) while maintaining optimization stability [10]:

$$\mathcal{L} = \sum_{k \in \{Q_x, Q_y, M_z\}} w_k \cdot \text{Huber}(\hat{y}_k - y_k)\,, \tag{3}$$

where, $w_{Q_x} = 1$, $w_{Q_y} = 1$, and $w_{M_z} > 1$ (set to 3 in our baseline configuration).

C. Optimization and Regularization

Training is performed using the Adam optimizer [11] with mini-batches on normalized data. Regularization combines dropout (to reduce co-adaptation) and L_2 weight decay. Early stopping is driven by validation performance, together with a reduce-on-plateau learning rate schedule to improve convergence stability.

D. Reproducibility and Verification Utilities

Beyond standard training logs, the pipeline includes deterministic verification utilities that check: (i) array shapes and NaN/Inf values, (ii) normalization round-trip correctness, and (iii) regression tests against stored baselines. These checks are designed to detect latent dataset misalignment and prevent silent performance regressions when the simulation model, feature extraction, or training scripts change.

V. EXPERIMENTAL RESULTS

This section reports open-loop prediction performance of the rigid and soil surrogates on held-out runs. All metrics are computed after denormalizing network outputs back to physical units.

A. Evaluation Protocol and Metrics

Datasets are split at the *run level* into train/validation/test partitions to avoid temporal leakage. For each test run, the surrogate predicts $\hat{\mathbf{y}}_t = [\hat{Q}_x, \hat{Q}_y, \hat{M}_z]$ from the feature vector $\mathbf{x}_t$ in Table I, and predictions are compared to the physics model outputs $\mathbf{y}_t = [Q_x, Q_y, M_z]$.

B. Experimental Setup

All experiments were executed on an Intel Core i7-13700HX CPU using MATLAB R2024b. Each dataset (rigid and soil) contains 300 runs and 1,140,300 samples in total (average 3,801 samples/run). The run-level split is 70%, 15%, and 15% (train/validation/test), yielding 210/45/45 runs and 798,210 / 171,045 / 171,045 samples per split in each mode.

We report standard regression metrics per output channel: (i) root-mean-square error (RMSE), (ii) mean absolute error (MAE), (iii) coefficient of determination (R^2), (iv) signed mean error (Bias), (v) 95th percentile absolute error (P95Abs), and (vi) maximum absolute error (MaxAbs). For example,

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(\hat{y}_t - y_t)^2}, \tag{4}$$

and $\mathrm{Bias} = \frac{1}{N}\sum_{t=1}^{N}(\hat{y}_t - y_t)$. When the target variance is near zero for a given subset (e.g., straight motion for Q_y and M_z), R^2 is not informative and is treated as invalid in the evaluation scripts.

TABLE II

RIGID SURROGATE TEST METRICS (TEST SPLIT).

Out	RMSE	MAE	R^2	Bias	P95Abs	MaxAbs
Q_x	1238.18	640.37	0.98848	-196.97	2862.90	10906.93
Q_y	743.01	345.39	0.99630	-154.36	1770.95	6791.31
M_z	893.65	422.78	0.98727	-49.01	2053.20	10690.81

TABLE III

SOIL SURROGATE TEST METRICS (TEST SPLIT).

Out	RMSE	MAE	R^2	Bias	P95Abs	MaxAbs
Q_x	1020.91	507.54	0.98851	-5.48	2254.89	9164.10
Q_y	501.26	210.32	0.99373	-22.60	1105.96	9298.07
M_z	772.22	319.54	0.98616	-24.61	1623.94	12063.18

C. Test-Split Accuracy

Tables II and III summarize test-split performance for the saved baseline models. Across both modes, all channels achieve high R^2, with yaw moment M_z exhibiting the largest tail errors, consistent with its sensitivity to left/right force asymmetries and lever arms. Fig. 1 shows parity plots on the test split for rigid and soil modes.

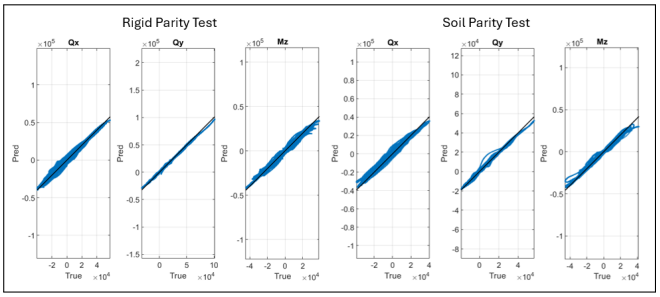

Fig. 1. Test-split parity plots for rigid (left) and soil (right) surrogates.

D. Baseline and Ablation Study

To contextualize the MLP performance, we include a simple linear ridge regression baseline trained on the same run-level splits and normalized data. We also perform controlled ablations of the MLP training configuration by toggling the loss (Huber vs. MSE), output weighting, dropout, and L_2 regularization. Table IV summarizes the resulting accuracy trade-offs for rigid and soil modes. In both modes, the ridge baseline is significantly worse than the MLP baseline (e.g., M_z RMSE ≈ 5475 rigid and ≈ 5144 soil vs. ≈ 894 and ≈ 772 for the MLP). Removing output weights increases M_z error (1222 rigid, 900 soil), confirming the benefit of yaw-moment weighting. On this split, disabling dropout yields the best mean RMSE (598 rigid, 557 soil), indicating that dropout may be overly aggressive for these datasets and should be tuned rather than fixed.

E. Worst-Run Analysis and Tail Behavior

To characterize tail risk, per-run metrics are computed on the test split and runs are ranked by M_z RMSE. Table V lists

TABLE IV

BASELINE AND ABLATION RESULTS FOR RIGID AND SOIL MODES.

Rigid

Variant	$\mathbf{RMSE}_{Mz}$	R^2_{Mz}	$\mathbf{RMSE}_{mean}$	R^2_{mean}
Baseline MLP	893.65	0.9873	958.28	0.9907
Ridge	5475.19	0.5222	2800.74	0.8282
MSE loss	805.53	0.9897	1102.21	0.9892
No weights	1222.41	0.9762	1063.73	0.9870
No dropout	628.25	0.9937	598.42	0.9960
No L2	735.71	0.9914	958.16	0.9917

Soil

Variant	$\mathbf{RMSE}_{Mz}$	R^2_{Mz}	$\mathbf{RMSE}_{mean}$	R^2_{mean}
Baseline MLP	772.22	0.9862	764.80	0.9895
Ridge	5143.82	0.3858	2423.56	0.7837
MSE loss	687.00	0.9890	877.55	0.9850
No weights	899.84	0.9812	757.98	0.9889
No dropout	680.71	0.9892	556.78	0.9939
No L2	739.44	0.9873	834.37	0.9874

TABLE V

WORST TEST RUNS BY M_z RMSE (TOP 3 PER MODE).

Mode	Run	$\mathbf{RMSE}_{M_z}$	P95Abs	MaxAbs
Rigid	196 (prbs)	2437.33	5023.22	9388.98
Rigid	26 (prbs)	2258.80	4142.86	6193.62
Rigid	172 (prbs)	2199.16	4541.38	10690.81
Soil	275 (prbs)	2647.13	5396.02	12059.41
Soil	86 (prbs)	1935.20	3543.59	6977.21
Soil	97 (skid_reversal)	1611.87	2642.87	12063.18

the top worst cases for each mode. In both modes, pseudo-random (PRBS) excitation dominates the worst M_z errors, indicating that aggressive, rapidly varying differential track commands are the primary stress regime for a memoryless surrogate.

Fig. 2 compares physics vs surrogate on the worst M_z runs for each mode.

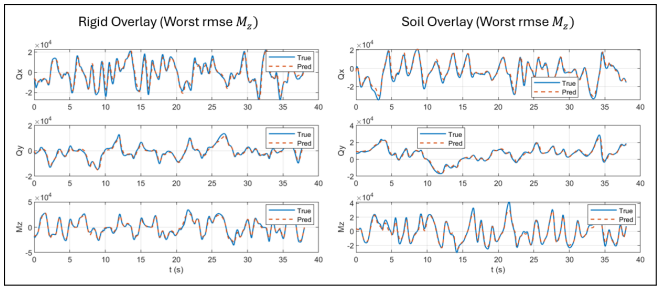

Fig. 2. Worst-run time-series overlays for M_z on the test split: rigid (left, run 196) and soil (right, run 275).

VI. GENERALIZATION STRESS TESTS

Beyond the standard test split, two stress-test protocols are used to probe generalization under distribution shift: (i) unseen maneuver holdout (leave-one-maneuver-out) and (ii) held-out terrain regimes using a percentile-based strategy. These tests directly target failure modes that are masked by in-distribution sampling.

A. Unseen Maneuver Holdout

In the unseen maneuver evaluation, models are assessed on maneuvers not present in training, and results are summarized across holdout folds. Table VI reports the minimum and median valid R^2 over holdout maneuvers. PRBS is consistently the most challenging maneuver, especially for M_z, which is expected due to rapid transient forcing.

TABLE VI

UNSEEN MANEUVER GENERALIZATION.

Mode	Out	$\mathbf{R}^2_{min}$	$\mathbf{R}^2_{med}$	Worst maneuver
Rigid	Q_x	0.7906	0.9959	prbs
Rigid	Q_y	0.9515	0.9944	prbs
Rigid	M_z	0.6032	0.9845	prbs
Soil	Q_x	0.8453	0.9967	prbs
Soil	Q_y	0.9270	0.9935	prbs
Soil	M_z	0.6526	0.9707	prbs

TABLE VII

HELD-OUT TERRAIN GENERALIZATION.

Mode	Out	$\mathbf{R}^2_{min}$	$\mathbf{R}^2_{med}$	Worst scenario
Rigid	Q_x	0.9764	0.9939	μ_{low} ($\mu \leq 0.5163$)
Rigid	Q_y	0.9969	0.9989	μ_{low} ($\mu \leq 0.5163$)
Rigid	M_z	0.9421	0.9808	μ_{low} ($\mu \leq 0.5163$)
Soil	Q_x	0.8838	0.9943	ϕ_{low} ($\phi \leq 9.0126°$)
Soil	Q_y	0.9691	0.9985	ϕ_{low} ($\phi \leq 9.0126°$)
Soil	M_z	0.8157	0.9802	ϕ_{low} ($\phi \leq 9.0126°$)

B. Held-Out Terrain Regimes

To evaluate domain shift in terrain parameters, a percentile-based holdout strategy withholds extreme parameter ranges during training and evaluates on those regimes. Table VII summarizes the minimum and median R^2 across terrain scenarios, together with the most challenging regime thresholds. For rigid terrain, low μ is the hardest regime; for soil, low ϕ is the hardest regime, with the largest sensitivity observed in M_z.

Fig. 3 visualizes parity under the hardest maneuver (PRBS) and the hardest terrain regimes (low μ, low ϕ).

C. Discussion

The stress tests indicate that performance is localized to in regimes where (i) excitation is highly transient (PRBS) and (ii) terrain traction limits shift the operating envelope (low μ or low ϕ). These regimes are consistent with the worst-run tail behavior observed on the standard test split. This suggests that further robustness should focus on transient handling for M_z (e.g., short temporal context via lag features or a lightweight temporal front-end) and on explicit coverage/weighting of extreme traction regimes during dataset generation.

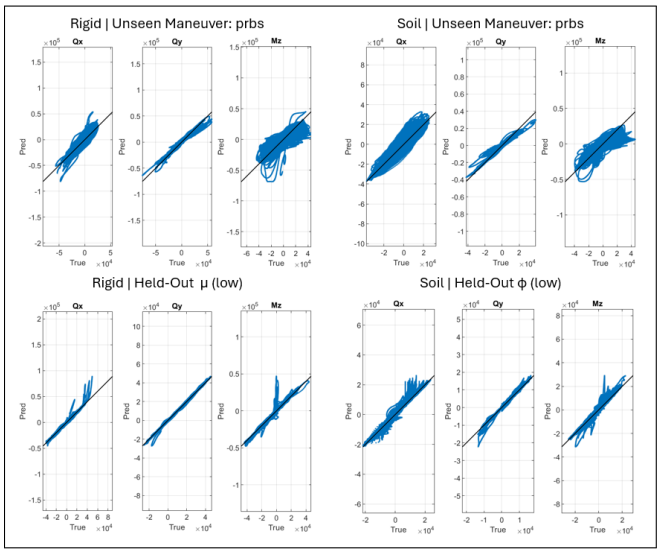

Fig. 3. Generalization hard cases: unseen PRBS maneuver parity (top row) and held-out terrain worst-regime parity (bottom row).

VII. Model Complexity and Architecture Sweep

The baseline surrogate uses a fully-connected multi-layer perceptron (MLP) to map the normalized feature vector to the interaction outputs $\{Q_x, Q_y, M_z\}$. While the baseline networks (rigid/soil) already achieve strong test-set accuracy, the real-time objective motivates a systematic exploration of the accuracy–compute trade space.

A. Sweep Setup and Metrics

Rigid and soil architecture sweeps were executed by training multiple candidate MLP topologies and reporting: (i) parameter count, (ii) training time, (iii) best validation loss, (iv) mean RMSE and mean R^2 across outputs, and (v) batch throughput metrics.[1]

B. Accuracy Versus Size (Rigid and Soil)

Table VIII summarizes representative points from the architecture sweeps for both rigid and soil modes. In both regimes, very small networks (e.g., 64×64) underfit, whereas moderately sized networks (e.g., 128×128) provide a strong accuracy jump with modest training cost. The best mid-size accuracy is obtained by deeper configurations such as MLP-128x4 ($128\times128\times128\times128$) and tapered pyramids. While very wide/deep networks (MLP-256x4) further reduce RMSE, they incur substantially higher compute cost with diminishing returns.

C. Why Batch Throughput is Insufficient for Real-Time

Batch throughput (e.g., samples/second) can appear extremely high for neural networks, but the closed-loop tracked vehicle simulation requires *step latency*: the time to compute one inference per simulation tick. The report therefore differentiates throughput-only batch timings from true per-step latency, and uses dedicated per-step benchmarks (Section VIII) to make deployability decisions.

[1]Batch timing represents throughput and does not equal per-step latency.

TABLE VIII

Model Complexity Sweep (Rigid vs. Soil).

Architecture	Params	RMSE mean	R^2 mean
Rigid Mode			
64×64	5,059	1659.89	0.9723
96×96	10,659	1275.02	0.9813
128×128	18,307	1099.27	0.9859
MLP-128x4	51,331	920.86	0.9909
Pyramid (256–128–64)	44,163	916.03	0.9909
Wide (512–256)	137,731	852.65	0.9914
MLP-256x4	200,963	749.65	0.9939
Soil Mode			
64×64	5,123	1029.37	0.9795
96×96	10,755	895.16	0.9847
128×128	18,435	828.52	0.9871
MLP-128x4	51,459	792.50	0.9887
Pyramid (256–128–64)	44,419	701.94	0.9911
Wide (512–256)	138,243	665.55	0.9917
MLP-256x4	201,219	581.23	0.9939

TABLE IX

Manual MEX per-step latency (Rigid vs. Soil).

Arch.	Params	Latency (s)	Speed-up
Rigid Mode			
64×64	5,059	3.64×10^{-6}	$6.46\times$
128×128	18,307	6.28×10^{-6}	$3.77\times$
MLP-128x4	51,331	1.04×10^{-5}	$2.29\times$
MLP-256x4	200,963	2.76×10^{-5}	$0.86\times$
Soil Mode			
64×64	5,123	3.51×10^{-6}	$7.10\times$
128×128	18,435	6.11×10^{-6}	$4.02\times$
MLP-128x4	51,459	1.17×10^{-5}	$2.07\times$
MLP-256x4	201,219	5.85×10^{-5}	$0.41\times$

D. Latency-Aware Operating Points via Manual MEX Inference

To connect the architecture sweep to real-time feasibility, each trained rigid model was exported to a lightweight manual MEX implementation (no Deep Learning Toolbox runtime). The resulting per-step latency and speed-up relative to the physics baseline establish practical operating points (Table IX).

The latency sweep highlights three usable regimes:

- **Latency-first:** 96×96 achieves $\sim 6.1\times$ speed-up but with reduced accuracy.
- **Balanced:** 128×128 improves accuracy substantially while remaining $\sim 3.8\times$ faster than physics.
- **Accuracy-first (still faster than physics):** MLP-128x4 provides the strongest accuracy under a speed-up constraint ($\sim 2.3\times$ faster).

VIII. Runtime Benchmark and Deployment Discussion

Beyond predictive accuracy, the primary motivation of this project is *real-time feasibility*: replacing a cell-integrated terramechanics block with a surrogate that reduces per-step compute while maintaining acceptable dynamic fidelity.

TABLE X

RIGID RUNTIME BENCHMARK (OFFLINE, CPU).

Benchmark	Per-step time (s)	Notes
physics_loop_cpu	2.35×10^{-5}	baseline terramech
nn_step_cpu	4.32×10^{-4}	`dlnetwork` step latency
nn_batch_cpu	2.47×10^{-6}	`dlnetwork` throughput
nn_step_series_cpu	1.68×10^{-3}	SeriesNetwork step (slow)
nn_batch_series_cpu	1.07×10^{-5}	SeriesNetwork throughput
nn_step_mex	9.98×10^{-6}	manual MEX step latency
nn_batch_mex	1.97×10^{-6}	manual MEX throughput

A. Benchmark Methodology

Runtime was measured as per-step wall-clock time for (i) the physics terramechanics function and (ii) several neural inference paths. The benchmarks explicitly distinguish:

1) **Step latency:** one inference per simulation tick (relevant for real-time control/simulation).
2) **Batch throughput:** many samples per call (useful offline, but not directly deployable in closed-loop).

B. Baseline Results (rigid)

Table X summarizes the rigid-mode benchmark. The physics baseline achieves 2.35×10^{-5} s per step. Direct `dlnetwork` step inference is substantially slower due to runtime overhead, despite strong batch throughput. In contrast, the manual MEX implementation achieves 9.98×10^{-6} s per step, i.e., approximately $2.35\times$ faster than the physics baseline on the same machine and benchmark harness.

C. Implications for Real-Time Simulation

The benchmark highlights a decisive bottleneck for real-time deployment: the fixed overhead of standard deep learning frameworks. Our results show that the MATLAB Deep Learning Toolbox runtime imposes an overhead of $\approx 4.3 \times 10^{-4}$ s per call regardless of network size, which is an order of magnitude slower than the physics baseline (2.35×10^{-5} s). Consequently, the "Manual MEX" approach—a stripped-down C++ implementation of the forward pass without object-oriented overhead or safety checks—is not merely an optimization; it is a prerequisite for achieving a net speed-up.

For integration into high-fidelity simulators (e.g., Unreal Engine or Simulink Real-Time), these findings dictate a strict deployment pattern:

- **Bypass Managed Runtimes:** The trained network must be exported to a bare-metal C/C++ inference path (emulating our manual MEX results) to eliminate interpretation overhead.
- **Latency-Driven Selection:** Model selection must be governed by single-step latency at the target tick rate (e.g., 1 kHz), rendering batch throughput metrics irrelevant for closed-loop stability.
- **Compute Budgeting:** The 128×128 architecture offers a balanced operating point ($3.8\times$ speed-up), while the deeper MLP-128x4 variant provides maximum accuracy for budgets that can tolerate a slightly lower margin ($2.3\times$ speed-up).

D. Summary

On the tested CPU platform, the best deployable configuration (manual MEX inference) achieves a per-step speed-up while maintaining high prediction accuracy. This supports the project hypothesis that a black-box NN surrogate can replace the cell-integrated terramechanics block in time-critical simulation loops, provided that deployment is engineered for low per-step overhead.

IX. CONCLUSION

This work presents a deployable surrogate modeling pipeline for tracked vehicle terramechanics with strict data integrity, run-level splitting to prevent leakage, and train-only normalization for consistent deployment. The resulting rigid and soil MLP surrogates achieve high test precision across all outputs, with the yaw moment remaining the most sensitive channel. Generalization stress tests identify PRBS maneuvers and low-traction terrain regimes as the primary failure modes. A manual MEX deployment path reduces per-step latency below the physics baseline, establishing real-time feasibility. Crucially, while validated in Simulink, this C++ inference schema is inherently portable, enabling the direct transfer of these high-fidelity surrogates into Unreal Engine or Chaos Vehicle workflows. Future work will focus on adding minimal temporal context to improve transient M_z behavior, expanding stress tests to closed-loop control scenarios, and integrating OOD detection with safe physics fallback.

REFERENCES

[1] L. Dimauro, S. Venturini, A. Tota, *et al.*, "High-speed tracked vehicle model order reduction for static and dynamic simulations," *Defence Technology*, 2024.

[2] R. He, C. Sandu, A. K. Khan, *et al.*, "Review of terramechanics models and their applicability to real-time applications," *Journal of Terramechanics*, vol. 81, pp. 3–22, 2019.

[3] C. Hua, C. Jiang, R. Niu, *et al.*, "Double neural networks enhanced global mobility prediction model for unmanned ground vehicles in off-road environments," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 5, pp. 7547–7560, 2024.

[4] J. Miao, R. Yan, B. Zhang, *et al.*, "Residual learning towards high-fidelity vehicle dynamics modeling with transformer," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 7404–7411, 2025.

[5] J. Y. Wong and C. F. Chiang, "A general theory for skid steering of tracked vehicles on firm ground," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 215, no. 3, pp. 343–355, 2001.

[6] M. N. Özdemir, V. Kılıç, and Y. S. Ünlüsoy, "A new contact slip model for tracked vehicle transient dynamics on hard ground," *Journal of Terramechanics*, vol. 73, pp. 3–23, 2017.

[7] Z. Janosi and B. Hanamoto, "The analytical determination of drawbar pull as a function of slip for tracked vehicles in sand," in *Proceedings of the 1st International Conference on Terrain-Vehicle Systems*, Turin, Italy, 1961, pp. 707–736.

[8] J. Y. Wong, *Theory of Ground Vehicles*, 1st ed. New York, NY, USA: John Wiley & Sons, 1978.

[9] G. Gat, Y. Franco, and I. Shmulevich, "Fast dynamic modeling for off-road track vehicles," *Journal of Terramechanics*, vol. 92, pp. 1–12, 2020.

[10] J. Lederer, "Risk bounds for robust deep learning," *arXiv preprint arXiv:2009.06202*, 2020.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.