

Hybrid Deep Learning and Newton Refinement: A Baseline-Aware Correction Framework for Nonlinear Pipe-Flow Equations

Seokjun Jeong

Daegu University

Yoosoo Oh

`yoosoo.oh@daegu.ac.kr`

Daegu University

Article

Keywords:

Posted Date: October 21st, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11211693/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Hybrid Deep Learning and Newton Refinement: A Baseline-Aware Correction Framework for Nonlinear Pipe-Flow Equations

SeokJun Jeong¹ and Yoosoo Oh^{1*}

¹Department of Computer and Information Engineering, Daegu
University, Gyeongsan, 38453, Republic of Korea.

*Corresponding author(s). E-mail(s): yoosoo.oh@daegu.ac.kr;
Contributing authors: wjdtjrwns1109@gmail.com;

Abstract

This study presents a hybrid deep-learning and Newton-refinement framework for nonlinear pipe-flow equations governed by Colebrook–White residuals and hydraulic coupling constraints. The neural module predicts a normalized correction to a physically motivated baseline initializer, and Newton refinement remains the final high-precision solver. In two-branch coupled experiments, learned correction reduces the mean Newton iteration count from 5.9251 to approximately 2.61–2.73 while preserving final errors on the order of 10^{-9} . Ablations show that direct solution regression is less stable than baseline-aware correction, raw-ratio flow correction weakens strict post-Newton convergence, and Taylor-feature benefits are distribution-dependent. The coupled system is the primary application, while the scalar Colebrook–White equation provides a controlled validation setting. Multilayer perceptron, long short-term memory, gated recurrent unit, and Transformer backbones are compared using Taylor-coefficient features, global physical parameters, and the baseline state. The flow split is corrected in logit space to preserve its admissible range, and branch-wise friction-related variables are corrected additively. Runtime profiling is reported on the scalar validation benchmark. These results support learned correction as a reusable warm-start mechanism for repeated nonlinear solves, with validation currently limited to synthetic scalar and two-branch pipe-flow problems.

Keywords: Colebrook–White Equation, Nonlinear Pipe-Flow Equations, Newton Refinement, Taylor Coefficients, Baseline-Aware Learning

MSC Classification: 65H10 , 65Y05 , 68T07 , 68T05 , 76M99

1 Introduction

Numerical solution of nonlinear transcendental equations and nonlinear systems is a fundamental task in computational mathematics, fluid mechanics, and control engineering. The Colebrook equation is a representative empirical relation for evaluating the Darcy friction factor in turbulent pipe flow and has been widely used in pipe-network design and hydraulic-system analysis. Because the Colebrook equation is implicit with respect to the unknown friction-related variable, a direct analytic solution is generally unavailable. Therefore, classical iterative methods such as Newton-Raphson iteration, fixed-point iteration, and bisection-type methods have traditionally been used to solve the equation.

Newton’s method provides high precision and locally quadratic convergence. However, its convergence speed, iteration count, and robustness depend heavily on the quality of the initial point. A poor initial value can lead to slow convergence, excessive iteration, or even divergence. These issues are especially important in real-time control systems, large-scale pipe-network simulations, and repeated parameter sweeps where the same type of nonlinear equation must be solved many times. The importance of initialization is also repeatedly emphasized in warm-started optimization and model-predictive-control studies.

Recent learning-based numerical methods have been studied in forms such as physics-informed neural networks, neural ordinary differential equations, differentiable optimization layers, and surrogate models. Many such approaches attempt to replace classical numerical solvers or directly approximate the target solution. In practical engineering problems, however, strong heuristic or analytical initializers often already exist. In such cases, fully replacing a reliable baseline may be less effective than learning a small correction to improve it.

The main focus of this paper is the coupled pipe-flow warm-start problem. The one-dimensional Colebrook–White equation is retained as a scalar validation setting because it has a transparent monotone structure and allows controlled tests of Taylor-feature correction learning. The coupled two-branch system is the primary target because it combines branch-wise implicit Colebrook relations with a flow-distribution constraint, and therefore better reflects the initialization difficulty encountered in repeated hydraulic network simulations.

This study addresses the following questions. First, can a learned correction reduce the Newton refinement cost for coupled pipe-flow systems when a physically meaningful nonlearned initializer is already available? Second, does logit-space correction of the flow ratio, combined with additive correction of branch-wise Colebrook variables, produce a stable warm start for the coupled system? Third, how do MLP, LSTM, GRU, and Transformer correction modules compare when the input contains ordered Taylor-coefficient features and global pipe-flow parameters?

The main contributions of this paper are as follows.

- A coupled pipe-flow warm-start acceleration framework is proposed in which a heuristic or explicit pipe-flow initializer is improved by a learned correction model and then refined by Newton iteration.

- A stable correction parameterization is introduced for two-branch parallel pipe flow: the flow split is corrected in logit space and branch-specific Colebrook variables are corrected additively.
- Taylor-expansion-based residual features are evaluated together with no-Taylor ablations to quantify how much local residual-shape information contributes beyond global pipe-flow parameters and the baseline state.
- MLP, LSTM, GRU, and Transformer backbones are compared under the same baseline-aware correction formulation. The scalar degree-sweep results show that the MLP provides the lowest direct RMSE, whereas sequence-aware models provide stronger Newton-iteration reduction; the coupled pipe-flow results further support sequence-aware processing for Taylor-coefficient inputs.
- Component ablations test direct regression, removal of Taylor features, and removal of logit-space flow-ratio correction, clarifying which design choices most strongly affect Newton-friendly warm starts.
- Diagnostic tables, figures, and inference-time measurements quantify direct residual reduction, Newton-iteration savings, and per-sample runtime, making explicit that the learned component acts as a practical warm-start accelerator rather than a solver replacement.
- A detailed reproducibility protocol is reported, including train/validation/test sizes, random-seed policy, hyperparameter-selection procedure, hardware environment, timing protocol, and leakage-control measures for the synthetic data distribution.

2 Related Work

Numerical methods for nonlinear equations and systems are commonly categorized into bracketing methods, fixed-point iterations, and Newton-type methods. Among them, the Newton-Raphson method is widely used because of its fast local convergence. However, the method is sensitive to initialization, and this sensitivity becomes more pronounced for coupled pipe-flow systems where Jacobian conditioning, flow-distribution constraints, and basin-of-attraction structure affect convergence.

For the Colebrook–White equation, a large body of work has focused on explicit or semi-explicit approximations that avoid repeated iterative solution. Haaland proposed one of the most widely used explicit friction-factor formulas for turbulent pipe flow, offering a simple engineering approximation to the implicit Colebrook relation [1]. Swamee and Jain derived explicit equations for pipe-flow problems, and their formula has been widely adopted in hydraulic engineering practice [2]. More accurate multi-logarithmic approximations, such as the Serghides formula, were later introduced to reduce the error relative to the implicit Colebrook solution [3]. Sonnad and Goudar further developed mathematically reformulated alternatives and explicit representations for turbulent-flow friction-factor calculation [4, 5]. In addition to individual approximations, several comparative and review studies have evaluated the accuracy and computational efficiency of explicit Colebrook approximations, including the review by Brkić and comparative studies by Winning and Coole and Niazkar [6–8]. Recent work has also explored symbolic-regression and artificial-intelligence-based approximations of the Colebrook relation [8, 9].

These studies demonstrate that explicit Colebrook approximations are strong engineering tools and should be treated as meaningful nonlearned baselines. However, their role is different from that of the present framework. Explicit formulas provide fixed closed-form estimates, whereas the proposed method learns a data-dependent correction relative to a baseline initializer and then applies Newton refinement to recover high-precision solutions. In this sense, our method is not intended to replace explicit Colebrook approximations. Instead, explicit approximations can be incorporated as strong baseline initializers, and the learned correction module can further improve their suitability as Newton warm-starts, especially in repeated parameter-sweep settings and coupled pipe-flow systems where initialization quality directly affects refinement efficiency.

Learning-based approaches have also been applied to numerical and engineering problems. Examples include genetic-programming-based approximations, physics-informed neural networks, neural ordinary differential equations, warm-started optimization, and differentiable optimization layers. Most existing approaches focus on direct approximation, explicit surrogate formulation, or solver replacement. In contrast, the present study focuses on a baseline-aware correction strategy. Instead of predicting the solution directly, the neural model predicts the residual correction relative to a heuristic or explicit baseline. This formulation is expected to reduce target scale, improve training stability, and produce a Newton-friendly initial value.

3 Problem Formulation and Proposed Framework

The proposed framework consists of five stages: sample generation for the one-dimensional Colebrook–White equation or for a coupled pipe-flow system, Taylor-expansion-based input construction, baseline initialization, learned correction prediction, and Newton refinement. Figure 1 summarizes the full pipeline. The neural correction module does not replace the original nonlinear solver. Instead, it moves the baseline estimate to a refinement-friendly region and then allows Newton iteration to recover a high-precision solution.

Throughout this paper, scalar Colebrook variables are denoted by x , with λ the Darcy friction factor, Re the Reynolds number, and ϵ/D the relative roughness. The parameters a and b denote the transformed scalar coefficients $a = (\epsilon/D)/3.7$ and $b = 2.51/Re$. For the coupled system, $z = [Q_1, x_1, x_2]^T$ is the unknown vector, Q_T is total flow, $Q_2 = Q_T - Q_1$, $r = Q_1/Q_T$, and θ denotes the physical parameter set $(D_1, D_2, \epsilon_1, \epsilon_2, L_1, L_2, \rho, \mu, g)$. Baseline and corrected initializers are written as z_0 and $\hat{z}_0$, respectively.

3.1 One-Dimensional Colebrook–White Equation

The standard Colebrook–White relation for turbulent pipe flow can be written in terms of the Darcy friction factor λ . In this study, we use the common transformation:

$$x = \frac{1}{\sqrt{\lambda}}$$

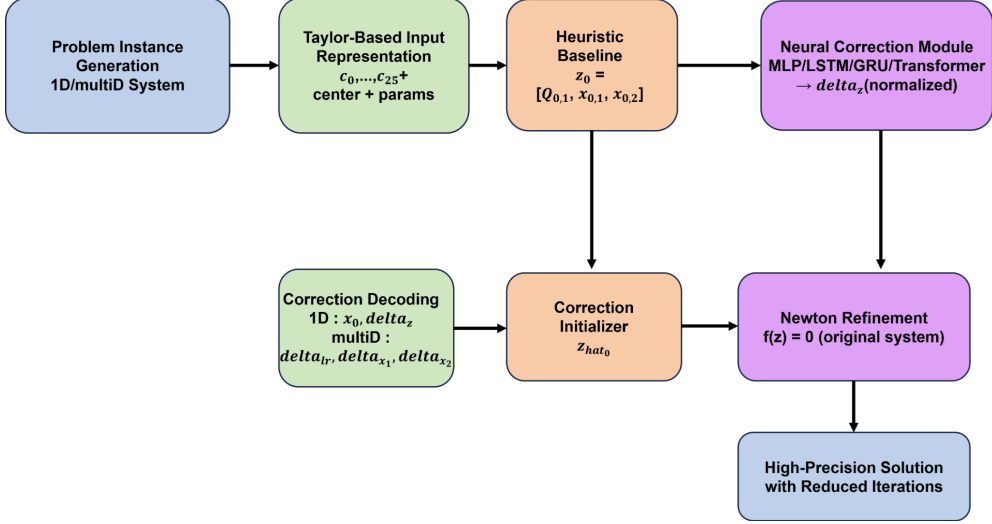


Fig. 1 Revised framework with baseline-aware learned correction. The pipeline combines Taylor-based input features, a baseline initializer, neural correction decoding, and Newton refinement.

This gives the following scalar residual:

$$f(x; Re, \epsilon/D) = x + 2 \log_{10} \left(\frac{\epsilon/D}{3.7} + \frac{2.51x}{Re} \right) = 0$$

This form is equivalent to the following parameterized expression:

$$f(x; a, b) = x + 2 \log_{10}(a + bx) = 0$$

where $a = (\epsilon/D)/3.7$ and $b = 2.51/Re$. The unknown x is the friction-related state used for Newton refinement.

Let x_{base} denote a baseline initializer. In the one-dimensional comparison, we first report two simple nonlearned reference initializers. The first is a deliberately simple fixed reference, $x_{\text{base}} = 1.0$, applied to all test cases. The second is a scale-based heuristic, $x_{\text{base}} = a/b$, motivated by balancing the two terms a and bx inside the logarithmic argument. These references do not use ground-truth roots or Taylor expansion centers. In addition, this revised version evaluates strong explicit Colebrook initializers derived from Haaland, Swamee–Jain, and Serghides formulas. These explicit formulas provide closed-form friction-factor estimates λ_{exp} , which are converted to initializer values by $x_{\text{base}} = 1/\sqrt{\lambda_{\text{exp}}}$.

A direct learning approach would train a neural network to predict x^* , the root of the Colebrook–White residual. However, this study instead learns a correction relative to x_{base} . The corrected initializer is defined by:

$$\hat{x}_0 = x_{\text{base}} + \Delta x$$

where Δx is predicted by a neural correction model.

3.2 Explicit Colebrook Initializers

To address the existence of strong analytical engineering approximations, three explicit Colebrook initializers are included as additional nonlearned baselines. The Haaland approximation is used as a simple and widely adopted turbulent-flow formula. The Swamee–Jain approximation is included because of its broad use in pipe-flow calculations. The Serghides approximation is included as a more accurate multi-logarithmic explicit estimator. In all cases, the explicit formula first computes a friction-factor estimate λ_{exp} , and the corresponding Newton initializer is constructed as $x_{\text{exp}} = 1/\sqrt{\lambda_{\text{exp}}}$.

These explicit initializers serve two purposes. First, they provide a stricter comparison than the fixed and scale-based references. Second, they allow us to test whether the proposed learned correction can improve a strong engineering baseline rather than only compensating for a weak baseline. Therefore, the experimental section evaluates both explicit baseline plus Newton refinement and proposed correction over explicit baseline plus Newton refinement.

3.3 Taylor-Expansion-Based Input Representation

To encode local information near a reference point, the residual function is expanded around a baseline-related point. Let c be the expansion center and $t = x - c$. The residual can be written as:

$$f(c + t; a, b) \approx \sum_{k=0}^K \gamma_k(a, b, c) t^k \quad (1)$$

where $K = 25$ in the experiments and γ_k denotes the k -th Taylor coefficient. The input vector consists of Taylor coefficients, the expansion center, and auxiliary equation parameters. Because the coefficients preserve an ordered structure, recurrent and attention-based models may exploit them more effectively than a fully connected network that treats the features as an unordered vector.

3.4 Baseline-Aware Correction Target

The central idea is to replace direct solution regression with correction learning. In the one-dimensional case, the normalized correction target is defined as:

$$y_{\text{corr}} = \frac{x^* - x_{\text{base}}}{s_x} \quad (2)$$

where s_x is a scaling factor used for numerical stability. During inference, the neural prediction $\hat{y}_{\text{corr}}$ is decoded as:

$$\Delta x = s_x \hat{y}_{\text{corr}}, \quad \hat{x}_0 = x_{\text{base}} + \Delta x \quad (3)$$

This target is typically easier to learn than the absolute solution because a strong baseline already captures much of the solution structure.

3.5 Coupled Pipe-Flow System with Branch-Wise Colebrook–White Residuals

The framework is extended to a coupled pipe-flow problem rather than to a separate “multidimensional Colebrook equation.” In the two-branch formulation used in the experiments, each branch has its own Colebrook–White residual, and the branches are coupled through flow-distribution and hydraulic-balance constraints. The unknown vector may be written as:

$$z = [Q_1, x_1, x_2]^T$$

where Q_1 is a branch-flow variable and x_1, x_2 are branch-specific friction-related Colebrook state variables. The total flow is denoted by Q_T , and the flow ratio is represented by:

$$r = \frac{Q_1}{Q_T}$$

The coupled residual can be expressed abstractly as:

$$F(z; \theta) = 0$$

where θ collects system parameters such as diameters, roughness values, pipe lengths, fluid density, viscosity, and gravitational constants. This formulation should be understood as a coupled pipe-flow system composed of Colebrook–White branch equations and network constraints, not as a new standalone multidimensional Colebrook equation.

A direct regression target for z^* may suffer from scale imbalance because the flow variable and branch variables have different physical units and numerical ranges. To address this issue, the proposed method uses a correction target in which the flow ratio is corrected in logit space:

$$\ell_r = \log \left(\frac{r}{1-r} \right)$$

and branch-specific variables are corrected additively. If $z_{\text{base}} = [Q_1^0, x_1^0, x_2^0]^T$ is the heuristic baseline initializer, the normalized correction vector is defined as:

$$\Delta z_{\text{norm}} = [\Delta \ell_r, \Delta x_1, \Delta x_2]^T$$

The decoded initial value $\hat{z}_0$ is obtained by combining this correction with the baseline. In this notation, $\Delta \ell_r$ is the correction to the logit-transformed flow ratio and $\Delta x_1, \Delta x_2$ are additive corrections to the two branch Colebrook states. This parameterization reduces target-scale imbalance and produces a numerically stable initializer for the Newton refinement stage. Figure 2 visualizes the correction-target design.

4 Neural Correction Architectures

Four backbone architectures are compared as neural correction modules: MLP, LSTM, GRU, and Transformer. All models receive the same Taylor-based input representation

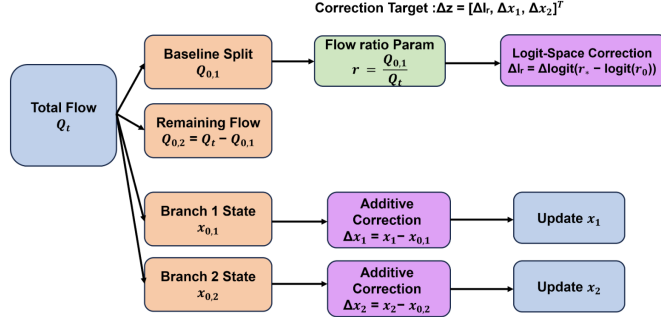


Fig. 2 Correction-target design for the coupled pipe-flow system. The flow split is represented as a ratio and corrected in logit space, while branch-specific Colebrook state variables are corrected additively.

and predict the same normalized correction target. The difference lies in how each architecture processes the coefficient sequence.

The MLP baseline flattens the Taylor coefficients and auxiliary variables into a single vector and passes them through fully connected layers. This architecture is simple and computationally efficient, but it does not explicitly model the ordered structure of the Taylor coefficients. The LSTM and GRU models treat Taylor coefficients as a sequence, allowing recurrent structures to model dependencies among lower- and higher-order terms. The Transformer backbone applies self-attention to the Taylor-coefficient sequence, enabling the model to learn global interactions among all coefficient positions. Figure 3 compares the four backbone options under a shared correction head.

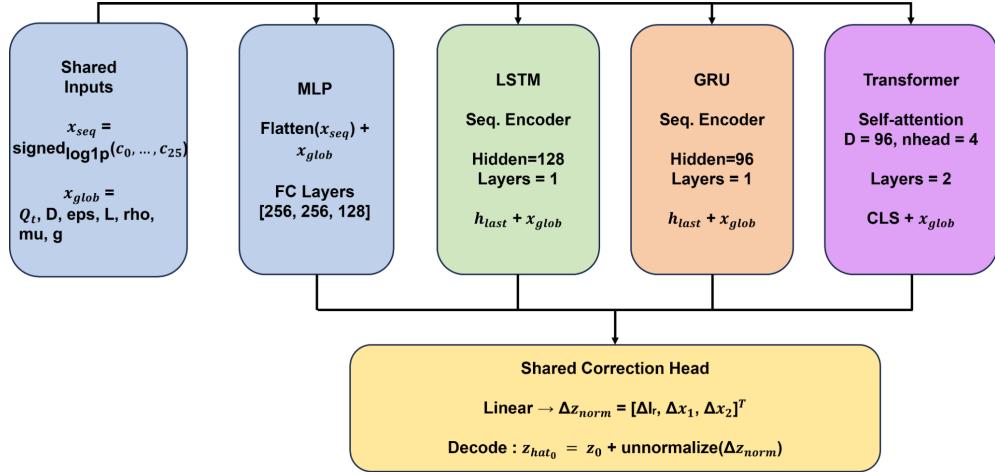


Fig. 3 Backbone options for the neural correction module. MLP, LSTM, GRU, and Transformer models use the same correction target but process the Taylor representation differently.

For all backbones, the shared sequence input is constructed from signed log-transformed Taylor coefficients and expansion-center information. A global feature vector includes system-level quantities such as Q_T , D_1 , D_2 , ϵ_1 , ϵ_2 , L_1 , L_2 , ρ , μ , and g . Each model outputs the normalized correction vector $\Delta z_{\text{norm}} = [\Delta \ell_r, \Delta x_1, \Delta x_2]^T$, which is decoded and combined with the heuristic baseline $z_0 = [Q_1^0, x_1^0, x_2^0]^T$.

4.1 Detailed Model Diagrams and Feature Flow

Figures 4–7 illustrate how the sequence input x_{seq} , the global input x_{glob} , and the heuristic baseline z_0 are combined for each backbone. The shared sequence input consists of signed-log transformed Taylor coefficients and expansion-center information, while the global vector contains physical and geometric parameters such as Q_T , D_1 , D_2 , ϵ_1 , ϵ_2 , L_1 , L_2 , ρ , μ , and g . The numerical hyperparameters shown in the diagrams correspond to the settings used in the repeated experiments, not merely illustrative examples.

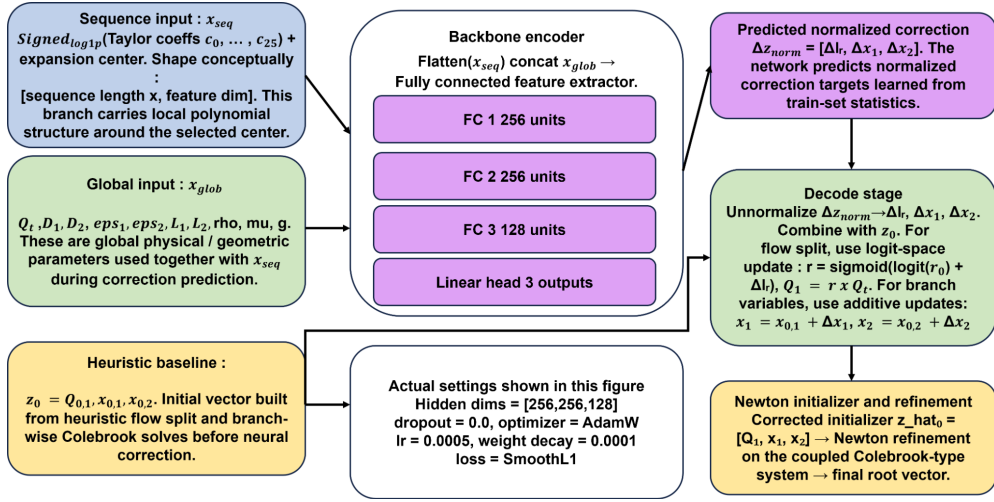


Fig. 4 Detailed MLP correction architecture. Taylor features are flattened and concatenated with the global input before being mapped to the normalized correction vector.

5 Newton Refinement

In the one-dimensional setting, the predicted initializer $\hat{x}_0$ is used as the starting point for Newton’s method applied to $f(x; a, b) = 0$. The update is:

$$x_{k+1} = x_k - \frac{f(x_k; a, b)}{f'(x_k; a, b)} \quad (4)$$

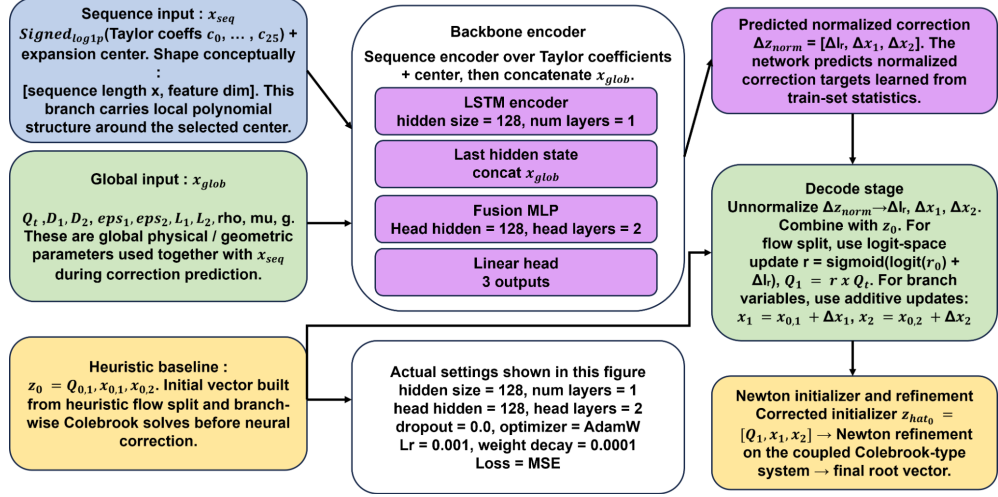


Fig. 5 Detailed LSTM correction architecture. The LSTM encoder processes the Taylor sequence and combines the last hidden state with global features.

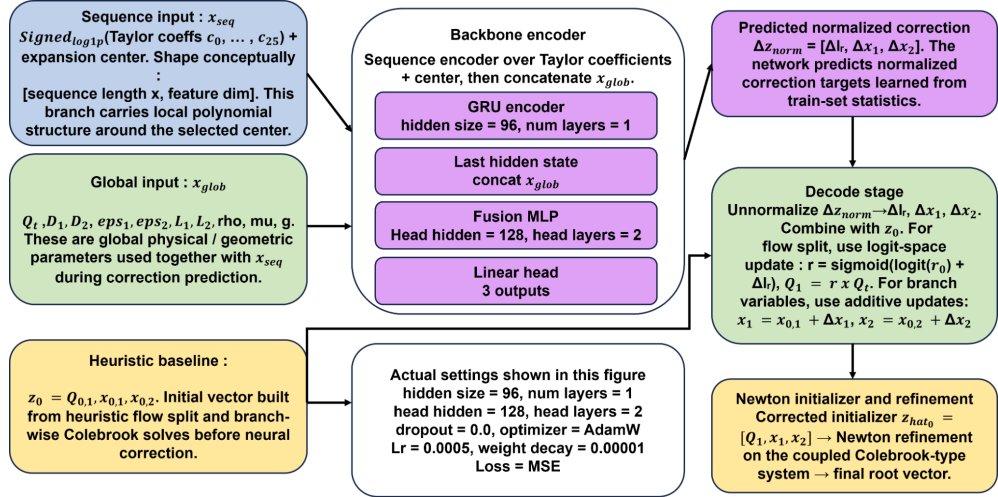


Fig. 6 Detailed GRU correction architecture. The GRU encoder provides a sequence-aware representation with fewer recurrent parameters than the LSTM setting.

In the coupled-system setting, the corrected initial vector $\hat{z}_0$ is used as the starting point for Newton-type refinement:

$$z_{k+1} = z_k - J_F(z_k)^{-1} F(z_k) \quad (5)$$

where $J_F(z_k)$ is the Jacobian matrix of the coupled system. The purpose of the proposed method is not to replace Newton refinement but to provide an initializer that lies closer to the attraction basin of the desired root. In implementation, numerical

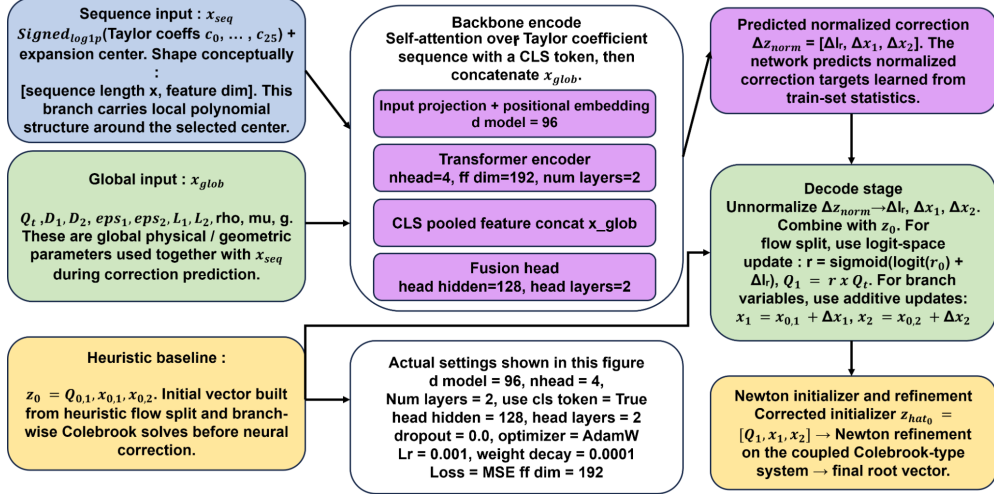


Fig. 7 Detailed Transformer correction architecture. Self-attention is applied to the Taylor-coefficient sequence before a fusion head predicts the normalized correction vector.

safeguards such as domain projection, small-derivative stabilization, step clipping, and half-step correction are used to improve robustness.

6 Numerical Interpretation of the Correction Initializer

The proposed neural component should be interpreted as a correction mechanism for Newton initialization rather than as a stand-alone solver. Let z^* denote the exact solution of the nonlinear system and let z_0 be the baseline initializer. The baseline error is defined as:

$$e_0 = z_0 - z^* \quad (6)$$

The learned correction model predicts a normalized correction Δz_{norm} , which is unnormalized and decoded to obtain a corrected initializer:

$$\hat{z}_0 = z_0 + \Delta z \quad (7)$$

The corresponding corrected error is:

$$\hat{e}_0 = \hat{z}_0 - z^* \quad (8)$$

If the correction is accurate in the sense that $\|\hat{e}_0\| < \|e_0\|$, then the initial point supplied to Newton refinement is closer to the target root. More importantly, the corrected initializer is more likely to lie inside the local attraction basin of the desired solution. Once an initializer is in this basin, Newton-type refinement can recover high-precision accuracy with a small number of iterations.

This interpretation explains why the correction target is defined relative to a baseline rather than as direct solution regression. The network does not need to learn

the full solution map from physical parameters to the root. Instead, it learns the local displacement that moves a reliable but imperfect heuristic estimate toward a refinement-friendly region. This reduces the dynamic range of the target, improves numerical conditioning during training, and preserves the final accuracy of the classical Newton solver. Therefore, the main benefit of the proposed method is not to replace Newton’s method, but to reduce the initial error and consequently reduce refinement cost and failure risk.

6.1 Local error and Newton-basin interpretation

The following elementary estimates clarify the role of the correction model. They are not intended to replace the empirical evaluation, but they make explicit why the learned component is used only as a warm-start module.

Proposition 1 (Scalar uniqueness and local Newton convergence) *For $a > 0$ and $b > 0$, the scalar residual:*

$$f(x; a, b) = x + 2 \log_{10}(a + bx)$$

has exactly one root on the domain $x > -a/b$. Moreover, if an initializer x_0 is sufficiently close to the root x^* , Newton iteration converges locally and quadratically.

Proof On the domain $x > -a/b$,

$$f'(x) = 1 + \frac{2b}{(a + bx) \ln 10} > 0$$

Hence f is strictly increasing. In addition, $f(x) \rightarrow -\infty$ as $x \rightarrow (-a/b)^+$, while $f(x) \rightarrow +\infty$ as $x \rightarrow +\infty$. Therefore a unique root exists. The derivative is nonzero at the root, and f is twice continuously differentiable in a neighborhood of the root. The classical local convergence theorem for Newton’s method then gives local quadratic convergence for sufficiently close initializers. $\square$

Lemma 1 (Correction-error reduction) *Let z^* be a root of a scalar or vector nonlinear system and let z_0 be a baseline initializer. Suppose that the learned decoded correction can be written as:*

$$\Delta z = (z^* - z_0) + \eta$$

where η is the correction prediction error. Then the corrected initializer satisfies:

$$\hat{z}_0 - z^* = \eta$$

Consequently, whenever $\|\eta\| < \|z_0 - z^*\|$, the corrected initializer is closer to the desired root than the original baseline.

Proof Using $\hat{z}_0 = z_0 + \Delta z$, we obtain:

$$\hat{z}_0 - z^* = z_0 + (z^* - z_0) + \eta - z^* = \eta$$

The norm statement follows immediately. $\square$

Proposition 2 (Newton cost reduction under basin entry) *Assume that a nonlinear residual F is continuously differentiable in a neighborhood of a nonsingular root z^* , and that a Newton basin $\mathcal{B}$ exists around z^* . If the baseline z_0 and corrected initializer $\hat{z}_0$ both belong to $\mathcal{B}$, but $\|\hat{z}_0 - z^*\| < \|z_0 - z^*\|$, then the corrected initializer has a smaller initial error and is expected to require fewer refinement steps to reach a fixed residual tolerance. If $z_0 \notin \mathcal{B}$ but $\hat{z}_0 \in \mathcal{B}$, the correction changes a potentially unreliable warm start into a locally convergent one.*

Proof The first claim follows from the local Newton error relation:

$$\|e_{k+1}\| \leq C\|e_k\|^2$$

valid in a sufficiently small neighborhood of a nonsingular root, where $e_k = z_k - z^*$ and $C > 0$ is a local constant. A smaller initial error yields a smaller first Newton error bound and therefore a smaller expected number of iterations for a fixed tolerance. The second claim is a direct consequence of membership in the local convergence basin. $\square$

These statements also explain the normalized target used in this paper. Direct solution regression requires the model to learn the full map from problem parameters to z^* . By contrast, correction learning only needs to learn the residual displacement from z_0 to z^* . The remaining error is precisely the correction prediction error η , which is then handled by Newton refinement.

7 Experimental Setup

Experiments are conducted for both the primary two-branch coupled pipe-flow system and a supporting one-dimensional Colebrook-type validation problem. The coupled experiment is the main evaluation setting because it contains the full warm-start structure considered in this work: branch-wise Colebrook residuals, a flow-distribution unknown, and a hydraulic head-loss balance constraint. The scalar experiment is used as a controlled diagnostic case to verify the baseline-aware correction idea under a monotone equation with a unique root.

The primary coupled-system evaluation compares conductance-based, diameter-proportional, and equal flow-split rules, each combined with branch-wise Haaland, Swamee–Jain, or Serghides explicit Colebrook initializers. These nonlearned initializers are then compared with MLP, LSTM, GRU, and Transformer correction models followed by the same Newton refinement protocol. The main performance indicators are direct initialization error, initial residual quality, Newton iteration count, convergence ratio, and per-sample runtime.

The one-dimensional experiments include the fixed reference $x_0 = 1.0$, the scale heuristic $x_0 = a/b$, and explicit Colebrook initializers based on Haaland, Swamee–Jain, and Serghides formulas. These scalar tests are reported after the coupled results and should be interpreted as validation and stress-test experiments rather than as the primary contribution of the paper.

7.1 Dataset Generation and Reproducibility Protocol

All datasets used in this study were generated from scripts included in the evaluation package. For the one-dimensional scalar validation problem, samples were generated

from:

$$F(x; a, b) = x + 2 \log_{10}(a + bx) = 0, \quad a > 0, \quad b > 0$$

where a and b were sampled from log-uniform distributions. For each accepted sample, a unique root x^* was computed by a bracketing-based root solver, and Taylor coefficients were computed around the selected expansion center. The scalar NPZ files store the parameters a and b , the target root x^* , the Taylor center, the residual, and the coefficient vector. For the scalar validation experiments, 100,000 generated samples were split into 80,000 training samples, 10,000 validation samples, and 10,000 test samples.

The primary coupled-system dataset was generated from a two-branch parallel pipe-flow system. Each problem instance was specified by total flow rate, branch diameters, pipe roughnesses, pipe lengths, density, viscosity, and gravitational acceleration. For each accepted instance, the branch-wise Colebrook residuals and the hydraulic head-loss balance equation were solved to obtain the target vector:

$$z^* = (Q_1^*, x_1^*, x_2^*)$$

Samples failing the residual-quality check were discarded. The final coupled-system NPZ files store Taylor coefficient tensors, expansion centers, targets, physical parameters, equation descriptors, and residuals. In particular, the Taylor coefficient tensor has shape $(N_s, 3, K + 1)$, where N_s denotes the number of samples, the second dimension corresponds to the three residual equations, and K is the Taylor truncation degree. The stored center is $(Q_{1,c}, x_{1,c}, x_{2,c})$, and the stored target is (Q_1^*, x_1^*, x_2^*) . Additional stored fields include Q_T , D_1 , D_2 , ϵ_1 , ϵ_2 , L_1 , L_2 , ρ , μ , g , residual value, expression descriptor, and feature descriptor.

For the coupled-system experiments, each Taylor degree used a fixed split of 20,000 training samples, 4,000 validation samples, and 4,000 test samples. The default random seed was 42; validation and test splits were generated using seed offsets 43 and 44, respectively. The training set was used only for parameter fitting, the validation set was used for hyperparameter and checkpoint selection, and the test set was used only once for the final reported metrics. This separation was used to reduce the risk of data leakage and test-set overfitting.

The tested Taylor truncation degrees were:

$$K \in \{10, 15, 20, 25, 30, 35\}$$

The nonlearned heuristic initializer does not use Taylor coefficients; therefore its direct and Newton-refined performance remains identical across Taylor degrees when the same test parameter set is used. In contrast, the neural correction models use Taylor coefficient features and global physical parameters, so their degree-wise results evaluate the robustness of the learned correction with respect to the truncation order.

7.2 Training and Hyperparameter Selection

Four neural correction backbones were evaluated: MLP, LSTM, GRU, and Transformer. All models were trained in the same baseline-aware correction framework.

Table 1 Reproducibility summary of the main coupled-system experiments.

Item	Setting
Problem	Two-branch parallel pipe-flow system
Unknowns	$z = (Q_1, x_1, x_2)$
Taylor degrees	$K = \{10, 15, 20, 25, 30, 35\}$
Train / validation / test	20,000 / 4,000 / 4,000
Random seeds	42 / 43 / 44
Input features	Taylor coefficients, centers, global pipe parameters
Target	Baseline-aware normalized correction to z^*
Backbones	MLP, LSTM, GRU, Transformer
Optimizer	AdamW
Selection set	Validation split only
Final reporting	Held-out test split only
Newton tolerance	10^{-12}
Maximum Newton iterations	20
Training hardware	NVIDIA H100 NVL GPU
Timing mode	CPU-only per-sample runtime benchmark for neural and nonlearned pipelines

Instead of directly regressing the full solution vector, each model predicts a normalized correction relative to a nonlearned baseline initializer. For the coupled system, the flow split is corrected in logit-ratio space, while the branch-wise Colebrook variables are corrected additively. This parameterization preserves the physical range of the flow split and reduces the scale of the learning target.

Hyperparameters were selected using validation-set performance. The grid-search records store the model type, Taylor degree, best epoch, elapsed training time, direct-stage metrics, Newton-refined metrics, and all hyperparameters used in the trial. The searched and recorded hyperparameters include optimizer, learning rate, weight decay, dropout, hidden dimensions, recurrent hidden size, number of recurrent layers, correction-head size, Transformer dimension, number of attention heads, feed-forward dimension, and CLS-token usage. After validation-based selection, the selected checkpoint was evaluated on the held-out test set. No test-set metric was used for model selection.

The final selected settings are summarized in Table 2. All reported models use the AdamW optimizer and zero dropout. Training was run for up to 5,000 epochs with early stopping patience of 200 epochs. The selected checkpoint is the checkpoint with the best validation-stage record under the same correction target and Newton-evaluation protocol.

Table 2 Final hyperparameter settings used for the neural correction models.

Model	Loss	Learning rate	Weight decay	Main architecture
MLP	SmoothL1	5×10^{-4}	1×10^{-4}	Hidden dimensions [256, 256, 128]
LSTM	MSE	1×10^{-3}	1×10^{-4}	Hidden size 128, 1 layer, correction head 128
GRU	MSE	5×10^{-4}	1×10^{-5}	Hidden size 96, 1 layer, correction head 128
Transformer	MSE	1×10^{-3}	1×10^{-4}	$d_{\text{model}} = 96$, 4 heads, 2 layers, FF dimension 192

7.3 Implementation, Hardware, and Runtime Measurement

All experiments were implemented in Python with PyTorch and NumPy. Model training was conducted on a CUDA-enabled workstation equipped with an NVIDIA H100 NVL GPU. The main software environment used Ubuntu 20.04, PyTorch 2.5.1 with CUDA 12.1, and NVIDIA driver 550.90.07. The implementation package contains the dataset-generation scripts, model definitions, training scripts, evaluation scripts, figure-generation scripts, and result logs used to reproduce the reported tables. The code availability arrangements are described in Section 7.6.

For the reported timing experiments, saved checkpoints were loaded and evaluated on the fixed test split without retraining. No model selection was performed during timing evaluation. The timing values reported in Tables 15, 16, and 17 are CPU-only per-sample measurements from the scalar validation benchmark. For the hybrid neural pipeline, “Forward CPU” isolates the neural forward stage under CPU-only execution, “Forward+Copy” includes input preparation, array/tensor conversion, and memory-copy overhead in the CPU timing pipeline, “Direct total” includes direct-stage correction and evaluation, and “Neural+Newton” measures the complete neural warm-start and Newton-refinement path. For the nonlearned baselines, “Init CPU”, “Newton CPU”, and “Total CPU” are also measured under CPU-only execution on the same scalar test split. This separation is important because the proposed model is a warm-start correction module rather than a standalone solver replacement, and it allows both the neural overhead and the classical CPU baseline cost to be inspected directly under the same CPU-only timing assumption.

Newton refinement was applied with the same residual tolerance and convergence criterion for all models. A run was counted as converged when the residual norm reached the prescribed tolerance within the maximum number of Newton iterations. In the one-dimensional setting, the residual is measured by $|f(x; a, b)|$. In the coupled-system setting, residual quality is measured using the norm of the coupled-system residual vector. The evaluation metrics include mean absolute error (MAE), root mean square error (RMSE), coefficient of determination (R^2), valid ratio, residual mean/median/p90, Newton iteration mean/median/p90, convergence ratio, and inference time.

7.4 Distributional Scope and Leakage Control

Because the datasets are synthetically generated, the reported results should be interpreted within the sampled parameter ranges of the generation scripts. The goal of the experiments is not to claim universal generalization to all possible pipe-flow networks, but to evaluate whether a baseline-aware learned correction can accelerate Newton refinement under a controlled and reproducible distribution of nonlinear pipe-flow instances. To reduce leakage risk, the target solution is used only as the supervised label and is not included in the input features. The nonlearned baselines do not use the target solution or the Taylor expansion center as an initializer. Training, validation, and test splits are generated separately with fixed seed offsets, and the test split is reserved for final reporting only. Future work will extend the distribution to larger pipe networks, broader parameter ranges, and real measurement-based hydraulic datasets.

7.5 Data availability

The datasets used in this study were synthetically generated. The dataset-generation scripts are publicly available in the GitHub repository, <https://github.com/jseongnam/Residual.Verified.Neural.Warmstart>.

7.6 Code availability

The code for dataset generation, model training, and evaluation is available in the GitHub repository, <https://github.com/jseongnam/Residual.Verified.Neural.Warmstart>.

8 Results

8.1 Two-Branch Parallel Pipe-Flow System: Nonlearned Initializers

The coupled-system experiments are now described more precisely as a two-branch parallel pipe-flow system composed of branch-wise Colebrook–White residuals and a head-loss balance constraint. The unknown vector is $z = [Q_1, x_1, x_2]^T$, where Q_1 is the branch-1 flow rate and x_1, x_2 are the branch-wise Colebrook state variables. The total flow Q_T is fixed, so $Q_2 = Q_T - Q_1$. The system therefore consists of two branch-wise Colebrook residual equations and one coupling equation enforcing equal head loss between the two parallel branches.

Table 3 reports direct initialization performance for nonlearned two-branch initializers. Three flow-split rules are evaluated: conductance-based split, diameter-proportional split, and equal split. For each split, branch-wise x values are initialized with Haaland, Swamee–Jain, or Serghides formulas. The conductance-based flow split is clearly the strongest nonlearned strategy. In particular, conductance split with Haaland and Swamee–Jain branch initializers obtains the lowest direct RMSE among the evaluated nonlearned initializers, whereas equal flow split produces much larger residuals because it ignores hydraulic asymmetry between branches.

Table 3 Direct initialization performance of two-branch parallel pipe-flow nonlearned initializers. Cond., Diam., and Equal denote conductance-based, diameter-proportional, and equal flow-split initializers, respectively.

Initializer	MAE	RMSE	R^2	Residual Mean	Residual p90
Cond.-Haaland	0.0303	0.0463	0.9999	0.5122	0.6500
Cond.-Swamee–Jain	0.0414	0.0470	0.9999	0.5210	0.6472
Cond.-Serghides	0.0310	0.0517	0.9999	0.5005	0.6448
Diam.-Swamee–Jain	0.1706	0.2861	0.9961	13.8055	26.7891
Diam.-Serghides	0.1706	0.2945	0.9959	13.7545	26.6895
Diam.-Haaland	0.1711	0.2979	0.9958	13.7789	26.6710
Equal-Swamee–Jain	0.2288	0.3652	0.9937	111.5320	128.0320
Equal-Serghides	0.2236	0.3678	0.9936	111.3620	127.7424
Equal-Haaland	0.2207	0.3681	0.9936	111.5681	127.9661

Table 4 reports the corresponding Newton-refinement results. Conductance split with Haaland or Swamee–Jain branch initializers converges for all test cases and reaches final RMSE values on the order of 10^{-9} , with mean iteration counts of 3.3129 and 3.3738, respectively. These results establish a stronger nonlearned baseline than the earlier generic heuristic. Nevertheless, the learned correction results in Table 6 remain more efficient, requiring approximately 2.61–2.73 Newton iterations while preserving the same high-precision final accuracy. Thus, even when the nonlearned initializer is strengthened using pipe-flow structure and explicit Colebrook–White approximations, learned correction still improves refinement efficiency.

Table 4 Newton refinement performance of two-branch parallel pipe-flow nonlearned initializers.

Initializer	Final RMSE	Iter. Mean	Iter. p90	Conv. Ratio	Residual Mean
Cond.-Haaland + Newton	5.3708×10^{-9}	3.3129	4.0000	1.0000	5.4146×10^{-12}
Cond.-Swamee–Jain + Newton	5.3718×10^{-9}	3.3738	4.0000	1.0000	4.7727×10^{-12}
Cond.-Serghides + Newton	0.0116	4.2617	5.0000	0.9882	2.1677×10^{-6}
Diam.-Swamee–Jain + Newton	0.0135	5.3132	8.0000	0.9982	5.9904×10^{-6}
Diam.-Haaland + Newton	0.0276	5.4045	8.0000	0.9950	1.5893×10^{-5}
Equal-Haaland + Newton	0.0221	5.9969	9.0000	0.9977	9.3630×10^{-6}
Equal-Swamee–Jain + Newton	0.0280	6.0581	9.0000	0.9963	1.2822×10^{-5}
Diam.-Serghides + Newton	0.0468	6.2027	8.0000	0.9808	3.0686×10^{-5}
Equal-Serghides + Newton	0.0431	6.7335	10.0000	0.9831	2.4874×10^{-5}

8.2 Two-Branch Parallel Pipe-Flow System: Learned Correction Results

In the two-branch parallel pipe-flow system, the neural models predict corrections relative to a nonlearned baseline instead of directly regressing the full solution vector. The experimental results show that the convergence ratio reaches almost 1.0 for most combinations, while the main differences among models appear in direct-stage RMSE, R^2 , residual quality, and Newton iteration count. This indicates that in a coupled pipe-flow setting with a strong baseline, the main contribution of correction learning is better direct initialization and fewer refinement iterations rather than improved final precision alone.

Table 5 Direct initialization performance of learned correction models in the two-branch parallel pipe-flow system.

Model	MAE	RMSE	R^2	Residual Mean	Residual p90	Max Abs. Error
Heuristic baseline	0.097100	0.210257	0.883475	103.266274	126.510988	3.518491
MLP	0.000842	0.003702	0.999961	0.032826	0.031617	1.146684
LSTM	0.000889	0.001669	0.999993	0.017140	0.019273	0.062834
GRU	0.000958	0.001771	0.999992	0.019582	0.021716	0.061975
Transformer	0.000885	0.001668	0.999993	0.018255	0.019712	0.060114

Table 6 shows the results after Newton refinement. The final RMSE values are nearly identical across learned models, while the iteration counts are lower than those

of the strengthened conductance-based nonlearned initializers in Table 4. This confirms that the proposed learned correction model mainly improves the quality of the initial point and reduces the cost of refinement.

Table 6 Post-refinement performance of learned correction models in the two-branch parallel pipe-flow system.

Model	Final RMSE	Iter. Mean	Iter. Median	Iter. p90	Converged Ratio
Heuristic + Newton	3.6814×10^{-9}	5.9251	6.00	9.00	0.9999
MLP + Newton	3.4548×10^{-9}	2.7281	3.00	3.00	1.0000
LSTM + Newton	3.4548×10^{-9}	2.6119	2.94	3.00	1.0000
GRU + Newton	3.4548×10^{-9}	2.6395	2.98	3.00	1.0000
Transformer + Newton	3.4548×10^{-9}	2.6150	2.92	3.00	1.0000

In contrast to the one-dimensional case, the coupled pipe-flow results suggest that the Transformer and LSTM models provide the best direct initialization performance. This supports the interpretation that sequence-aware backbones are more advantageous when the input contains branch-wise Taylor-coefficient sequences and coupled global features. Thus, the architecture ranking is problem-dependent: MLP is strongest for the scalar correction task, while Transformer and LSTM are strongest for the coupled pipe-flow correction task.

8.3 Coupled Warm-Start Gain Analysis

The additional diagnostics in this subsection make explicit the connection between direct initialization quality and Newton refinement cost. The existing performance tables are real numerical tables; however, the previous narrative in this section referred to initialization-error reduction, residual reduction, and refinement-cost reduction without giving a dedicated summary table or figure for those relationships. Tables 7 and 8, together with Figures 8 and 9, provide this explicit support.

Table 7 Coupled pipe-flow Newton-iteration savings relative to the heuristic baseline. The values are computed from Tables 4 and 6.

Initializer	Iter. Mean	Saving vs. heuristic	Converged Ratio
Heuristic + Newton	5.9251	0.00%	0.9999
Cond.-Haaland + Newton	3.3129	44.09%	1.0000
Cond.-Swamee-Jain + Newton	3.3738	43.06%	1.0000
MLP + Newton	2.7281	53.96%	1.0000
LSTM + Newton	2.6119	55.92%	1.0000
GRU + Newton	2.6395	55.45%	1.0000
Transformer + Newton	2.6150	55.87%	1.0000

Table 8 Direct residual reduction in the coupled pipe-flow setting. The values summarize the residual means reported in Tables 3 and 5.

Initializer	Direct residual mean	Reduction vs. heuristic
Heuristic baseline	103.266274	0.00%
Cond.-Haaland	0.512200	99.50%
Cond.-Swamee–Jain	0.521000	99.50%
Cond.-Serghides	0.500500	99.52%
MLP	0.032826	99.97%
LSTM	0.017140	99.98%
GRU	0.019582	99.98%
Transformer	0.018255	99.98%

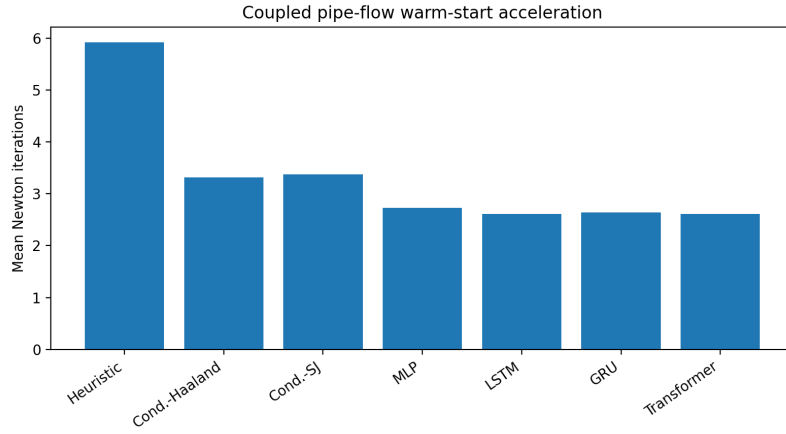


Fig. 8 Mean Newton iteration counts for coupled pipe-flow warm starts. Learned correction models reduce refinement cost below both the heuristic baseline and the strongest conductance-based explicit initializers.

These diagnostics support the main interpretation of the paper. Conductance-based explicit initializers already provide strong nonlearned warm starts, but the learned correction stage further reduces the initial residual and lowers the average Newton iteration count. Therefore, the primary contribution is a coupled-system warm-start accelerator that preserves Newton refinement as the final high-accuracy solver.

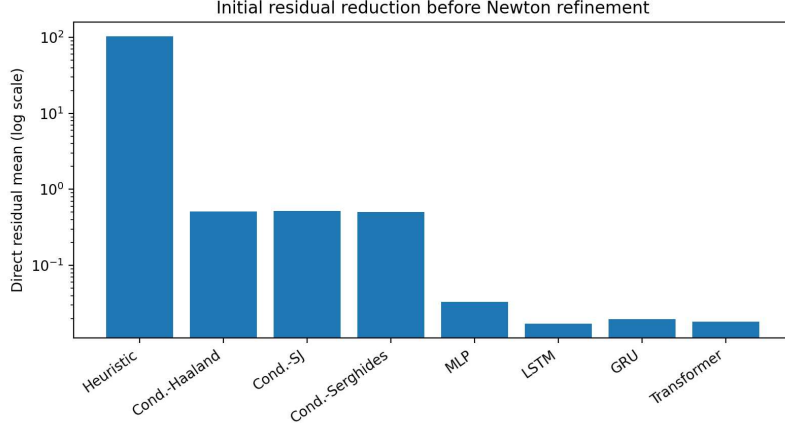


Fig. 9 Direct residual means before Newton refinement in the coupled pipe-flow setting. The logarithmic scale highlights the residual reduction obtained by the learned correction stage.

8.4 One-Dimensional Taylor-Degree and Backbone Sensitivity

The one-dimensional Colebrook–White setting is used as a controlled scalar validation problem. In the scalar degree-sweep tables, the symbol N denotes the Taylor truncation degree and corresponds to K in the general formulation. In contrast to the coupled pipe-flow results, which evaluate the full branch-wise hydraulic system, the scalar experiment isolates the effect of Taylor-degree choice and backbone architecture on baseline-aware correction. The nonlearned heuristic baseline is independent of the Taylor truncation degree, because it does not use Taylor coefficients as input. Therefore, its reference values are reported once: direct RMSE is 2.124×10^{-1} , Newton-refined RMSE is 3.101×10^{-9} , mean Newton iteration count is 5.914, and convergence ratio is 0.99990. The degree-wise tables below report only neural results and, where applicable, iteration reduction relative to this fixed heuristic reference.

Tables 9 and 10 summarize the degree-wise best models under two different criteria. Table 9 selects the model with the lowest neural direct RMSE for each Taylor degree. The MLP achieves the lowest direct RMSE in five of the six tested degrees, including the overall best direct value of 3.550×10^{-4} at $N = 25$. This indicates that a compact fully connected model is highly effective for minimizing direct correction error in the scalar validation problem.

However, the post-refinement criterion gives a different architecture ranking. Although all neural warm starts reach the same Newton-refined RMSE of 3.101×10^{-9} , the number of Newton updates varies across models. Table 10 selects the model with the lowest mean Newton iteration count for each Taylor degree. Sequence-aware models dominate this criterion: GRU, LSTM, and Transformer provide the best iteration count depending on the Taylor degree. The best overall iteration result is obtained by the LSTM at $N = 25$, which reduces the mean Newton iteration count from the fixed heuristic reference of 5.914 to 2.111, corresponding to a 64.30% reduction.

Table 9 Best direct neural initializer for each Taylor degree in the scalar Colebrook–White validation problem. The fixed nonlearned heuristic reference is independent of the Taylor degree and is therefore not repeated in the table.

Degree	Best model	Neural RMSE	Neural R^2	Neural iter.	RMSE red. vs. fixed ref.
10	MLP	4.864×10^{-4}	0.99999938	2.340	99.77%
15	GRU	4.837×10^{-4}	0.99999939	2.181	99.77%
20	MLP	3.755×10^{-4}	0.99999963	2.324	99.82%
25	MLP	3.550×10^{-4}	0.99999967	2.275	99.83%
30	MLP	4.228×10^{-4}	0.99999953	2.360	99.80%
35	MLP	4.664×10^{-4}	0.99999943	2.574	99.78%

Table 10 Best Newton-iteration warm start for each Taylor degree in the scalar Colebrook–White validation problem. Iteration reduction is computed relative to the fixed heuristic baseline with mean iteration count 5.914.

Degree	Best model	Neural RMSE	Newton RMSE	Neural iter.	Iter. red. vs. fixed ref.
10	GRU	5.857×10^{-4}	3.101×10^{-9}	2.142	63.79%
15	Transformer	7.611×10^{-4}	3.101×10^{-9}	2.173	63.25%
20	Transformer	7.370×10^{-4}	3.101×10^{-9}	2.139	63.84%
25	LSTM	6.323×10^{-4}	3.101×10^{-9}	2.111	64.30%
30	LSTM	5.854×10^{-4}	3.101×10^{-9}	2.140	63.81%
35	Transformer	6.309×10^{-4}	3.101×10^{-9}	2.188	62.99%

For completeness, Table 11 reports model-wise averages over $N = 10, 15, 20, 25, 30, 35$. The MLP has the lowest mean direct RMSE, whereas the Transformer has the lowest mean Newton iteration count and the largest mean iteration reduction. This confirms that direct prediction accuracy and Newton refinement efficiency are related but not identical. A lower direct RMSE does not always imply the smallest number of Newton updates, because refinement cost also depends on the local residual geometry, derivative behavior, and the location of the initializer relative to the Newton attraction region.

Table 11 Model-wise average performance over Taylor degrees $N = 10, 15, 20, 25, 30, 35$.

Model	Mean direct RMSE	Best direct RMSE	Mean iter.	Best iter.	Mean iter. red.
MLP	4.462×10^{-4}	3.550×10^{-4}	2.382	2.275	59.72%
GRU	5.762×10^{-4}	4.837×10^{-4}	2.239	2.142	62.14%
LSTM	6.771×10^{-4}	5.854×10^{-4}	2.243	2.111	62.07%
Transformer	6.791×10^{-4}	5.857×10^{-4}	2.199	2.135	62.81%

These scalar results refine the interpretation of the backbone comparison. The MLP is the most accurate direct scalar corrector, while recurrent and attention-based models

can provide more efficient Newton warm starts. Therefore, evaluating learned initializers only by direct prediction error is insufficient. For hybrid neural-Newton solvers, both direct-stage error and post-refinement computational cost should be reported.

8.5 Direct-Regression Ablation

To test whether the baseline-aware correction target is necessary, we additionally evaluate a direct-regression ablation. In this setting, the model is trained without the initial correction formulation: instead of predicting a normalized displacement from the baseline initializer, it directly predicts the target state used as the Newton initializer. The same degree/model grouping, validation-based trial selection, and Newton-refinement protocol are used. Therefore, the comparison isolates the effect of using baseline-aware correction rather than direct solution regression.

Table 12 reports the best direct-regression trial for each Taylor degree and model backbone. Compared with the baseline-aware correction results in Tables 9–11, direct regression is substantially less stable. Its direct RMSE values are typically in the 10^{-3} – 10^{-2} range, whereas the correction-based scalar models reach the 10^{-4} range. In addition, the direct-regression convergence ratios remain below 1.0 for all reported entries, with several MLP and Transformer settings below 0.90. This supports the main design choice of the paper: learning a correction relative to a physically meaningful initializer is easier and more Newton-friendly than directly regressing the solution state.

Table 12 Direct-regression ablation results grouped by Taylor degree and model backbone. The best trial for each degree-model pair is selected by convergence ratio, Newton iteration count, and direct RMSE.

Degree	Model	Trial	Direct RMSE	Direct R^2	Newton RMSE	Newton Residual Mean	Newton Iter.	Conv. Ratio
10	GRU	25	3.781e-03	0.99996	2.859e-02	2.838e-01	4.028	0.95100
10	LSTM	16	6.058e-03	0.99990	1.608e-03	2.196e-01	4.290	0.97340
10	MLP	1	1.325e-02	0.99954	3.675e-02	9.073e-01	3.992	0.88620
15	GRU	26	4.673e-03	0.99994	1.466e-03	3.424e-01	4.127	0.95240
15	LSTM	11	5.467e-03	0.99992	2.616e-02	3.186e-01	3.900	0.95210
15	MLP	1	2.030e-02	0.99893	3.180e-02	8.094e-01	4.386	0.91060
15	Transformer	42	7.502e-03	0.99985	3.078e-02	9.017e-01	3.737	0.86840
20	GRU	31	5.193e-03	0.99993	1.877e-03	4.756e-01	3.808	0.92540
20	LSTM	23	6.738e-03	0.99988	2.468e-03	5.423e-01	4.196	0.93450
20	MLP	3	2.131e-02	0.99882	1.014e-02	9.507e-01	4.109	0.86870
20	Transformer	41	7.293e-03	0.99986	3.009e-03	6.740e-01	3.746	0.88920
25	GRU	31	5.893e-03	0.99991	2.673e-02	4.284e-01	4.249	0.95360
25	LSTM	15	5.269e-03	0.99993	1.754e-03	3.263e-01	4.032	0.95200
25	MLP	4	2.696e-02	0.99811	1.230e-02	1.001e+00	4.208	0.86990
30	GRU	37	5.618e-03	0.99992	1.987e-03	5.001e-01	4.112	0.92600
30	LSTM	10	5.110e-03	0.99993	8.231e-04	2.514e-01	4.453	0.97570
30	MLP	1	2.140e-02	0.99881	3.460e-02	6.622e-01	4.029	0.90330
35	GRU	32	7.944e-03	0.99984	1.625e-02	3.763e-01	4.402	0.95580
35	LSTM	11	6.184e-03	0.99990	1.722e-03	3.870e-01	4.190	0.94640
35	MLP	2	2.747e-02	0.99803	1.275e-02	1.013e+00	4.122	0.86420
35	Transformer	41	8.411e-03	0.99982	4.299e-03	1.036e+00	3.250	0.82580

The ablation also clarifies the role of the Newton stage. Direct regression can sometimes produce a Newton-refined RMSE near 10^{-3} or 10^{-4} , but its convergence

ratio and residual mean are consistently weaker than the correction-based formulation. The proposed method therefore benefits not merely from using a neural network, but from using the network to predict a baseline-relative displacement that is aligned with the subsequent Newton refinement.

8.6 Additional Component Ablations: Taylor Features and Logit Correction

The direct-regression ablation above evaluates the target formulation. We further evaluate two component-level ablations at the representative coupled-system degree $K = 25$. The first removes Taylor coefficient features while retaining the baseline-aware correction formulation. Two variants are considered: a zero-center variant, in which the Taylor-related channel is suppressed, and a degree-zero keep-center variant, in which center-related information is retained while higher-order Taylor information is removed. The second replaces the proposed logit-space flow-ratio correction with raw ratio-space additive correction, while keeping the baseline-aware target and the same model backbones. These experiments isolate whether the local Taylor representation and the logit-space flow-split parameterization are necessary under the same Newton-refinement protocol.

Table 13 reports the no-Taylor ablation. The results show that the baseline-aware correction target and global/baseline inputs already provide a strong learning signal in the present synthetic distribution. In particular, the no-Taylor MLP variants remain highly accurate and require approximately 2.12–2.23 Newton iterations after refinement. This result should be interpreted cautiously: it does not prove that Taylor features are unnecessary in general, because the present benchmark is in-distribution and already provides informative global parameters and a physically meaningful baseline state. Instead, the current evidence suggests that Taylor information is an evaluated representation whose benefit is problem- and distribution-dependent, while the dominant contribution in this benchmark is the baseline-aware correction formulation combined with the baseline initializer.

Table 13 No-Taylor feature ablation at Taylor degree $K = 25$ in the coupled pipe-flow benchmark. Both variants retain baseline-aware correction and Newton refinement; they differ in whether center-related information is suppressed or retained.

Variant	Model	Direct RMSE	Direct Residual Mean	Newton RMSE	Newton Residual Mean	Iter. Mean	Conv. Ratio
Zero-center	MLP	2.123e-04	3.830e-03	3.101e-09	6.363e-14	2.123	1.0000
Zero-center	LSTM	1.865e-03	1.175e-02	3.101e-09	8.016e-14	2.484	1.0000
Zero-center	GRU	1.826e-03	1.986e-02	3.101e-09	7.844e-14	2.493	1.0000
Zero-center	Transformer	2.204e-03	1.365e-02	3.101e-09	8.341e-14	2.479	1.0000
Keep-center	MLP	2.367e-04	4.725e-03	3.101e-09	6.823e-14	2.233	1.0000
Keep-center	LSTM	6.916e-04	3.199e-03	3.101e-09	6.334e-14	2.147	1.0000
Keep-center	GRU	6.592e-04	5.358e-03	3.101e-09	8.221e-14	2.249	1.0000
Keep-center	Transformer	9.748e-04	5.052e-03	3.101e-09	7.837e-14	2.281	1.0000

Table 14 reports the no-logit correction ablation. Although the direct RMSE values remain small, replacing the logit-space update by raw ratio-space additive correction causes all evaluated backbones to hit the maximum Newton iteration limit under

the strict tolerance, with a convergence ratio of 0.0. This is an important diagnostic result: direct-stage error alone is not sufficient to certify a Newton-friendly warm start. The proposed logit-space correction keeps the flow split in a physically meaningful parameterization and is substantially more compatible with the subsequent Newton refinement stage.

Table 14 No-logit correction ablation at Taylor degree $K = 25$ in the coupled pipe-flow benchmark. The baseline-aware correction target is retained, but the flow ratio is corrected in raw ratio space instead of logit space.

Model	Direct RMSE	Direct Residual Mean	Newton RMSE	Newton Residual Mean	Iter. Mean	Conv. Ratio
MLP	4.935e-04	3.339e-02	4.119e-03	3.011e-04	20.000	0.0000
LSTM	5.999e-04	2.431e-02	1.186e-02	1.296e-03	20.000	0.0000
GRU	6.237e-04	3.913e-02	2.120e-02	3.193e-03	20.000	0.0000
Transformer	6.174e-04	2.925e-02	1.606e-02	1.200e-03	20.000	0.0000

Taken together, the ablations refine the interpretation of the method. The direct-regression results show that the baseline-aware target is more stable than predicting the solution state directly. The no-logit results show that the logit-space flow-ratio parameterization is important for satisfying the strict post-Newton convergence criterion in the coupled benchmark. The no-Taylor results are more nuanced: they show that Taylor features are not strictly required for strong in-distribution performance in this dataset, but they should not be interpreted as evidence against Taylor features in broader settings. The paper therefore treats Taylor features as an evaluated representation rather than as the sole source of improvement.

8.7 Inference-Time Analysis

Because the proposed method adds a neural correction stage before Newton refinement, runtime overhead must be evaluated together with accuracy and iteration count. The timing results in Tables 15, 16, and 17 are all measured on the one-dimensional scalar validation benchmark with 10,000 test samples; they are not coupled-system timing results. Table 15 summarizes model-wise average inference-time measurements over all tested Taylor degrees in this scalar setting. The average isolated CPU forward stage is very small for every backbone, ranging from 0.001735 ms/sample for the MLP to 0.004743 ms/sample for the LSTM. The direct total time remains within approximately 0.22–0.23 ms/sample on average, while the complete neural-plus-Newton path remains close to 1 ms/sample for all models. Thus, the neural correction stage introduces a small overhead relative to the full refined computation in this scalar timing benchmark.

Table 16 reports the full degree-model inference-time comparison. The fastest complete neural-plus-Newton runtime is observed for the Transformer at $N = 25$, with 0.95452 ms/sample and 2.135 mean Newton iterations. The LSTM at $N = 30$ is similar, requiring 0.95576 ms/sample and 2.140 iterations. The slowest complete runtime among the listed neural configurations is 1.09543 ms/sample for the MLP at $N = 35$, which is consistent with its larger iteration count of 2.574. These results indicate that the runtime trend follows the post-refinement iteration count more strongly

Table 15 Model-wise average inference-time performance in the scalar validation benchmark over Taylor degrees $N = 10, 15, 20, 25, 30, 35$. Times are reported in ms/sample.

Model	Mean direct RMSE	Forward CPU	Forward+Copy	Direct total	Neural+Newton	Mean iter.	Conv.
MLP	4.462×10^{-4}	0.001735	0.005888	0.224548	1.049948	2.382	1.00000
GRU	5.762×10^{-4}	0.002182	0.005685	0.227237	1.009362	2.239	1.00000
LSTM	6.771×10^{-4}	0.004743	0.004640	0.229940	0.991962	2.243	1.00000
Transformer	6.791×10^{-4}	0.004543	0.005503	0.229820	0.984805	2.199	1.00000

than the isolated forward-pass time. Consequently, a slightly more expensive sequence-aware neural forward stage can still produce a faster end-to-end neural-plus-Newton computation when it yields a better Newton warm start.

Table 16 Inference-time comparison of the hybrid neural correction initializer in the scalar validation benchmark across Taylor degrees and model backbones.

Degree	Model	Direct RMSE	Forward CPU ms/sample	Forward+Copy ms/sample	Direct Total ms/sample	Neural+Newton ms/sample	Neural Iter.	Neural Conv.
10	GRU	5.857e-04	0.00032	0.00179	0.21900	0.99936	2.142	1.00000
10	LSTM	6.170e-04	0.00098	0.00487	0.21003	1.00302	2.216	1.00000
10	MLP	4.864e-04	0.00248	0.00658	0.22673	1.02335	2.340	1.00000
10	Transformer	7.048e-04	0.00568	0.00483	0.23078	1.02859	2.352	1.00000
15	GRU	4.837e-04	0.00268	0.00652	0.22952	0.99080	2.181	1.00000
15	LSTM	8.106e-04	0.00552	0.00467	0.23098	1.00810	2.352	1.00000
15	MLP	5.711e-04	0.00062	0.00466	0.21312	1.05428	2.421	1.00000
15	Transformer	7.611e-04	0.00133	0.00777	0.22153	0.99198	2.173	1.00000
20	GRU	5.709e-04	0.00285	0.00634	0.22958	1.03425	2.326	1.00000
20	LSTM	6.371e-04	0.00536	0.00441	0.23089	1.00985	2.303	1.00000
20	MLP	3.755e-04	0.00287	0.00636	0.23215	1.03351	2.324	1.00000
20	Transformer	7.370e-04	0.00419	0.00634	0.22396	0.96208	2.139	1.00000
25	GRU	6.200e-04	0.00133	0.00743	0.22907	1.01782	2.249	1.00000
25	LSTM	6.323e-04	0.00562	0.00475	0.24420	0.96783	2.111	1.00000
25	MLP	3.550e-04	0.00068	0.00595	0.21847	1.02182	2.275	1.00000
25	Transformer	5.857e-04	0.00568	0.00515	0.23185	0.95452	2.135	1.00000
30	GRU	5.723e-04	0.00302	0.00564	0.22931	1.01042	2.306	1.00000
30	LSTM	5.854e-04	0.00566	0.00479	0.23342	0.95576	2.140	1.00000
30	MLP	4.228e-04	0.00070	0.00603	0.22057	1.06530	2.360	1.00000
30	Transformer	6.553e-04	0.00553	0.00519	0.22970	0.97823	2.206	1.00000
35	GRU	6.243e-04	0.00289	0.00639	0.22694	1.00352	2.231	1.00000
35	LSTM	7.799e-04	0.00532	0.00435	0.23012	1.00721	2.336	1.00000
35	MLP	4.664e-04	0.00306	0.00575	0.23625	1.09543	2.574	1.00000
35	Transformer	6.309e-04	0.00485	0.00374	0.24110	0.99343	2.188	1.00000

Table 17 reports CPU-only runtime for the nonlearned baseline initializers with Newton refinement on the same 10,000-sample scalar validation benchmark. The heuristic initializer requires 2.43743 ms/sample in total because its mean Newton iteration count is 5.9136. Stronger explicit initializers reduce the CPU-only total time to 1.25075–1.30679 ms/sample by lowering the mean iteration count to approximately 3.08–3.22. Compared with these CPU-only baselines, the hybrid neural-plus-Newton configurations in Table 16 require approximately 0.95452–1.09543 ms/sample for the complete corrected-initializer and Newton-refinement path. This shows that the learned correction is not only iteration-efficient but also competitive in measured end-to-end runtime for the tested scalar benchmark.

Overall, the runtime results close an important practical gap in the evaluation. The learned correction is not only accurate and iteration-efficient, but also computationally lightweight in this benchmark. The average neural-plus-Newton time over all degree-model combinations is approximately 1.009 ms/sample, and the best hybrid

Table 17 CPU-only runtime of nonlearned baseline initializers with Newton refinement in the scalar validation benchmark. All times are reported in milliseconds per test sample and include only CPU execution.

(a) Accuracy and residuals					
Initializer	N Samples	Direct RMSE	Direct Residual Mean	Final RMSE	Final Residual Mean
Heuristic	10000	2.124e-01	1.114e+02	3.101e-09	5.621e-14
Cond.-Haaland	10000	2.617e-02	3.746e-02	3.101e-09	4.795e-14
Cond.-Swamee-Jain	10000	2.597e-02	5.691e-02	3.101e-09	4.487e-14
Cond.-Serghides	10000	2.456e-02	2.620e-02	3.101e-09	4.941e-14
(b) Runtime and convergence					
Initializer	Init CPU ms/sample	Newton CPU ms/sample	Total CPU ms/sample	Iter. Mean	Conv. Ratio
Heuristic	0.02764	2.40980	2.43743	5.9136	0.99990
Cond.-Haaland	0.00008	1.25067	1.25075	3.0825	1.00000
Cond.-Swamee-Jain	0.00011	1.30668	1.30679	3.2249	1.00000
Cond.-Serghides	0.00012	1.29874	1.29886	3.1755	1.00000

configurations are faster than the CPU-only explicit-initializer baselines measured in Table 17. This supports the use of the proposed method in repeated simulation or parameter-sweep settings, where a trained correction model can be reused over many nonlinear pipe-flow instances.

9 Limitations and Additional Validation Requirements

The present results demonstrate that baseline-aware neural correction can reduce Newton refinement cost under the controlled scalar and two-branch pipe-flow settings considered in this study. Nevertheless, several validation gaps remain and should be addressed before making broader claims about general engineering deployment.

9.1 Ablation Study Requirements

The current experiments compare nonlearned explicit initializers, neural correction backbones, Taylor truncation degrees, direct-stage accuracy, Newton-refined accuracy, runtime, and three component ablations. The explicit-initializer-only comparison is reported in Tables 3, 4, 7, and 8. The direct-regression ablation is reported in Table 12, the no-Taylor feature ablation is reported in Table 13, and the no-logit correction ablation is reported in Table 14. These results show that baseline-aware correction and logit-space flow-ratio correction are important for Newton-friendly initialization. The Taylor-feature ablation gives a more nuanced result: no-Taylor variants remain competitive in the present in-distribution benchmark, so Taylor coefficients are treated as an evaluated representation rather than as a universally necessary component. The remaining unreported component ablation is removal of the baseline input while retaining the correction target.

Table 18 Component ablation status for isolating the contribution of each design choice.

Ablation item			Purpose	Current status
Explicit only	initializer		Strong nonlearned baseline	Reported
Direct regression			Compare solution regression with correction learning	Reported
Remove Taylor features	fea-		Test value of local residual coefficients	Reported in Table 13
Remove logit correction	correc-		Test stability of flow-ratio parameterization	Reported in Table 14
Remove input	baseline		Test baseline-aware conditioning	Requires retraining

9.2 Runtime Benchmarking Limitations

Section 8.7 reports both hybrid neural-pipeline timing and CPU-only nonlearned baseline timing for the scalar 10,000-sample benchmark. The previous absence of runtime comparison against heuristic and explicit Newton baselines is therefore addressed for this setting. The current timing claim should still be interpreted carefully. First, the runtime study is reported for the scalar benchmark, not for the full two-branch coupled system. Second, the measured values depend on the implementation, vectorization strategy, and hardware configuration used by the runtime scripts. Therefore, the manuscript claims measured CPU-only efficiency for the reported scalar benchmark, but it does not claim a universal speedup guarantee for all pipe-network configurations, larger networks, or deployment hardware.

9.3 Out-of-Distribution Generalization

The reported train, validation, and test sets are sampled from the same synthetic generation distribution with fixed random-seed offsets. This is appropriate for measuring in-distribution generalization and leakage-controlled performance, but it does not establish robustness outside the sampled range. A stronger test should evaluate Reynolds number, roughness, diameter ratio, total-flow regimes, and combined parameter shifts outside the training distribution. Since these results were not provided in the current experiment logs, the present manuscript does not claim out-of-distribution robustness.

9.4 Scope and Failure Modes

The main coupled experiment is restricted to a two-branch parallel pipe-flow system. The method has not yet been validated on looped networks, multi-branch systems, transient hydraulic simulations, or real measured pipe-network data. The datasets are synthetic and should be interpreted as controlled benchmarks rather than a complete representation of all hydraulic operating conditions. In addition, neural inference introduces overhead before Newton refinement. Although the measured CPU-only hybrid runtime is competitive in the present scalar benchmark, the overhead may be more significant on lower-power embedded platforms, smaller processors, or nonvectorized implementations. Finally, failure modes may occur when the learned correction moves

the initializer outside a favorable Newton basin, when the physical parameters lie far outside the training distribution, or when the coupled residual becomes ill-conditioned. These limitations motivate future work on larger networks, out-of-distribution testing, basin-aware loss functions, uncertainty-based fallback to explicit initializers, and CPU-oriented runtime optimization.

10 Conclusion and Future Work

This paper presented a hybrid deep-learning and Newton-refinement framework for nonlinear pipe-flow equations. The proposed method combines a nonlearned baseline initializer, a baseline-aware learned correction model, and Newton refinement. The central objective is not to replace Newton’s method, but to provide a more refinement-friendly initial point for the original nonlinear system.

The coupled pipe-flow experiments show that learned correction models improve both direct initialization quality and refinement efficiency. In particular, the mean Newton iteration count decreases from 5.9251 for the heuristic baseline and 3.3129 for the strongest conductance-based explicit initializer to approximately 2.61–2.73 for the learned correction models, while final RMSE remains on the order of 10^{-9} . These results indicate that the neural module is most useful as a warm-start accelerator, not as a replacement for explicit engineering approximations or Newton-type solvers.

The scalar Taylor-degree sweep further clarifies the role of the learned initializer. Across $N = 10, 15, 20, 25, 30, 35$, neural correction reduces the direct RMSE by approximately 99.62–99.83% relative to the heuristic initializer. After Newton refinement, all neural warm starts reach the same final RMSE of 3.101×10^{-9} , but the neural models reduce the mean Newton iteration count from 5.914 to approximately 2.11–2.57 depending on degree and architecture. The best direct RMSE is obtained by the MLP at $N = 25$, whereas the best iteration reduction is obtained by the LSTM at $N = 25$. These results reinforce the interpretation that direct prediction accuracy and Newton refinement efficiency should both be evaluated when assessing learned warm starts. The component ablations further refine the design interpretation. Direct regression is less stable than baseline-aware correction, raw-ratio flow correction fails to satisfy the strict post-Newton convergence criterion in the coupled benchmark, and no-Taylor variants remain competitive under the present in-distribution setting. Thus, the strongest evidence is for the baseline-aware target and logit-space flow parameterization, while the role of Taylor features should be interpreted as representation- and distribution-dependent rather than universally necessary.

Future work will extend the framework to larger pipe networks, looped hydraulic systems, and more general nonlinear network constraints. Additional research directions include the remaining baseline-input ablation, coupled-system runtime comparison against classical baselines, out-of-distribution generalization tests over Reynolds number and geometric ranges, basin-aware training objectives, residual-aware curricula, uncertainty-guided refinement, symbolic or AST-based hybrid representations, combinations with Halley or other higher-order refinements, larger-batch runtime benchmarking, and sharper theoretical analysis of correction-target parameterization for coupled nonlinear systems.

Acknowledgements. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2026-25482392).

Statements and Declarations

Funding. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2026-25482392).

Data availability. The datasets used in this study were synthetically generated. The dataset-generation scripts are publicly available in the GitHub repository, <https://github.com/jseongnam/Residual-Verified-Neural-Warmstart>.

Code availability. The code for dataset generation, model training, and evaluation is available in the GitHub repository, <https://github.com/jseongnam/Residual-Verified-Neural-Warmstart>.

References

- [1] Haaland, S. E.: Simple and Explicit Formulas for the Friction Factor in Turbulent Pipe Flow. *Journal of Fluids Engineering* 105(1), 89–90 (1983). <https://doi.org/10.1115/1.3240948>.
- [2] Swamee, P. K., Jain, A. K.: Explicit Equations for Pipe-Flow Problems. *Journal of the Hydraulics Division* 102(5), 657–664 (1976).
- [3] Serghides, T. K.: Estimate Friction Factor Accurately. *Chemical Engineering* 91(5), 63–64 (1984).
- [4] Sonnad, J. R., Goudar, C. T.: Turbulent Flow Friction Factor Calculation Using a Mathematically Exact Alternative to the Colebrook–White Equation. *Journal of Hydraulic Engineering* 132(8), 863–867 (2006). [https://doi.org/10.1061/\(ASCE\)0733-9429\(2006\)132:8\(863\)](https://doi.org/10.1061/(ASCE)0733-9429(2006)132:8(863)).
- [5] Sonnad, J. R., Goudar, C. T.: Explicit Reformulation of the Colebrook–White Equation for Turbulent Flow Friction Factor Calculation. *Industrial & Engineering Chemistry Research* 46(8), 2593–2600 (2007). <https://doi.org/10.1021/ie0613199>.
- [6] Brkić, D.: Review of Explicit Approximations to the Colebrook Relation for Flow Friction. *Journal of Petroleum Science and Engineering* 77(1), 34–48 (2011). <https://doi.org/10.1016/j.petrol.2011.02.006>.
- [7] Winning, H. K., Coole, T.: Explicit Friction Factor Accuracy and Computational Efficiency for Turbulent Flow in Pipes. *Flow, Turbulence and Combustion* 90(1), 1–27 (2013). <https://doi.org/10.1007/s10494-012-9419-7>.

- [8] Niazkar, M.: Revisiting the Estimation of Colebrook Friction Factor: A Comparison Between Artificial Intelligence Models and C-W Based Explicit Equations. *KSCE Journal of Civil Engineering* 23(10), 4311–4326 (2019).
- [9] Praks, P., Brkić, D.: Symbolic Regression-Based Genetic Approximations of the Colebrook Equation for Flow Friction. *Water* 10(9), 1175 (2018). <https://doi.org/10.3390/w10091175>.
- [10] Schwenkel, L., Gharbi, M., Trimpe, S., Ebenbauer, C.: Online Learning with Stability Guarantees: A Memory-Based Warm Starting for Real-Time MPC. *Automatica* 122, 109247 (2020). <https://doi.org/10.1016/j.automatica.2020.109247>.
- [11] Chen, S., Saulnier, K., Atanasov, N., Lee, D. D., Kumar, V., Pappas, G. J., Morari, M.: Approximating Explicit Model Predictive Control Using Constrained Neural Networks. In: *Proceedings of the 2018 Annual American Control Conference (ACC)*, 2018, pp. 1520–1527. <https://doi.org/10.23919/ACC.2018.8431275>.
- [12] Brkić, D., Praks, P.: Accurate and Efficient Explicit Approximations of the Colebrook Flow Friction Equation Based on the Wright ω -Function. *Mathematics* 7(1), 34 (2019). <https://doi.org/10.3390/math7010034>.
- [13] Praks, P., Brkić, D.: Advanced Iterative Procedures for Solving the Implicit Colebrook Equation for Fluid Flow Friction. *Advances in Civil Engineering* 2018, 5451034 (2018). <https://doi.org/10.1155/2018/5451034>.
- [14] Vatankehah, A. R.: Approximate Analytical Solutions for the Colebrook Equation. *Journal of Hydraulic Engineering* 144(5), 06018007 (2018). [https://doi.org/10.1061/\(ASCE\)HY.1943-7900.0001454](https://doi.org/10.1061/(ASCE)HY.1943-7900.0001454).
- [15] Traub, J. F.: *Iterative Methods for the Solution of Equations*. 2nd Edition. AMS Chelsea Publishing, Providence, 1982.
- [16] Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P.: *Numerical Recipes: The Art of Scientific Computing*. 3rd Edition. Cambridge University Press, Cambridge, 2007.
- [17] Quarteroni, A., Sacco, R., Saleri, F.: *Numerical Mathematics*. 2nd Edition. Springer, Berlin, 2007.
- [18] Ortega, J. M., Rheinboldt, W. C.: *Iterative Solution of Nonlinear Equations in Several Variables*. SIAM, Philadelphia, 2000.
- [19] Boyd, S., Vandenberghe, L.: *Convex Optimization*. Cambridge University Press, Cambridge, 2004.
- [20] Biberg, D.: Fast and Accurate Approximations for the Colebrook Equation. *Journal of Fluids Engineering* 139(3), 031401 (2017). <https://doi.org/10.1115/1.4034950>.

- [21] Niazkar, M., Talebbeydokhti, N.: Comparison of Explicit Relations for Calculating Colebrook Friction Factor in Pipe Network Analysis Using h-Based Methods. *Iranian Journal of Science and Technology, Transactions of Civil Engineering* 44, 231–249 (2020).
- [22] Raissi, M., Perdikaris, P., Karniadakis, G. E.: Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *Journal of Computational Physics* 378, 686–707 (2019). <https://doi.org/10.1016/j.jcp.2018.10.045>.
- [23] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., Duvenaud, D. K.: Neural Ordinary Differential Equations. In: *Advances in Neural Information Processing Systems*, Vol. 31, 2018.
- [24] Amos, B., Kolter, J. Z.: OptNet: Differentiable Optimization as a Layer in Neural Networks. In: *Proceedings of the International Conference on Machine Learning*, PMLR 70, 136–145 (2017).