Mathematical Foundations of Neural Network Track Extrapolation in LHCb

G. Scriven

January 2026

Abstract

This document presents the complete mathematical derivations for neural network-based track extrapolation in the LHCb detector. We cover the physics of charged particle motion in magnetic fields, derive the governing equations in z-parameterization, and develop three neural network architectures: Multi-Layer Perceptron (MLP), Physics-Informed Neural Network (PINN), and Runge-Kutta Physics-Informed Neural Network (RK-PINN). We also provide a comprehensive treatment of Runge-Kutta numerical integration theory that motivates the RK-PINN architecture.

Contents

1 Introduction

Track reconstruction in the LHCb detector requires extrapolating particle trajectories through the magnetic field. The traditional approach uses numerical integration of the equations of motion (Runge-Kutta methods). We explore neural network alternatives that learn the extrapolation mapping directly from data or by incorporating physics constraints.

1.1 Notation

Throughout this document, we use the following notation:

- $\boldsymbol{x} = (x, y, z)^T$ - spatial position in mm
- $\boldsymbol{p} = (p_x, p_y, p_z)^T$ - momentum in MeV
- $\boldsymbol{B} = (B_x, B_y, B_z)^T$ - magnetic field in Tesla
- $t_x = \frac{dx}{dz}$, $t_y = \frac{dy}{dz}$ - track slopes (dimensionless)
- q/p - charge over momentum in MeV^{-1}
- $c_{\text{light}} = 2.99792458 \times 10^{-4}$ - speed of light factor

2 Physics of Charged Particle Motion

2.1 The Lorentz Force

A charged particle with charge q and velocity $\boldsymbol{v}$ in a magnetic field $\boldsymbol{B}$ experiences the Lorentz force:

$$\boldsymbol{F} = q(\boldsymbol{v} \times \boldsymbol{B}) \tag{1}$$

Using Newton's second law and the relativistic momentum $\boldsymbol{p} = \gamma m \boldsymbol{v}$:

$$\frac{d\boldsymbol{p}}{dt} = q(\boldsymbol{v} \times \boldsymbol{B}) \tag{2}$$

Remark 2.1. The Lorentz force is perpendicular to the velocity, so it does no work on the particle. Energy (and thus $|\boldsymbol{p}|$) is conserved in a pure magnetic field.

2.2 Z-Parameterization

In LHCb, tracks propagate predominantly in the $+z$ direction. It is therefore convenient to parameterize the trajectory by z rather than time. We define the **track state** at position z as:

$$\boldsymbol{y}(z) = \begin{pmatrix} x(z) \\ y(z) \\ t_x(z) \\ t_y(z) \end{pmatrix} \tag{3}$$

where $t_x = \frac{dx}{dz}$ and $t_y = \frac{dy}{dz}$ are the track slopes.
The relationship between time and z derivatives is:

$$\frac{d}{dt} = \frac{v_z}{1}\frac{d}{dz} = \frac{p_z}{\gamma m}\frac{d}{dz} \tag{4}$$

2.3 Derivation of the Equations of Motion

Theorem 2.1 (Equations of Motion in Z-Parameterization)**.** The evolution of the track state $\boldsymbol{y}(z)$ is governed by:

$$\frac{dx}{dz} = t_x \tag{5}$$

$$\frac{dy}{dz} = t_y \tag{6}$$

$$\frac{dt_x}{dz} = \kappa \cdot N \cdot \left[t_x t_y B_x - (1 + t_x^2) B_y + t_y B_z\right] \tag{7}$$

$$\frac{dt_y}{dz} = \kappa \cdot N \cdot \left[(1 + t_y^2) B_x - t_x t_y B_y - t_x B_z\right] \tag{8}$$

where:

$$\kappa = \frac{q}{p} \cdot c_{\text{light}}, \qquad N = \sqrt{1 + t_x^2 + t_y^2} \tag{9}$$

Proof. We derive the slope equations starting from the Lorentz force. The momentum components are related to the slopes by:

$$p_x = p \cdot \frac{t_x}{N}, \quad p_y = p \cdot \frac{t_y}{N}, \quad p_z = p \cdot \frac{1}{N} \tag{10}$$

where $N = \sqrt{1 + t_x^2 + t_y^2}$ ensures $|\boldsymbol{p}| = p$.

The velocity is $\boldsymbol{v} = \boldsymbol{p}/(\gamma m)$, so:

$$v_z = \frac{p_z}{\gamma m} = \frac{p}{N \gamma m} \tag{11}$$

From the Lorentz force $\frac{d\boldsymbol{p}}{dt} = q(\boldsymbol{v} \times \boldsymbol{B})$, the x-component is:

$$\frac{dp_x}{dt} = q(v_y B_z - v_z B_y) = \frac{q}{\gamma m}(p_y B_z - p_z B_y) \tag{12}$$

Converting to z-derivatives using $\frac{dt}{d=} v_z \frac{dz}{d}$:

$$\frac{dp_x}{dz} = \frac{1}{v_z}\frac{dp_x}{dt} = \frac{N\gamma m}{p} \cdot \frac{q}{\gamma m}(p_y B_z - p_z B_y) \tag{13}$$

Simplifying:

$$\frac{dp_x}{dz} = \frac{qN}{p}\left(\frac{p t_y}{N} B_z - \frac{p}{N} B_y\right) = q(t_y B_z - B_y) \tag{14}$$

Now, $t_x = p_x/p_z = p_x N/p$, so:

$$\frac{dt_x}{dz} = \frac{N}{p}\frac{dp_x}{dz} + \frac{p_x}{p}\frac{dN}{dz} \tag{15}$$

Since $|\boldsymbol{p}|$ is conserved and N depends on slopes:

$$\frac{dN}{dz} = \frac{t_x}{N}\frac{dt_x}{dz} + \frac{t_y}{N}\frac{dt_y}{dz} \tag{16}$$

After algebraic manipulation (similar derivation for t_y), we obtain:

$$\frac{dt_x}{dz} = \frac{q}{p} c_{\text{light}} \cdot N \cdot \left[t_x t_y B_x - (1 + t_x^2) B_y + t_y B_z\right] \tag{17}$$

$$\frac{dt_y}{dz} = \frac{q}{p} c_{\text{light}} \cdot N \cdot \left[(1 + t_y^2) B_x - t_x t_y B_y - t_x B_z\right] \tag{18}$$

where $c_{\text{light}} = 2.99792458 \times 10^{-4}$ accounts for unit conversions. □

2.4 Compact Form

We can write the system compactly as:

$$\frac{d\boldsymbol{y}}{dz} = \boldsymbol{f}(\boldsymbol{y}, z; q/p) \tag{19}$$

where $\boldsymbol{f} : \mathbb{R}^4 \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^4$ is defined by Equations (5)–(8).

3 The LHCb Magnetic Field

3.1 Dipole Magnet Characteristics

The LHCb dipole magnet produces a field primarily in the y-direction (vertical):

- Peak field: $|B_y| \approx 1.03$ T at $z \approx 5000$ mm
- Field integral: $\int B_y\, dz \approx 4.44$ T·m
- Dominant component: B_y (causes bending in the x-z plane)

3.2 Gaussian Field Approximation

For analytical work and differentiable physics loss, we approximate the field as:

$$B_y(z) = B_0 \exp\left(-\frac{1}{2}\left(\frac{z - z_c}{\sigma_z}\right)^2\right) \tag{20}$$

with fitted parameters:

$$B_0 = -1.0182 \text{ T}, \quad z_c = 5007 \text{ mm}, \quad \sigma_z = 1744 \text{ mm} \tag{21}$$

4 Multi-Layer Perceptron (MLP)

4.1 Architecture

The MLP learns the extrapolation mapping directly from data without explicit physics. Given input features:

$$\boldsymbol{x} = (x_0, y_0, t_{x,0}, t_{y,0}, q/p, \Delta z)^T \in \mathbb{R}^6 \tag{22}$$

the network predicts the final state:

$$\hat{\boldsymbol{y}} = (x_f, y_f, t_{x,f}, t_{y,f})^T \in \mathbb{R}^4 \tag{23}$$

4.2 Network Definition

Definition 4.1 (Multi-Layer Perceptron)**.** An MLP with L hidden layers is defined as:

$$\text{MLP}(\boldsymbol{x}) = \mathbf{W}_{L+1}\sigma_L(\mathbf{W}_L\sigma_{L-1}(\cdots\sigma_1(\mathbf{W}_1\boldsymbol{x} + \boldsymbol{b}_1)\cdots) + \boldsymbol{b}_L) + \boldsymbol{b}_{L+1} \tag{24}$$

where $\mathbf{W}_i \in \mathbb{R}^{d_i \times d_{i-1}}$ are weight matrices, $\boldsymbol{b}_i$ are bias vectors, and σ_i are activation functions.

4.3 Input/Output Normalization

For stable training, we apply z-score normalization:

$$\tilde{x}_i = \frac{x_i - \mu_i}{\sigma_i} \tag{25}$$

where μ_i and σ_i are the mean and standard deviation computed from training data.

The network operates on normalized inputs and outputs:

$$\hat{\boldsymbol{y}} = \sigma_y \odot \text{MLP}(\tilde{\boldsymbol{x}}) + \mu_y \tag{26}$$

4.4 Loss Function

The MLP is trained with Mean Squared Error (MSE) loss:

$$\mathcal{L}_{\text{data}} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i^*\|^2 \tag{27}$$

where $\boldsymbol{y}_i^*$ is the ground truth from Runge-Kutta integration.

4.5 Implicit Physics Learning

Remark 4.1. The MLP learns the physics *implicitly* from training data generated by the RK extrapolator. The network approximates the flow map:

$$\Phi_{\Delta z} : \boldsymbol{y}_0 \mapsto \boldsymbol{y}(\Delta z) \tag{28}$$

which is the solution operator for the ODE system (19).

5 Physics-Informed Neural Network (PINN)

5.1 Concept

Physics-Informed Neural Networks (PINNs) incorporate the governing physics equations directly into the loss function. Instead of learning only from data, the network is constrained to satisfy the differential equations.

5.2 Network as Trajectory Function

Unlike the MLP which predicts only the endpoint, the PINN learns the continuous trajectory:

$$\boldsymbol{y}_\theta : [0, 1] \to \mathbb{R}^4, \quad \zeta \mapsto \boldsymbol{y}_\theta(\zeta) \tag{29}$$

where $\zeta = (z - z_0)/\Delta z \in [0, 1]$ is the normalized position and θ represents network parameters.

5.3 Loss Function Components

The total PINN loss consists of three components:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}} + \lambda_{\text{PDE}}\mathcal{L}_{\text{PDE}} \tag{30}$$

5.3.1 Data Loss

At the endpoint $\zeta = 1$:

$$\mathcal{L}_{\text{data}} = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{y}_\theta(\zeta = 1; \boldsymbol{x}_i) - \boldsymbol{y}_i^*\|^2 \tag{31}$$

5.3.2 Initial Condition Loss

At $\zeta = 0$, the trajectory must match the initial state:

$$\mathcal{L}_{\text{IC}} = \frac{1}{N}\sum_{i=1}^{N} \|\boldsymbol{y}_\theta(\zeta = 0; \boldsymbol{x}_i) - \boldsymbol{y}_{0,i}\|^2 \tag{32}$$

where $\boldsymbol{y}_{0,i} = (x_{0,i}, y_{0,i}, t_{x,0,i}, t_{y,0,i})^T$.

5.3.3 PDE Residual Loss

The physics constraint is enforced at *collocation points* $\{\zeta_k\}_{k=1}^{K}$ sampled along the trajectory:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K}\left\| \left.\frac{d\boldsymbol{y}_\theta}{d\zeta}\right|_{\zeta_k} - \Delta z \cdot \boldsymbol{f}(\boldsymbol{y}_\theta(\zeta_k), z_k; q/p_i) \right\|^2 \tag{33}$$

5.4 Automatic Differentiation

The key insight of PINNs is that the derivative $\frac{\partial \boldsymbol{y}_\theta}{\partial \zeta}$ can be computed exactly using automatic differentiation:

$$\frac{\partial y_j}{\partial \zeta} = \frac{\partial}{\partial \zeta}\text{NN}_j(\boldsymbol{x}, \zeta; \theta) \tag{34}$$

This is computed via backpropagation through the network graph.

5.5 Algorithm

The PINN training procedure follows these steps:

1. **Input:** Batch of initial states $\{\boldsymbol{x}_i\}$, ground truth $\{\boldsymbol{y}_i^*\}$
2. Sample collocation points $\{\zeta_k\}_{k=1}^{K} \sim \text{Uniform}(0, 1)$
3. For each sample i:
   - Compute $\boldsymbol{y}_\theta(\zeta = 0)$, $\boldsymbol{y}_\theta(\zeta = 1)$, and $\boldsymbol{y}_\theta(\zeta_k)$ for all k
   - Compute $\frac{\partial \boldsymbol{y}_\theta}{\partial \zeta}$ at each ζ_k via autodiff
   - Evaluate physics residual using Eqs. (5)–(8)
4. Compute $\mathcal{L}_{\text{total}}$ using Eq. (30)
5. Update θ via gradient descent

6 Runge-Kutta Numerical Integration

6.1 General Framework

Runge-Kutta methods approximate the solution of an initial value problem:

$$\frac{d\boldsymbol{y}}{dz} = \boldsymbol{f}(\boldsymbol{y}, z), \quad \boldsymbol{y}(z_0) = \boldsymbol{y}_0 \tag{35}$$

Definition 6.1 (Explicit Runge-Kutta Method)**.** An s-stage explicit RK method advances the solution from $\boldsymbol{y}_n$ to $\boldsymbol{y}_{n+1}$ over step h by:

$$\boldsymbol{k}_1 = h \cdot \boldsymbol{f}(z_n, \boldsymbol{y}_n) \tag{36}$$

$$\boldsymbol{k}_2 = h \cdot \boldsymbol{f}(z_n + c_2 h, \boldsymbol{y}_n + a_{21}\boldsymbol{k}_1) \tag{37}$$

$$\boldsymbol{k}_3 = h \cdot \boldsymbol{f}(z_n + c_3 h, \boldsymbol{y}_n + a_{31}\boldsymbol{k}_1 + a_{32}\boldsymbol{k}_2) \tag{38}$$

$$\vdots \tag{39}$$

$$\boldsymbol{k}_s = h \cdot \boldsymbol{f}(z_n + c_s h, \boldsymbol{y}_n + \sum_{j=1}^{s-1} a_{sj}\boldsymbol{k}_j) \tag{40}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \sum_{i=1}^{s} b_i \boldsymbol{k}_i \tag{41}$$

6.2 Butcher Tableau

The RK method is characterized by its Butcher tableau:

$$\begin{array}{c|cccc} c_1 & 0 & & & \\ c_2 & a_{21} & 0 & & \\ c_3 & a_{31} & a_{32} & 0 & \\ \vdots & \vdots & & \ddots & \\ c_s & a_{s1} & a_{s2} & \cdots & 0 \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \tag{42}$$

6.3 Classical RK4

The classical fourth-order Runge-Kutta method (RK4) has the tableau:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} \tag{43}$$

Explicitly:

$$\boldsymbol{k}_1 = h \cdot \boldsymbol{f}(z_n, \boldsymbol{y}_n) \tag{44}$$

$$\boldsymbol{k}_2 = h \cdot \boldsymbol{f}\left(z_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{\boldsymbol{k}_1}{2}\right) \tag{45}$$

$$\boldsymbol{k}_3 = h \cdot \boldsymbol{f}\left(z_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{\boldsymbol{k}_2}{2}\right) \tag{46}$$

$$\boldsymbol{k}_4 = h \cdot \boldsymbol{f}(z_n + h, \boldsymbol{y}_n + \boldsymbol{k}_3) \tag{47}$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4) \tag{48}$$

6.4 Order Conditions

Theorem 6.1 (Local Truncation Error)**.** The local truncation error of RK4 is $O(h^5)$, making it a fourth-order method with global error $O(h^4)$.

The order conditions are derived by matching Taylor series coefficients. For order p, the method must satisfy:

$$\boldsymbol{y}(z_n + h) - \boldsymbol{y}_{n+1} = O(h^{p+1}) \tag{49}$$

6.5 Geometric Interpretation

RK4 can be interpreted as:

1. $\boldsymbol{k}_1$: Slope at the starting point
2. $\boldsymbol{k}_2$: Slope at the midpoint using $\boldsymbol{k}_1$
3. $\boldsymbol{k}_3$: Slope at the midpoint using $\boldsymbol{k}_2$
4. $\boldsymbol{k}_4$: Slope at the endpoint using $\boldsymbol{k}_3$

The final update is a weighted average: $(1 + 2 + 2 + 1)/6 = 1$.

7 RK-PINN: Runge-Kutta Physics-Informed Neural Network

7.1 Motivation

The RK-PINN combines the structure of Runge-Kutta methods with neural network learning. The key insight is that the intermediate stages of RK methods provide natural positions for physics constraints.

7.2 Architecture

Definition 7.1 (RK-PINN)**.** The RK-PINN consists of:

1. **Shared backbone $\boldsymbol{h}_\theta$** $: \mathbb{R}^6 \to \mathbb{R}^d$ that extracts features
2. **Stage heads** $\{g_{\phi_i}\}_{i=1}^s$ that predict states at stage positions
3. **Learnable weights** $\{w_i\}_{i=1}^s$ for combining stage outputs

For a 4-stage RK-PINN (analogous to RK4):

$$\boldsymbol{h} = \text{Backbone}(\boldsymbol{x}) \tag{50}$$

$$\hat{\boldsymbol{y}}_1 = g_{\phi_1}(\boldsymbol{h}, \zeta = 0.25) \quad (\text{at } z_0 + 0.25\Delta z) \tag{51}$$

$$\hat{\boldsymbol{y}}_2 = g_{\phi_2}(\boldsymbol{h}, \zeta = 0.50) \quad (\text{at } z_0 + 0.50\Delta z) \tag{52}$$

$$\hat{\boldsymbol{y}}_3 = g_{\phi_3}(\boldsymbol{h}, \zeta = 0.75) \quad (\text{at } z_0 + 0.75\Delta z) \tag{53}$$

$$\hat{\boldsymbol{y}}_4 = g_{\phi_4}(\boldsymbol{h}, \zeta = 1.00) \quad (\text{at } z_0 + \Delta z) \tag{54}$$

$$\hat{\boldsymbol{y}}_{\text{final}} = \sum_{i=1}^{4} \tilde{w}_i \hat{\boldsymbol{y}}_i \tag{55}$$

where $\tilde{w}_i = \text{softmax}(\boldsymbol{w})_i$ ensures weights sum to 1.

7.3 Weight Initialization

The weights are initialized to match RK4:

$$w_1^{(0)} = \frac{1}{6}, \quad w_2^{(0)} = \frac{2}{6}, \quad w_3^{(0)} = \frac{2}{6}, \quad w_4^{(0)} = \frac{1}{6} \tag{56}$$

During training, these weights can adapt to better fit the data.

7.4 Loss Function

The RK-PINN loss combines data and physics terms:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}} + \lambda_{\text{PDE}} \sum_{i=1}^{s} \mathcal{L}_{\text{PDE}}^{(i)} \tag{57}$$

where the PDE loss at each stage is:

$$\mathcal{L}_{\text{PDE}}^{(i)} = \frac{1}{N} \sum_{j=1}^{N} \left\| \frac{\partial \hat{\boldsymbol{y}}_i}{\partial \zeta} - \Delta z \cdot \boldsymbol{f}(\hat{\boldsymbol{y}}_i, z_i; q/p_j) \right\|^2 \tag{58}$$

Property	PINN	RK-PINN
Collocation points	Random sampling	Fixed at RK positions
Architecture	Single network	Backbone + stage heads
Stage weights	N/A	Learnable (init: RK4)
Physics structure	Soft constraint	Structured like RK

Table 1: Comparison of PINN and RK-PINN architectures.

7.5 Comparison with Standard PINN

8 Summary of Loss Functions

8.1 MLP

$$\mathcal{L}_{\text{MLP}} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i^*\|^2 \tag{59}$$

8.2 PINN

$$\mathcal{L}_{\text{PINN}} = \underbrace{\frac{1}{N}\sum_{i=1}^{N} \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i^*\|^2}_{\mathcal{L}_{\text{data}}} + \lambda_{\text{IC}} \underbrace{\frac{1}{N}\sum_{i=1}^{N} \|\hat{\boldsymbol{y}}_{0,i} - \boldsymbol{y}_{0,i}\|^2}_{\mathcal{L}_{\text{IC}}} + \lambda_{\text{PDE}} \underbrace{\frac{1}{NK}\sum_{i,k} \|\mathcal{R}_{ik}\|^2}_{\mathcal{L}_{\text{PDE}}} \tag{60}$$

where the residual is:

$$\mathcal{R}_{ik} = \left.\frac{\partial \boldsymbol{y}_\theta}{\partial \zeta}\right|_{\zeta_k} - \Delta z \cdot \boldsymbol{f}(\boldsymbol{y}_\theta(\zeta_k), z_k; q/p_i) \tag{61}$$

8.3 RK-PINN

$$\mathcal{L}_{\text{RK-PINN}} = \mathcal{L}_{\text{data}} + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}} + \lambda_{\text{PDE}} \sum_{s=1}^{4} \mathcal{L}_{\text{PDE}}^{(s)} \tag{62}$$

9 Implementation Notes

9.1 Critical Constants

$$c_{\text{light}} = 2.99792458 \times 10^{-4} \quad [\text{mm/ns} \times \text{unit conversions}] \tag{63}$$

This constant converts:

$$\kappa = \frac{q}{p} \cdot c_{\text{light}} \quad [\text{MeV}^{-1}\ \text{T}^{-1}\ \text{mm}^{-1}] \tag{64}$$

9.2 Normalization Factor

The path length factor:

$$N = \sqrt{1 + t_x^2 + t_y^2} = \frac{|\boldsymbol{p}|}{p_z} = \frac{ds}{dz} \tag{65}$$

accounts for the fact that the particle travels a distance $ds = N \cdot dz$ when advancing by dz in z.

9.3 Dominant Field Component

In LHCb, B_y dominates, causing bending in the x-z plane:

$$\frac{dt_x}{dz} \approx -\kappa \cdot N \cdot (1 + t_x^2) \cdot B_y \tag{66}$$

For small slopes ($t_x, t_y \ll 1$) and positive particles ($q > 0$) in a negative field ($B_y < 0$):

$$\frac{dt_x}{dz} > 0 \quad \Rightarrow \quad \text{track bends to positive } x \tag{67}$$

10 Conclusion

We have derived the complete mathematical framework for neural network-based track extrapolation:

1. **MLP**: Learns the extrapolation mapping implicitly from data
2. **PINN**: Enforces Lorentz force equations via automatic differentiation
3. **RK-PINN**: Combines RK structure with learnable physics constraints

The key physics are encoded in Equations (7)–(8), with the critical constant $c_{\text{light}} = 2.99792458 \times 10^{-4}$.