MTH/CSC 4150
Fall 2025
Draft Project Report: Mathematically Modeling a Distance Runner's Race with Interpolation

Daron Baltazar, Madi Price, Evan Dant

Tuesday, November 11th

1 Introduction

Distance runners and coaches frequently analyze split times recorded at specific checkpoints in a race. These discrete measurements provide partial information about the race, but the runner's pace, velocity, and acceleration change continuously between checkpoints. The goal of this project is to reconstruct a smooth approximation of the runner's motion from split data using interpolation and to use numerical differentiation to study pacing behavior throughout the race.

We model the runner's cumulative distance as a function of time, denoted by $s(t)$. From this interpolated position function, we approximate the instantaneous velocity $v(t)$ and acceleration $a(t)$. We compare several interpolation approaches studied in class, including Lagrange and Newton interpolating polynomials and natural cubic splines. We then examine how the choice of method influences the estimates of velocity and acceleration and how well each method captures realistic pacing patterns.

2 Theoretical Background

2.1 Polynomial Interpolation

Suppose there are $n + 1$ data points (t_i, s_i) with distinct time values $t_0 < t_1 < \cdots < t_n$. An interpolating polynomial $p_n(t)$ of degree n that satisfies

$$p_n(t_i) = s_i, \qquad i = 0, 1, \ldots, n. \tag{1}$$

There exists a unique polynomial of at most n that satisfies the equation.

In the Lagrange form,

$$p_n(t) = \sum_{i=0}^{n} s_i L_i(t), \tag{2}$$

where the Lagrange basis polynomials are

$$L_i(t) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{t - t_j}{t_i - t_j}, \qquad i = 0, 1, \ldots, n. \tag{3}$$

This basis is interpolatory, since $L_i(t_j) = \delta_{ij}$.

In Newton's form, an alternative basis is used:

$$p_n(t) = a_0 + a_1(t - t_0) + a_2(t - t_0)(t - t_1) + \cdots + a_n \prod_{k=0}^{n-1}(t - t_k), \tag{4}$$

where the coefficients a_i are computed from divided differences. Newton's form is more convenient for computation and for updating the polynomial when additional data points are added.

2.2 Interpolation Error

Let $f(t)$ be a sufficiently smooth function and let $p_n(t)$ be the degree-n interpolating polynomial constructed from $n+1$ distinct nodes $t_0, \ldots, t_n$. The interpolation error can be written as

$$f(t) - p_n(t) = \frac{f^{(n+1)}(c)}{(n+1)!} \prod_{i=0}^{n}(t - t_i), \tag{5}$$

for some c in the interval containing the nodes and the point t. This formula shows that the error depends on both the spacing of the nodes and the (n+1) derivative of f. When the nodes are equally spaced and n is large, the product term can lead to large oscillations near the ends of the interval (Runge's phenomenon).

2.3 Cubic Splines

To reduce the oscillations associated with high-degree polynomials, we use cubic splines. Given $n+1$ knots $t_0 < t_1 < \cdots < t_n$ and function values $s_i = f(t_i)$, a cubic spline S(t) of the form,

$$S(t) = \begin{cases} S_0(t), & t \in [t_0, t_1], \\ S_1(t), & t \in [t_1, t_2], \\ \vdots \\ S_{n-1}(t), & t \in [t_{n-1}, t_n], \end{cases} \tag{6}$$

where each $S_i(t)$ is a cubic polynomial

$$S_i(t) = a_i + b_i(t - t_i) + c_i(t - t_i)^2 + d_i(t - t_i)^3. \tag{7}$$

The spline is constructed so that

1. $S(t_i) = s_i$ for $i = 0, \dots, n$ (interpolation),
2. $S(t)$ is continuously differentiable, so $S'(t)$ is continuous at each interior knot $t_1, \dots, t_{n-1}$,
3. $S''(t)$ is continuous at each interior knot.

These conditions yield $4n - 2$ equations. Two additional conditions are needed at the endpoints. In this project, we use the *natural cubic spline*, which imposes

$$S''(t_0) = 0, \qquad S''(t_n) = 0. \tag{8}$$

Solving the resulting linear system produces the coefficients a_i, b_i, c_i, d_i for all subintervals.

2.4 Numerical Differentiation

Once a smooth interpolating function $S(t)$ is available, the velocity and acceleration of the runner are given by

$$v(t) = S'(t), \qquad a(t) = S''(t). \tag{9}$$

For spline models, derivatives can be computed analytically by differentiating the cubic expressions, or numerically using finite difference stencils.

For comparison, numerical differentiation formulas can also be derived directly from discrete data. For example, the centered difference approximation for the first derivative at an interior node is

$$f'(t_i) \approx \frac{f(t_{i+1}) - f(t_{i-1})}{2h}, \tag{10}$$

where $h = t_{i+1} - t_i$ is the time step. This stencil is second order accurate in h.

3 Data and Preprocessing

The data for this project consist of split times and cumulative distances from a single distance runner's race. For example, in a 5k race, splits may be recorded every 1 km, while in a 10k race splits may be recorded every mile. Let t_i denote the elapsed time at the ith checkpoint and let s_i denote the cumulative distance at this time.

We perform the following preprocessing steps:

1. Convert all distances to meters and all times to seconds.
2. Shift the time axis so that $t_0 = 0$ at the race start.
3. Remove any obvious measurement errors such as negative time differences between consecutive splits.

The resulting dataset $\{(t_i, s_i)\}_{i=0}^{n}$ serves as the input to all interpolation methods.

4 Numerical Methods and Implementation

4.1 Interpolation Models

We implement and compare the following models.

1. **Polynomial interpolation.** We construct the Lagrange and Newton interpolating polynomials using custom MATLAB functions based on (2)–(4). This approach provides a baseline but is expected to be sensitive to node spacing and prone to oscillations.

2. **Natural cubic spline.** We use MATLAB's `spline` command to construct the natural cubic spline that satisfies (8). We also write helper functions that evaluate $S(t)$, $S'(t)$, and $S''(t)$ using the spline coefficients.

4.2 Velocity and Acceleration

For each interpolant $S(t)$, we compute the velocity and acceleration by differentiating the spline pieces analytically. For a cubic

$$S_i(t) = a_i + b_i(t - t_i) + c_i(t - t_i)^2 + d_i(t - t_i)^3,$$

the derivatives are

$$S_i'(t) = b_i + 2c_i(t - t_i) + 3d_i(t - t_i)^2, \tag{11}$$

$$S_i''(t) = 2c_i + 6d_i(t - t_i). \tag{12}$$

Evaluating (11) and (12) on a fine time grid yields approximate velocity and acceleration profiles for the runner.

4.3 Accuracy Checks

We validate the interpolants in one main way.

1. **Checkpoint consistency.** For each method, we verify that the interpolant reproduces the input distances at the original split times, that is $S(t_i) \approx s_i$.

5 Results

5.1 Position vs. Time

The first plot compares the Newton interpolating polynomial and the natural cubic spline against the original split data. Both interpolants pass exactly through the data points, and they are almost indistinguishable over most of the race. Near the final split, the polynomial begins to bend more sharply than the spline, while the spline stays slightly smoother.

5.2 Velocity and Acceleration Profiles

Differentiating each interpolant with the centered finite-difference formula gives the velocity curves. Both show an initial acceleration phase, a roughly steady middle section, and a decrease toward the end of the race. However, the polynomial-based velocity reaches larger peaks and drops off more steeply near the finish, while the spline-based velocity changes more gradually.

The second derivative plots show the same pattern more clearly. The spline acceleration is small and slowly varying for most of the race, consistent with small adjustments in pace. The polynomial acceleration shows larger positive and negative swings, especially near the final split, which do not correspond to any obvious features in the original data.

6 Discussion

Because the Newton interpolating polynomial is a single degree-5 polynomial through all six points, small changes near the end of the interval affect the curve everywhere. This is the same phenomenon discussed in the interpolation error and Runge examples: high-degree polynomials with equally spaced nodes can oscillate, especially near the boundaries.

The natural cubic spline instead uses a different cubic on each subinterval and enforces continuity of the function, first derivative, and second derivative. This local, piecewise structure explains why the spline trajectory and its derivatives are smoother and more physically reasonable, even though both methods agree exactly at the split times.

The derivative plots show that the spline model leads to "nice" pacing behavior (smooth changes in speed and moderate accelerations), while the polynomial model suggests unrealistically large decelerations near the end. This supports the conclusion that spline interpolation is better suited than a single polynomial for modeling a runner's race from discrete splits.

7 Conclusion and Future Work

This project demonstrates that spline interpolation is a useful tool for modeling a distance runner's race based on discrete split data. Among the methods tested, natural cubic spline interpolation produces the most realistic pacing curves. High-degree polynomials, while theoretically valid interpolants, are not practical for this application due to numerical instability and spurious oscillations.

Future work could extend the current model by incorporating additional race information, such as elevation changes, or by exploring numerical integration methods once those techniques are introduced in class.

References

1. T. Sauer, *Numerical Analysis*, 3rd ed., Pearson, 2019.

2. Course notes and lecture slides from MTH/CSC 4150, Fall 2025.