Functional Data Analysis

Vadim Tyuryaev

Department of Mathematics & Statistics \ York University, Toronto, Canada

Slide 1a: What Is FDA?

- *"Functional data analysis (FDA), a repertoire of statistical methods that considers data as evaluations of curves (mathematical functions) over a discrete grid"* (Xu 2020).
- Rather than being a scalar or a vector, each observation i is a function, $x_i(t)$, defined over a domain $\mathcal{T} \subset \mathbb{R}$.
- A critical assumption in this context is the **smoothness** of the underlying function, $x(t)$. Should this smoothness condition not hold, adopting a functional data framework over a conventional multivariate analysis is not sensible (J. O. Ramsay and Silverman 2005).
- Many FDA methods are alterations of classical multivariate methods such as principal components analysis (PCA) and linear models (Müller 2022; J. O. Ramsay and Silverman 2001).

Slide 1b: Goals and Distinctions of FDA

- The goals (J. O. Ramsay and Silverman 2005) of functional data analysis align with classical statistical objectives:
  - Represent and transform data to enable further investigation.
  - Visualize data to highlight key patterns and dynamics over given domain.
  - Characterize and interpret sources of systematic variation among the data.
  - Model and explain variation in a response using functional predictors or covariates.

- FDA relies on:
  - Hilbert space theory.
  - Basis function expansions (e.g., Generalized Fourier Series, B-splines).
  - Regularization techniques to prevent overfitting.

Slide 2a: Hilbert Space and Fourier Series

- A **Hilbert space** (Blanchard and Brüning 2015), denoted as $\mathcal{H}$, is a real or complex vector space that is also a complete inner product space.

- Key concepts:
  - **Inner product**: $\langle f, g \rangle = \int_a^b f(t)g(t)dt$
  - **Norm**: $\|f\| = \sqrt{\langle f, f \rangle}$
  - **Distance** between **f** and **g**: $\|f - g\|$
  - **Orthonormal basis**: $\{\phi_k\}$ satisfying $\langle \phi_i, \phi_j \rangle = \delta_{ij}$, where δ_{ij} is the Kronecker delta.
  - **Generalized Fourier Series**: $f = \sum_{k=0}^{\infty} c_k \phi_k$ and $c_k = \langle f, \phi_k \rangle$

- In FDA, each function $x_i(t)$ is modeled as an element of $L^2([a, b])$, the space of **square-integrable functions**:

$$L^2([a, b]) = \{f : \int_a^b |f(t)|^2 dt < \infty\}$$

Slide 2b (Example): $f(t) = t + \cos(t)$ on $[-\pi, \pi]$

- To check whether $f(t) = t + \cos(t)$ belongs to $L^2([-\pi, \pi])$:

$$\int_{-\pi}^{\pi} |f(t)|^2 \, dt = \int_{-\pi}^{\pi} (t + \cos(t))^2 \, dt$$

Expand the square:

$$\int_{-\pi}^{\pi} (t + \cos(t))^2 \, dt = \int_{-\pi}^{\pi} \left(t^2 + 2t\cos(t) + \cos^2(t)\right) \, dt$$

Break into parts:

$$\int_{-\pi}^{\pi} t^2 \, dt = \frac{t^3}{3}\Bigg|_{-\pi}^{\pi} = \frac{2\pi^3}{3}$$

$$\int_{-\pi}^{\pi} 2t\cos(t) \, dt = [2t\sin(t)]\Big|_{-\pi}^{\pi} - 2\int_{-\pi}^{\pi} \sin(t) \, dt = 0$$

$$\int_{-\pi}^{\pi} \cos^2(t) \, dt = \int_{-\pi}^{\pi} \frac{1 + cos(2t)}{2} = \pi$$

Therefore:

$$\int_{-\pi}^{\pi} |f(t)|^2 \, dt = \frac{2\pi^3}{3} + \pi = \frac{2\pi^3 + 3\pi}{3} < \infty \Rightarrow f \in L^2([-\pi, \pi])$$

Slide 2c: Orthogonal Basis

- The standard Fourier basis is $\{1, cos(t), sin(t), cos(2t), sin(2t), \ldots\}$:

$$\phi_0(t) = 1, \quad \phi_{2k-1}(t) = \cos(kt), \quad \phi_{2k}(t) = \sin(kt)$$

- The basis forms a complete orthogonal system over $[-\pi, \pi]$:

$$\langle \phi_m, \phi_n \rangle = \int_{-\pi}^{\pi} \phi_m(t)\, \phi_n(t)\, dt = \begin{cases} 0, & \text{if } m \neq n \\ \pi, & \text{if } m = n > 0 \\ 2\pi, & \text{if } m = n = 0 \end{cases}$$

- The above is based on the following integral identities (Weisstein 2002):

$$\int_{-\pi}^{\pi} \sin(mt) \sin(nt)\, dt = \pi\delta_{mn}, \quad \int_{-\pi}^{\pi} \cos(mt) \cos(nt)\, dt = \pi\delta_{mn}$$

$$\int_{-\pi}^{\pi} \sin(mt) \cos(nt)\, dt = 0, \quad \int_{-\pi}^{\pi} \sin(mt)\, dt = 0, \quad \int_{-\pi}^{\pi} \cos(mt)\, dt = 0$$

where δ_{mn} is the Kronecker delta and $m, n \neq 0$.

Slide 2d: Generalized Fourier Series

- Thus, any periodic $f \in L^2([-\pi, \pi])$ can be decomposed as:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nt) + b_n \sin(nt)]$$

with Fourier coefficients calculated in the following manner:

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \cos(nt)\, dt \quad (n = 0, 1, 2, \ldots)$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \sin(nt)\, dt \quad (n = 1, 2, 3, \ldots)$$

- Any periodic $g \in L^2([-l, l])$ can be expressed in as:

$$g(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos\left(\frac{k\pi t}{l}\right) + b_k \sin\left(\frac{k\pi t}{l}\right)\right)$$

the the Fourier coefficients which are given by the formulas:

$$a_k = \frac{1}{l} \int_{-l}^{l} f(t) \cos\left(\frac{k\pi t}{l}\right)\, dt \quad (k = 0, 1, 2, \ldots)$$

$$b_k = \frac{1}{l} \int_{-l}^{l} f(t) \sin\left(\frac{k\pi t}{l}\right)\, dt \quad (k = 1, 2, 3 \ldots)$$

Slide 2e: First Fourier Coefficients of $f(t) = t + \cos(t)$

- Zeroth coefficient:

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\,dt = \frac{1}{\pi}\left[\int_{-\pi}^{\pi} t\,dt + \int_{-\pi}^{\pi} \cos(t)\,dt\right] = 0$$

- First cosine coefficient:

$$a_1 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\cos(t)\,dt = \frac{1}{\pi}\left[\int_{-\pi}^{\pi} t\cos(t)\,dt + \int_{-\pi}^{\pi} \cos^2(t)\,dt\right] =$$
$$= \frac{1}{\pi}\left[0 + \pi\right] = 1$$

- First sine coefficient:

$$b_1 = \frac{1}{\pi}\left[\int_{-\pi}^{\pi} t\sin(t)\,dt + \int_{-\pi}^{\pi} \cos(t)\sin(t)\,dt\right] =$$
$$= \frac{1}{\pi}\left[2\pi + 0\right] = 2$$

Slide 2f: Second Fourier Coefficients of $f(t) = t + \cos(t)$

- Second cosine coefficient:

$$a_2 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\cos(2t)\,dt =$$

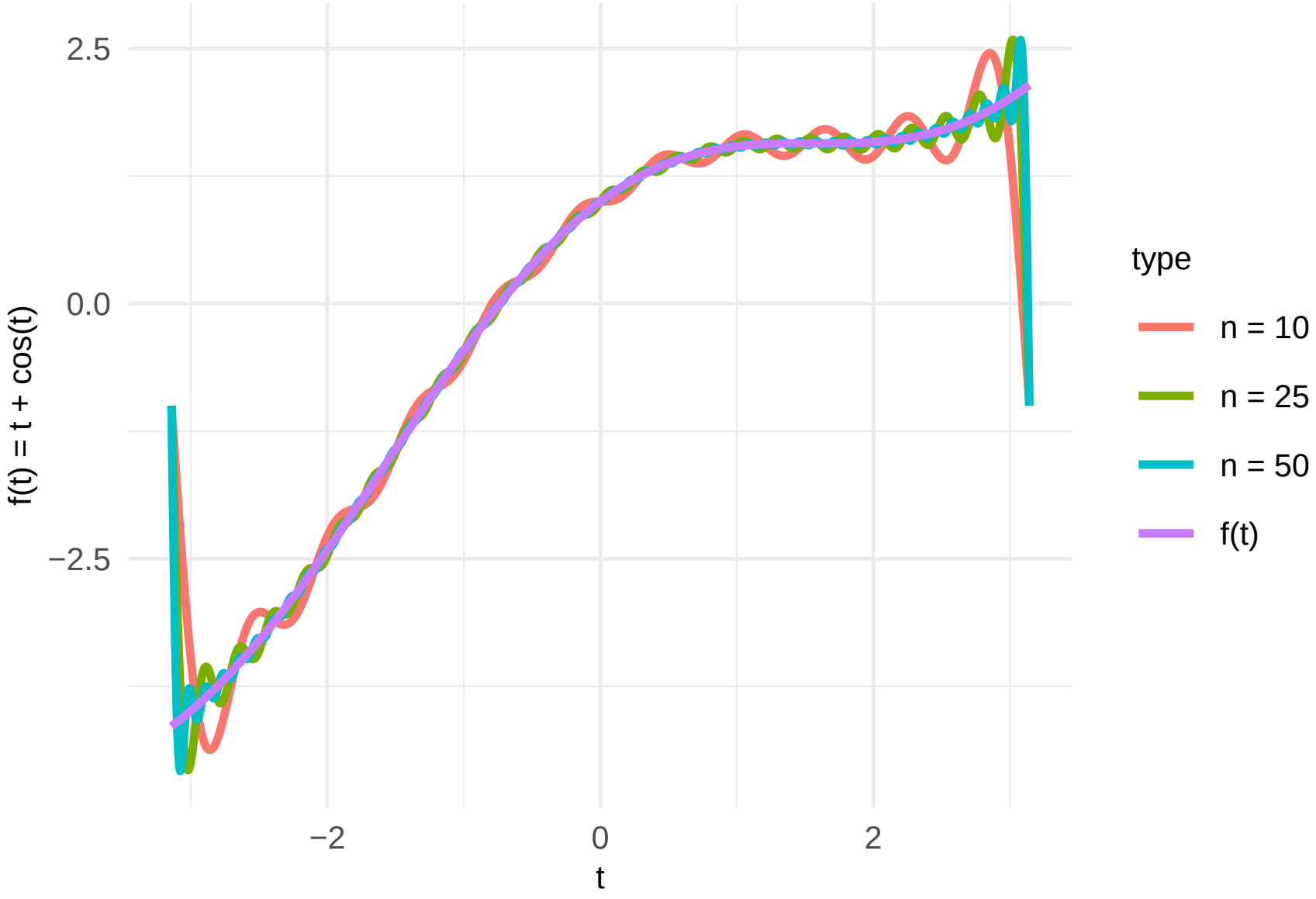
$$\frac{1}{\pi}\left[\int_{-\pi}^{\pi} t\cos(2t)\,dt + \int_{-\pi}^{\pi}\cos(t)\cos(2t)\,dt\right] = \frac{1}{\pi}\left[0+0\right] = 0$$

- Second sine coefficient:

$$b_2 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(t)\sin(2t)\,dt =$$

$$\frac{1}{\pi}\left[\int_{-\pi}^{\pi} t\sin(2t)\,dt + \int_{-\pi}^{\pi}\cos(t)\sin(2t)\,dt\right] = \frac{1}{\pi}\left[-\pi+0\right] = -1$$

Slide 2g (Example): Computational Approximation of $f(t)$



Slide 3a: Interpolation vs. Smoothing

- **Interpolation**: fit f such that $f(t_i) = y_i$
- **Smoothing**: fit f such that

$$\min_f \sum_{i=1}^{n} (y_i - f(t_i))^2 + \lambda \int (f''(t))^2 dt$$

- Why add regularization term?
  - Large values of $f''(t)$ indicate rapid curvature or oscillation.
  - Penalizing $\int (f''(t))^2 dt$ discourages overfitting to noise and promotes smoothness.
  - Larger λ results in smoother fits.

Slide 3b: Lagrange Polynomials

- Given $n+1$ distinct interpolation points $t_0, t_1, \ldots, t_n$ and function values $y_0, y_1, \ldots, y_n$, the **Lagrange interpolation polynomial** $P_n(t)$ is defined by (Gautschi 2011):

$$P_n(t) = \sum_{j=0}^{n} y_j \ell_j(t)$$

where the **Lagrange basis polynomials** $\ell_j(t)$ are:

$$\ell_j(t) = \prod_{\substack{0 \le m \le n \\ m \neq j}} \frac{t - t_m}{t_j - t_m}$$

- Each $\ell_j(t)$ satisfies:
  - $\ell_j(t_k) = \delta_{jk}$ (Kronecker delta)
  - $\deg(\ell_j) = n$

Slide 3c: Lagrange Polynomial for $f(t) = 3e^{-t^2}$

We choose three nodes: $t_0 = -2,\ t_1 = 0,\ t_2 = 2$

The interpolating polynomial of degree $n = 2$ is defined as:

$$P_2(t) = \sum_{i=0}^{2} f(t_i) \cdot \ell_i(t)$$

Lagrange basis polynomials:

$$\ell_0(t) = \frac{(t - t_1)(t - t_2)}{(t_0 - t_1)(t_0 - t_2)} = \frac{(t)(t-2)}{(-2)(-2-2)} = \frac{t(t-2)}{8}$$

$$\ell_1(t) = \frac{(t - t_0)(t - t_2)}{(t_1 - t_0)(t_1 - t_2)} = \frac{(t-(-2))(t-2)}{(0-(-2))(0-2)} = -\frac{(t+2)(t-2)}{4}$$

$$\ell_2(t) = \frac{(t - t_0)(t - t_1)}{(t_2 - t_0)(t_2 - t_1)} = \frac{(t-(-2))(t)}{(2-(-2))(2)} = \frac{t(t+2)}{8}$$

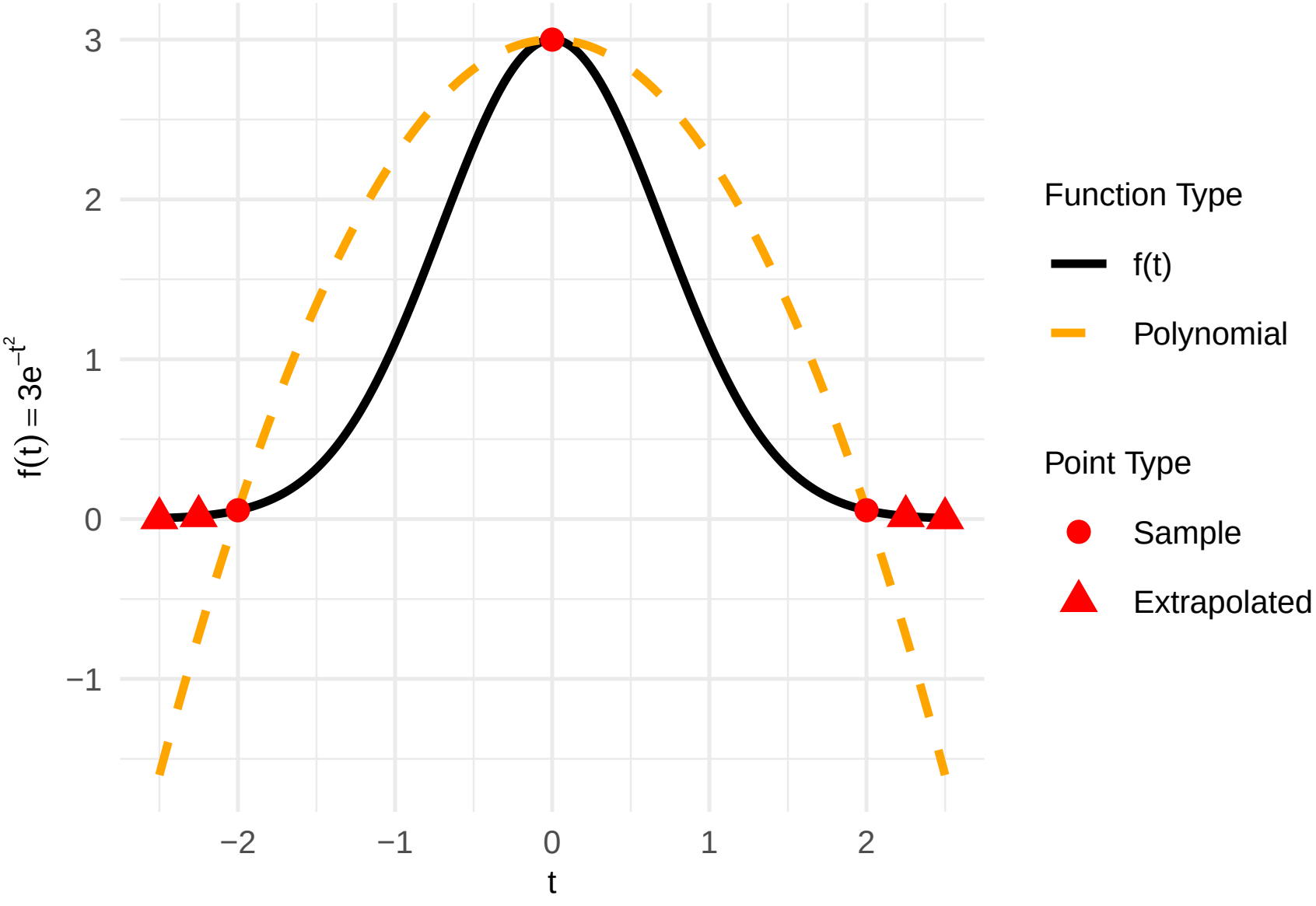
Function values:

$$f(-2) = 3e^{-4} = e \approx 0.054947, \quad f(0) = 3, \quad f(2) = 3e^{-4} \approx 0.054947$$

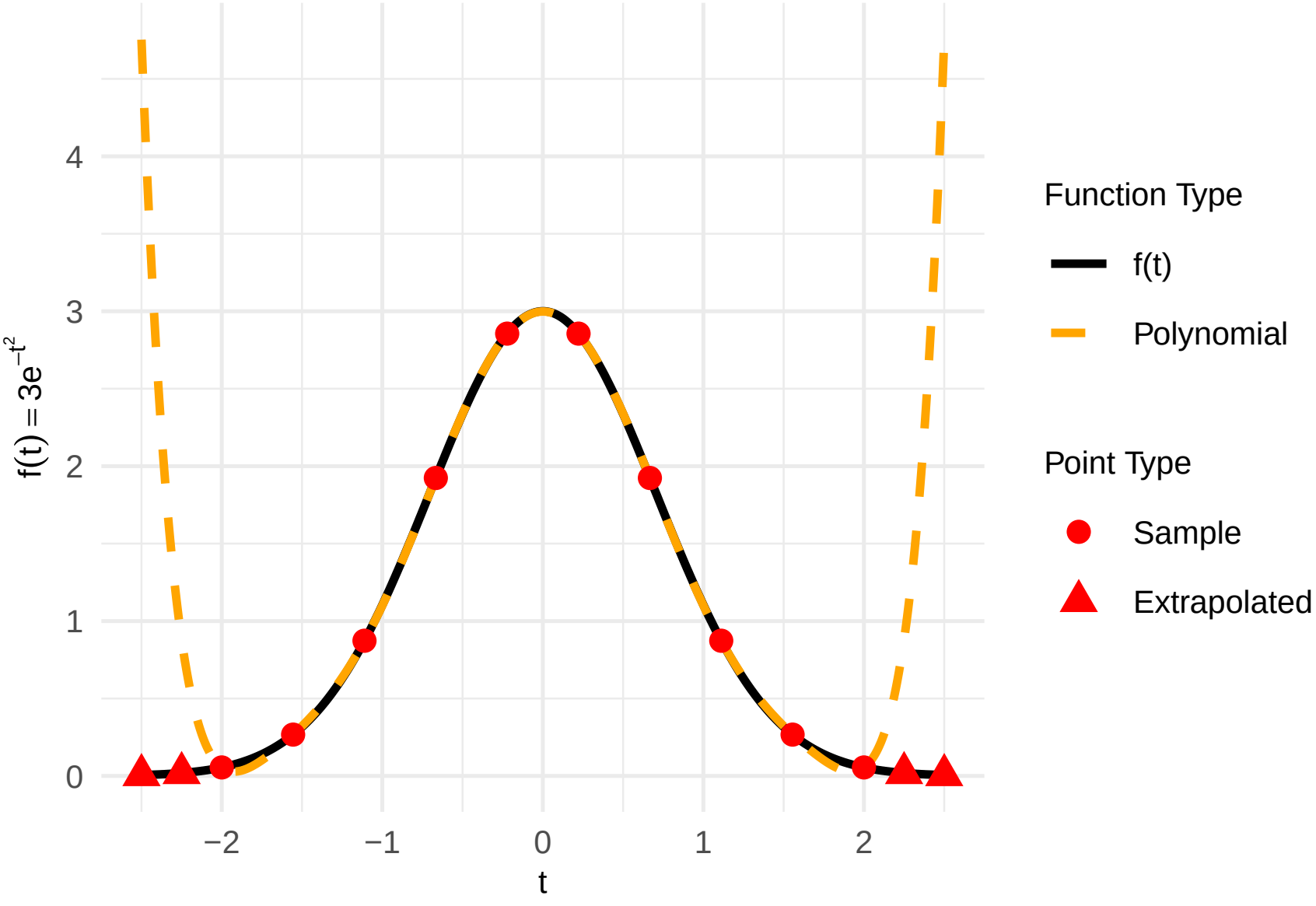
Interpolating polynomial:

$$P(t) = 3e^{-4} \cdot \frac{t(t-2)}{8} - 3\frac{(t+2)(t-2)}{4} + 3e^{-4} \cdot \frac{t(t+2)}{8}$$

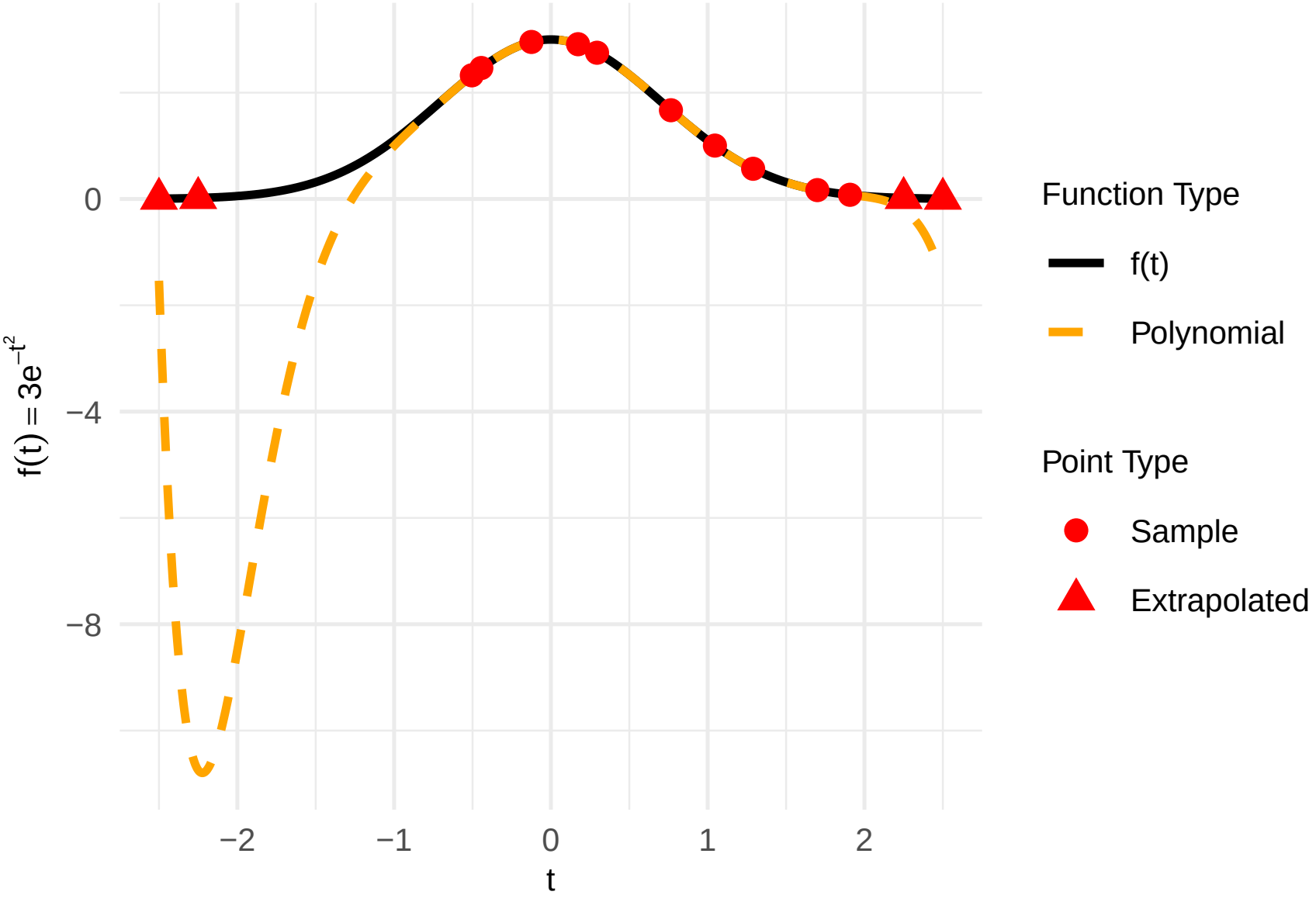
Slide 3d: $P_2(t)$ at t = -2,0,2



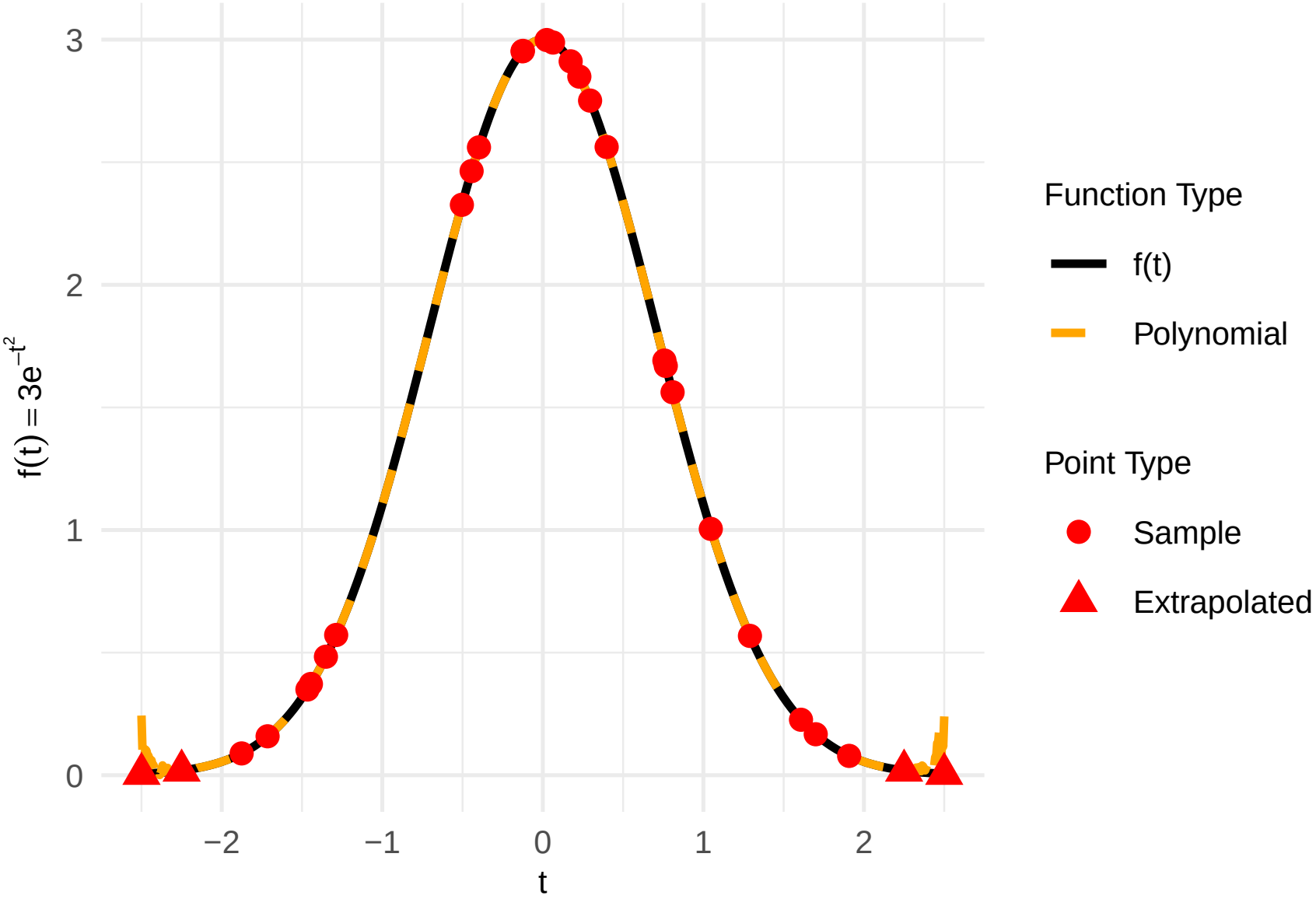
Slide 3e: $P_9(t)$ Using Ten Evenly Space Points



Slide 3f: $P_9(t)$ Using Ten Randomly Selected Points



Slide 3g: $P_{19}(t)$ Using Twenty Randomly Selected Points



Slide 3h: From Monomials to Basis Splines

- Lagrange polynomials and similar techniques focus on the approximation of a function over a finite interval, $[a, b]$, using a single polynomial (monomial).
- This method guarantees that for functions with sufficient smoothness, the **approximation error** can be made **arbitrarily small** by selecting a polynomial of a sufficiently **high degree**.
- An alternative approach to controlling accuracy is partitioning the interval $[a, b]$ into a series of subintervals:

$$a = x_1 < x_2 < x_3 < \cdots < x_{n-1} < x_n = b$$

- The function is approximated on each subinterval $[x_i, x_{i+1}]$ (for $i = 1, 2, \ldots, n-1$) using a separate, **low-degree polynomial**.

Slide 4a: Theory

- A **B-spline** (Gautschi 2011; Patrikalakis, Maekawa, and Cho 2009) is a piecewise polynomial with local support, defined over knots $\{\tau_i\}$.
- B-splines $B_{j,m}(t)$ of degree m satisfy:
  - Continuity up to $(m-1)$ derivatives.
  - Localized influence between τ_j and τ_{j+m+1}.
- Local support (influence) means that the basis function $B_{j,m}(t)$ is non-zero only on the interval $[\tau_j, \tau_{j+m+1})$
- Functional representation:

$$f(t) = \sum_{j=1}^{n} c_j B_{j,m}(t)$$

  with coefficients $c_j \in \mathbb{R}$ estimated from data.
- Because each control point c_j is paired with a basis function $B_{j,m}(t)$, the **influence** of that control point is confined to the interval where its basis function is non-zero.

Slide 4b: Setup for B-spline Approximation

Let's say we want to approximate $f(t)$ on $[-2, 2]$ using 2nd-degree B-splines.

- Degree $m = 2$, Order $k = 3$
- Divide $[-2, 2]$ into 3 equal subintervals:

$$[-2, -0.67],\ [-0.67, 0.67],\ [0.67, 2]$$

- Internal breakpoints: $\xi = \{-0.67,\ 0.67\}$

Construct a **clamped knot vector** by repeating boundary knots:

$$\tau = [-2, -2, -2, -0.67, 0.67, 2, 2, 2]$$

Number of B-spline basis functions:

$$n = \text{length}(\tau) - k = 8 - 3 = 5$$

We will now compute several examples of basis functions $B_{j,m}(t)$ for $j = 1, \ldots, 5$ and $m = 0, \ldots, 2$.

Slide 4c: Zeroth-Degree B-splines $B_{j,0}(t)$

Defined as:

$$B_{j,0}(t) = \begin{cases} 1, & \tau_j \leq t < \tau_{j+1} \\ 0, & \text{otherwise} \end{cases}$$

With knots $\tau = [-2, -2, -2, -0.67, 0.67, 2, 2, 2]$, the non-zero supports are:

- $B_{3,0}(t) = 1$ if $-2 < t < -0.67$
- $B_{4,0}(t) = 1$ if $-0.67 < t < 0.67$
- $B_{5,0}(t) = 1$ if $0.67 < t < 2$

Note: $B_{1,0}$ and $B_{2,0}$ are zero because $\tau_1 = \tau_2 = \tau_3 = -2$ are degenerate intervals. So are $B_{6,0}$ and $B_{7,0}$.

Slide 4d: Recursive formula

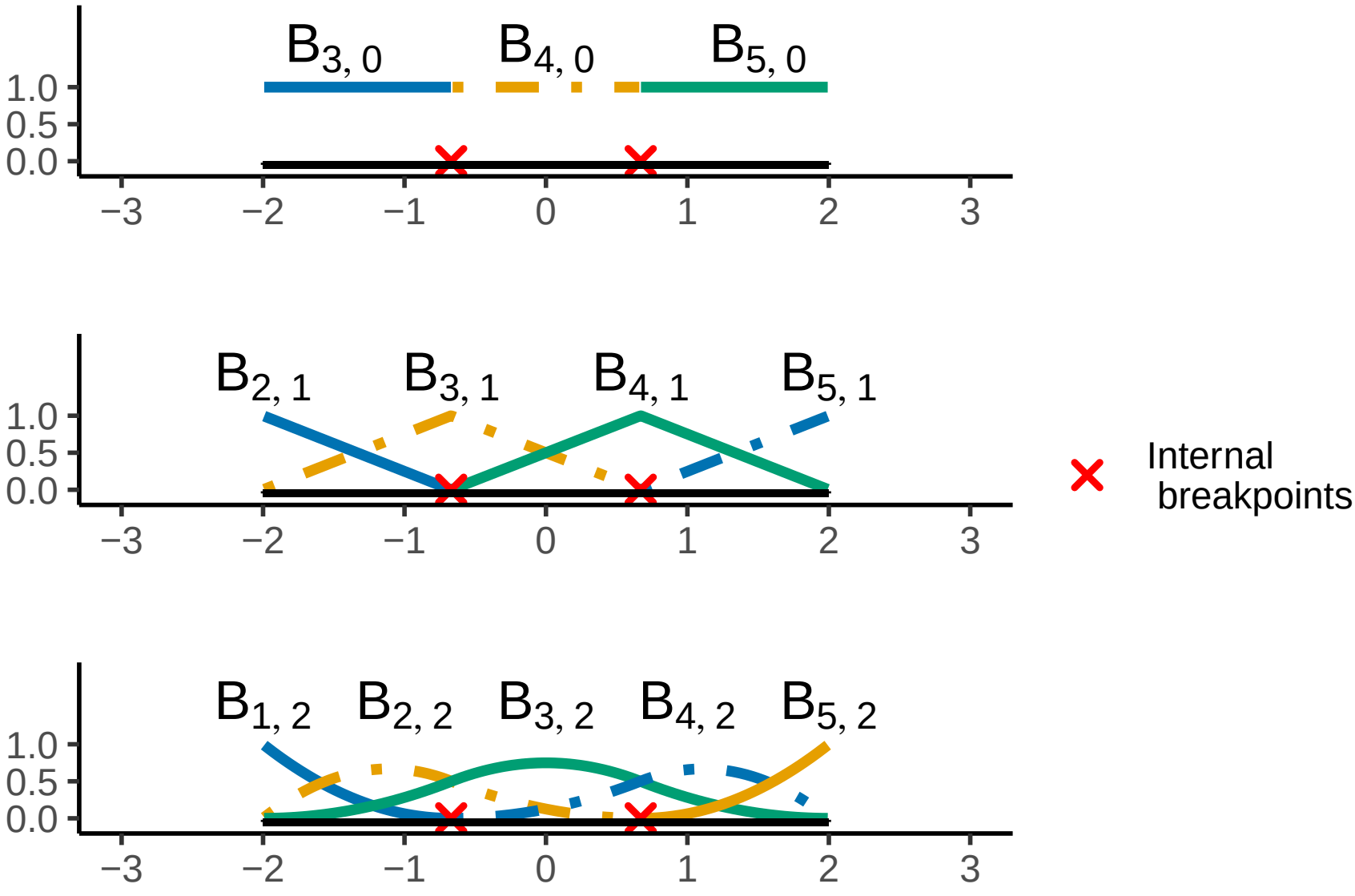
- From the 0-th degree B-splines, higher degree B-splines can be obtained via recurrence:

$$B_{j,k}(t) = \frac{t - \tau_j}{\tau_{j+k} - \tau_j} B_{j,k-1}(t) + \frac{\tau_{j+k+1} - t}{\tau_{j+k+1} - \tau_{j+1}} B_{j+1,k-1}(t)$$

- This construction, while **far from obvious**, ensures that $B_{j,k}$ has one more continuous derivative than does $B_{j,k-1}$.

- Therefore, while $B_{j,0}$ is discontinuous, $B_{j,1}$ is continuous, $B_{j,2} \in C^1(\mathbb{R})$, and $B_{j,3} \in C^2(\mathbb{R})$.

Slide 4e: B-Splines Construction Visualiation



Slide 4f: First-Degree B-splines

Use:

$$B_{j,1}(t) = \frac{t - \tau_j}{\tau_{j+1} - \tau_j} B_{j,0}(t) + \frac{\tau_{j+2} - t}{\tau_{j+2} - \tau_{j+1}} B_{j+1,0}(t)$$

to construct $B_{3,1}(t)$ and $B_{4,1}(t)$

$$\tau_3 = -2,\ \tau_4 = -0.67,\ \tau_5 = 0.67$$

$$B_{3,1}(t) = \frac{t+2}{1.33} B_{3,0}(t) + \frac{0.67 - t}{1.34} B_{4,0}(t)$$

- Support:
  - First term: $t \in (-2, -0.67)$
  - Second term: $t \in (-0.67, 0.67)$

Resulting piecewise linear function (triangle) over $(-2, 0.67)$.

Slide 4g: First-Degree B-splines (continued)

$B_{4,1}(t)$:

$$\tau_4 = -0.67,\ \tau_5 = 0.67,\ \tau_6 = 2$$

$$B_{4,1}(t) = \frac{t + 0.67}{1.34} B_{4,0}(t) + \frac{2 - t}{1.33} B_{5,0}(t)$$

- Support:
  - First term: $t \in (-0.67, 0.67)$
  - Second term: $t \in (0.67, 2)$

This function also forms a triangle over $(-0.67, 2)$.

Slide 4h: Second-Degree B-spline

Now use:

$$B_{j,2}(t) = \frac{t - \tau_j}{\tau_{j+2} - \tau_j} B_{j,1}(t) + \frac{\tau_{j+3} - t}{\tau_{j+3} - \tau_{j+1}} B_{j+1,1}(t)$$

to construct $B_{3,2}(t)$:

$$B_{3,2}(t) = \frac{t+2}{2.67} \cdot B_{3,1}(t) + \frac{2-t}{2.67} \cdot B_{4,1}(t)$$

Substitute from earlier results:

$$B_{3,1}(t) = \frac{t+2}{1.33} B_{3,0}(t) + \frac{0.67 - t}{1.34} B_{4,0}(t)$$

$$B_{4,1}(t) = \frac{t+0.67}{1.34} B_{4,0}(t) + \frac{2-t}{1.33} B_{5,0}(t)$$

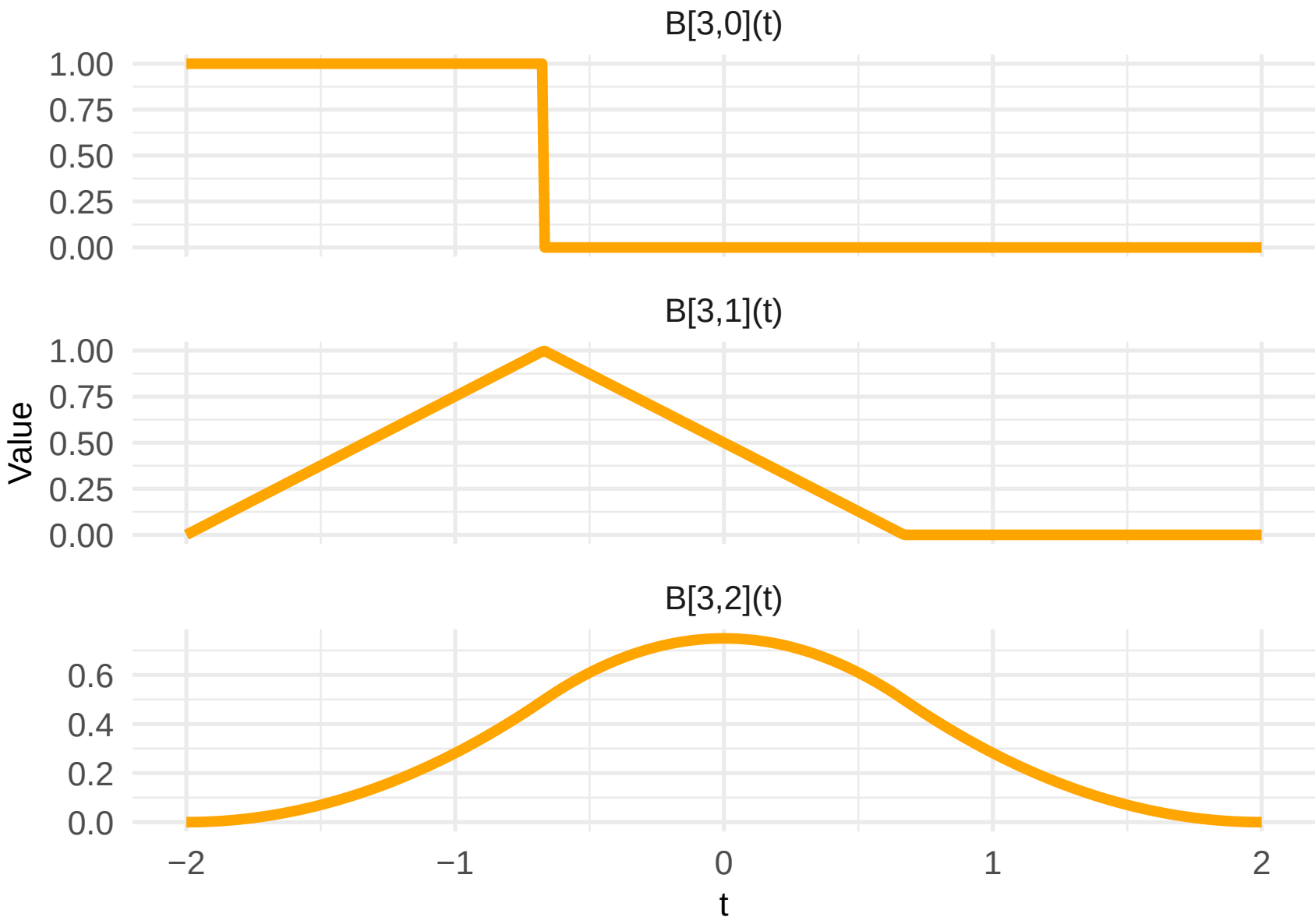
Slide 4i: Second-Degree B-spline (continued)

Putting it all together:

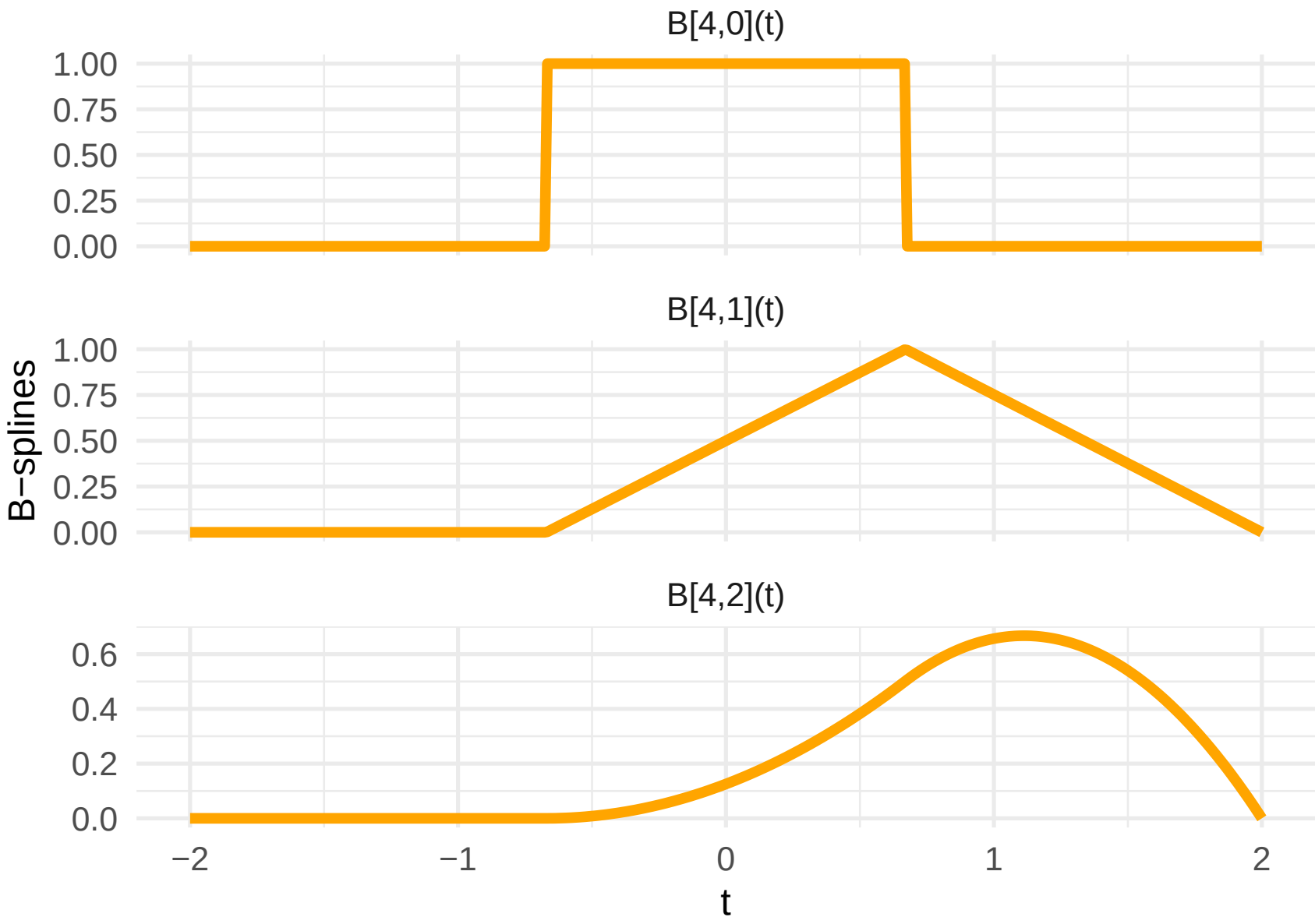
$$B_{3,2}(t) = \frac{t+2}{2.67}\left[\frac{t+2}{1.33}B_{3,0}(t) + \frac{0.67-t}{1.34}B_{4,0}(t)\right] + \frac{2-t}{2.67}\left[\frac{t+0.67}{1.34}B_{4,0}(t) + \frac{2-t}{1.33}B_{5,0}(t)\right]$$

- This yields a smooth, piecewise quadratic function over $[-2, 2]$.

Slide 4j: Constructed $B_{3,m}$ B-Splines



Slide 4k: Constructed $B_{4,m}$ B-Spline



Slide 4l: B-spline Approximation of $f(t) = 3e^{-t^2}$ on $[-2, 2]$

We approximate:

$$3e^{-t^2} \approx \sum_{j=1}^{5} c_j B_{j,2}(t)$$

The coefficients c_j are computed by solving the least squares problem:

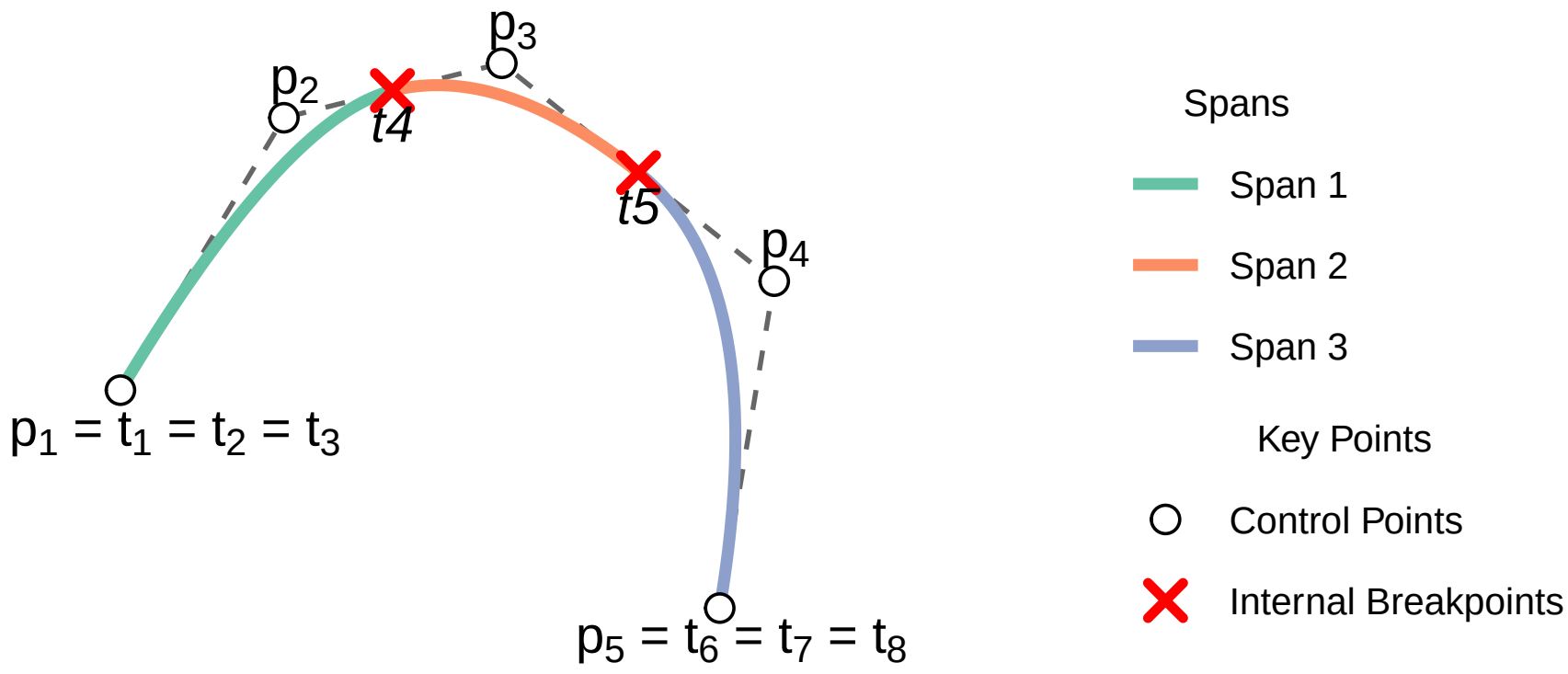
$$\min_{c_1,\ldots,c_5} \sum_{i=1}^{n} \left[3e^{-t_i^2} - \sum_{j=1}^{5} c_j B_{j,2}(t_i) \right]^2$$

With penalization, the objective becomes (note that cubic splines are required):

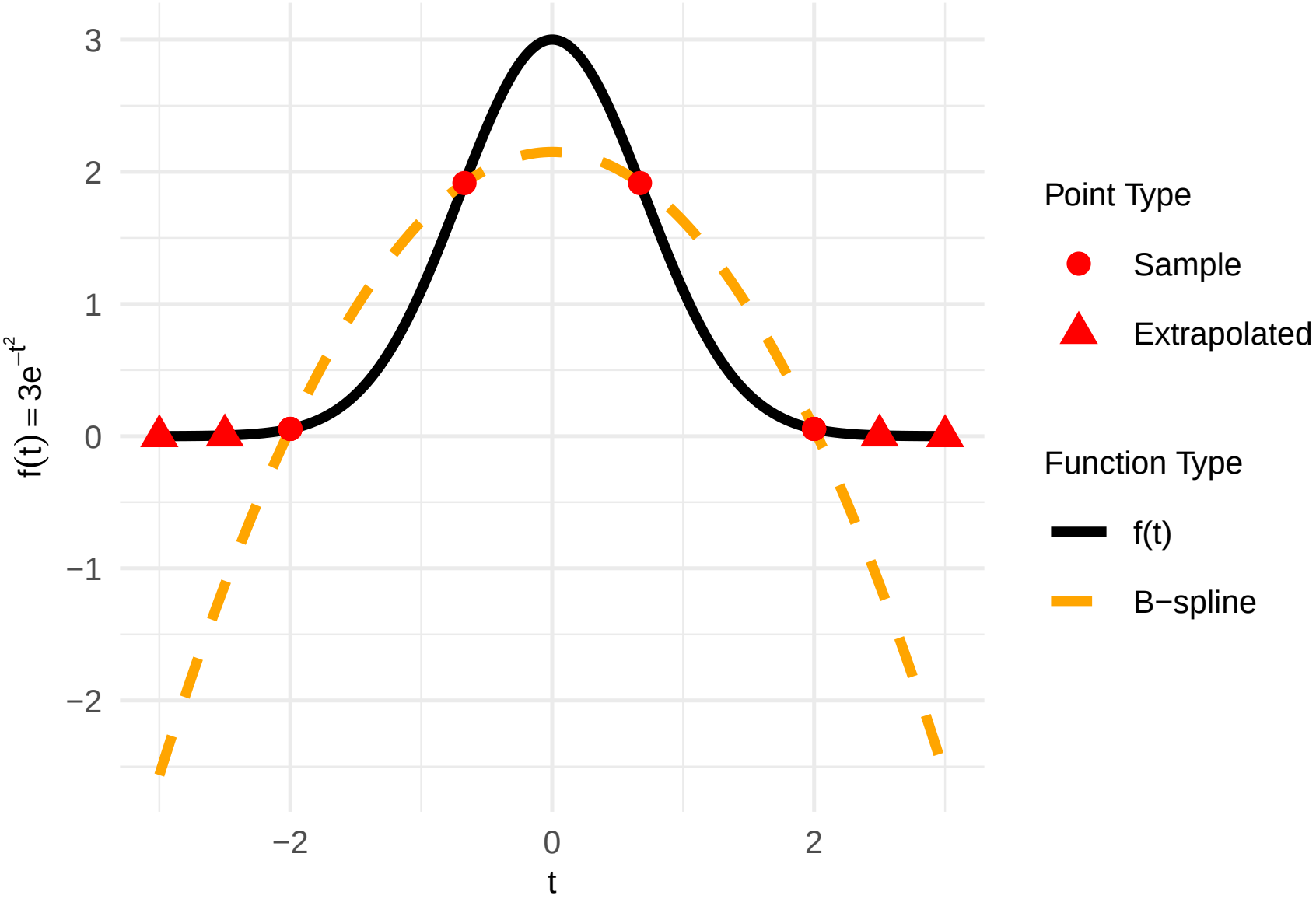
$$\min_{c_1,\ldots,c_5} \sum_{i=1}^{n} \left[3e^{-t_i^2} - \sum_{j=1}^{5} c_j B_{j,3}(t_i) \right]^2 + \lambda \int_{-2}^{2} \left(\left[\sum_{j=1}^{5} c_j B''_{j,3}(t) \right]^2 \right) dt$$

- If $\lambda = 0$: interpolation at the data points $\{t_i\}$
- If $\lambda > 0$: smoother approximation

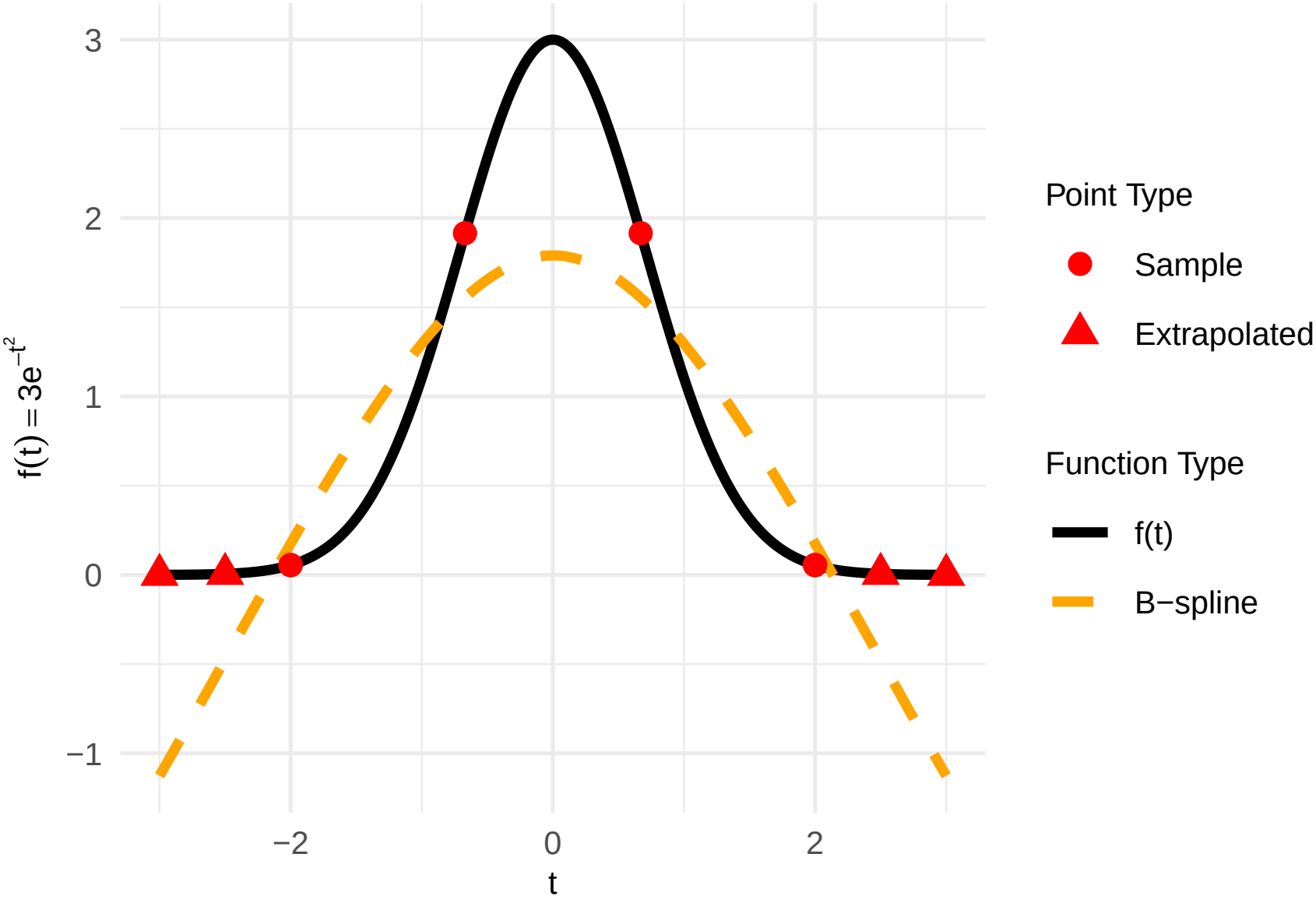
Slide 4m (Concept): A Clamped Quadratic B-spline



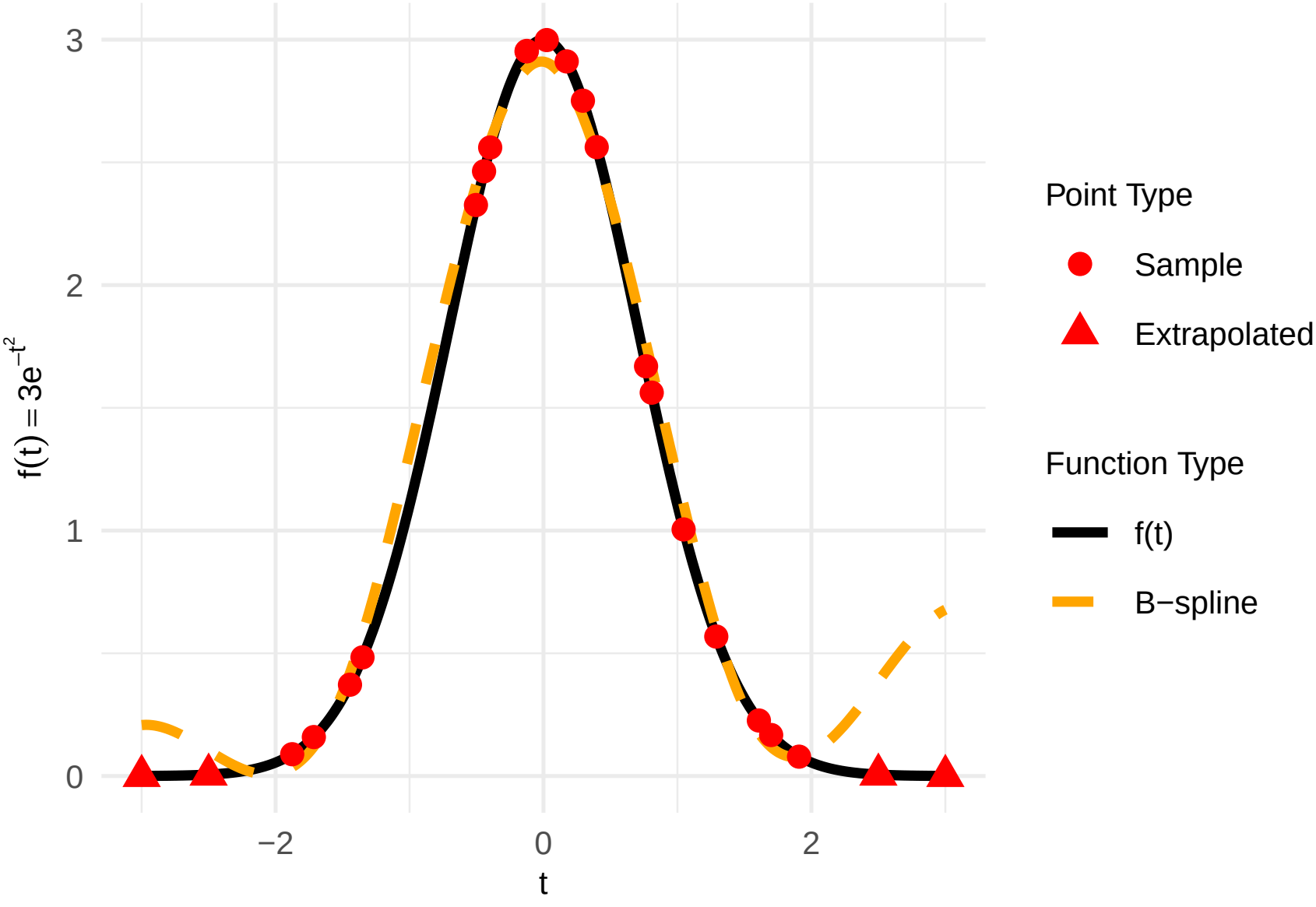
Slide 4n: Quadratic B-Spline Interpolation



Slide 4o: Cubic B-Spline Smoothing with $\lambda = 0.5$



Slide 4p: Cubic B-Spline, 20 RS Points, $\lambda = 0.01$



Slide 5a: Hierarchical Agglomerative Clustering (HAC)

- **Algorithm** (Dash et al. 2003; Ferreira and Hitchcock 2009):
  - Start with every data point (curve) in its own cluster.
  - Find the two "closest" clusters and merge them.
  - Repeat until all data points are in a single cluster.

- Define L^2 distance as norm:

$$d_{ij} = \sqrt{\int_{\mathcal{T}} [x_i(t) - x_j(t)]^2 dt}$$

- Linkage types:
  - **Single**: minimum distance between any two points in both clusters.
  - **Complete**: maximum distance between any two points in both clusters.
  - **Average**: average pairwise distances.
  - **Ward**: minimum increase in total within-cluster variance.

Slide 5b: Distance Calculation

- Functional data consist of observations that are inherently continuous functions.
- Although the underlying process is continuous, in practice we only observe discretized measurements sampled from these curves.
- Because the functions are only available at discrete grid points, the distance integral must be approximated numerically.
- A common approach in multivariate analysis is to use the **trapezoidal rule**, applied to the integrand $h(t)$, yielding:

$$I_n \approx \frac{t_n - t_0}{2n} \left[h(t_0) + 2 \sum_{k=1}^{n-1} h(t_k) + h(t_n) \right]$$

- In FDA, each discretely observed curve is commonly transformed into a smooth, continuous function using an appropriate smoothing technique and the integration is performed on the analytic expression.

Slide 5c: Convergence of Trapezoid Rule Approximation

Consider two functions on the interval $[0, 2\pi]$:

$$f(t) = sin(t) + t$$

$$g(t) = cos(t^2 + 0.5)$$

We are going to approximate $d_{f,g} = \sqrt{\int_0^{2\pi} [f(t) - g(t)]^2 dt}$:

Table 1: Convergence of Trapezoid Rule Approximation of L2 Distance

n	Approximation	True	AbsoluteError
5	10.554437	8.738884	1.8155533
10	8.053558	8.738884	0.6853266
15	9.108159	8.738884	0.3692750
25	8.806260	8.738884	0.0673761
50	8.752400	8.738884	0.0135160
100	8.742063	8.738884	0.0031789
500	8.739008	8.738884	0.0001237
1000	8.738915	8.738884	0.0000309

Slide 5d: Convergence of B-Splines Approximation

Table 2: Convergence of B-Spline Approximation of L2 distance

Nbasis	Approximation	True	AbsoluteError
5	8.622966	8.738884	0.1159182
8	8.656217	8.738884	0.0826672
10	8.643527	8.738884	0.0953568
15	8.650243	8.738884	0.0886414
20	8.692921	8.738884	0.0459630
25	8.729542	8.738884	0.0093424
30	8.914712	8.738884	0.1758279

Slide 5e: Rand Index for Clustering Evaluation

- Measures agreement between predicted and true partitions of curves.
- Let:
  - $TP = \#$ of pairs correctly clustered together.
  - $TN = \#$ of pairs correctly separated.
  - $FP = \#$ of pairs clustered together in the prediction but separated in the truth.
  - $FN = \#$ of pairs separated in the prediction but clustered together in the truth.
- Rand Index (Rand 1971):

$$\text{RI} = \frac{TP + TN}{TP + TN + FP + FN} \in [0, 1]$$

- Adjusted Rand Index (ARI) (Hubert and Arabie 1985):

$$ARI = \frac{Index - Expected\ Index}{Max\ Index - Expected\ Index} = \frac{RI - E[RI]}{Max[RI] - E[RI]} \in [-1, 1]$$

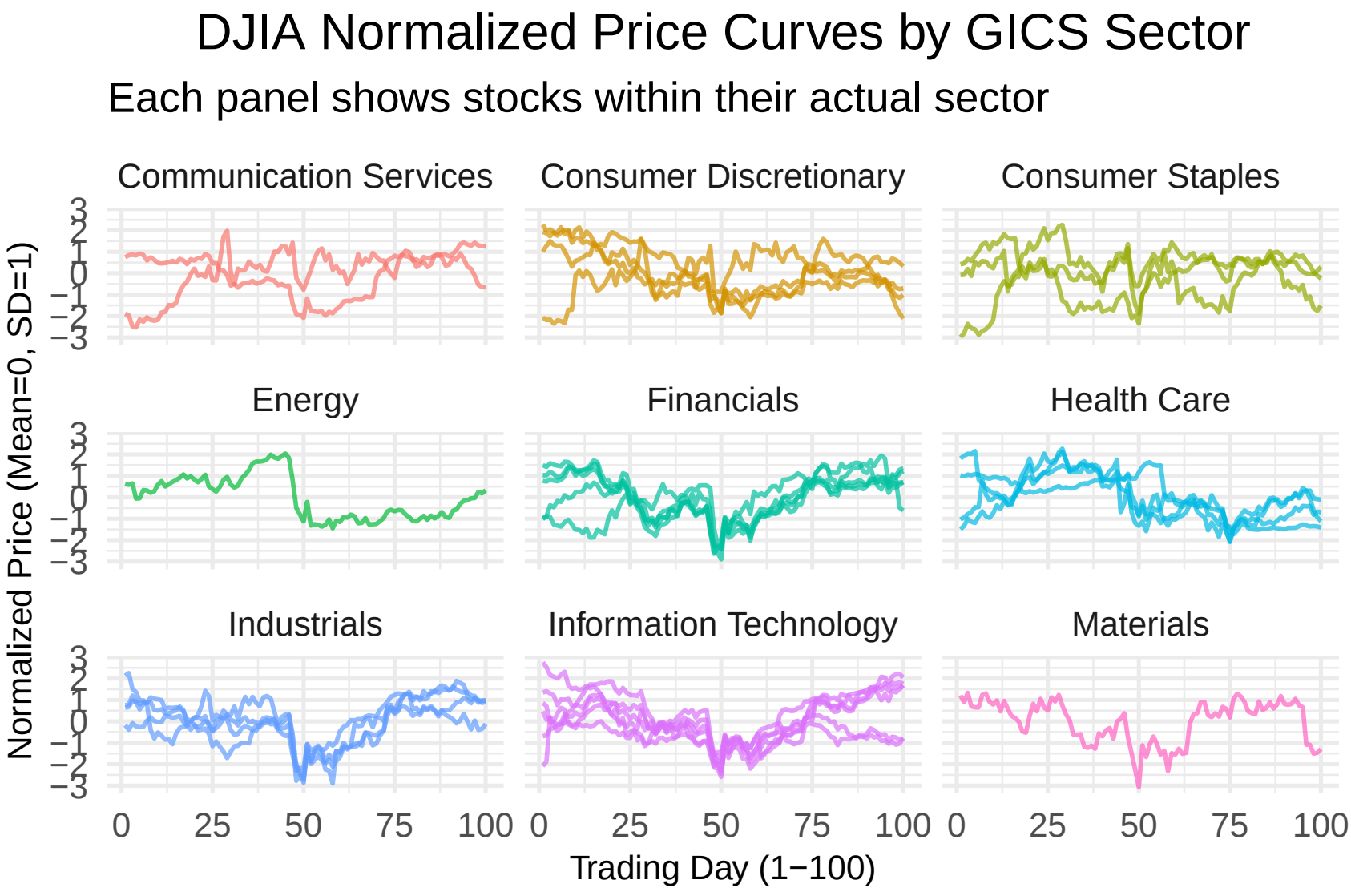
Slide 5f (Example): DJIA Index

- Retrieve historical price data for Dow Jones Industrial Average (DJIA) components (e.g., Amazon, NVIDIA) over the past 100 trading days.
- Normalize each stock's price series.
- Represent each time series as a smooth functional curve using a basis of 20 cubic B-spline functions.
- Apply HAC to the functional data representation.
- Assess the quality of clustering by comparing the derived clusters to known sector classifications (e.g., Financials, Industrial).
- Plot visual diagnostics.

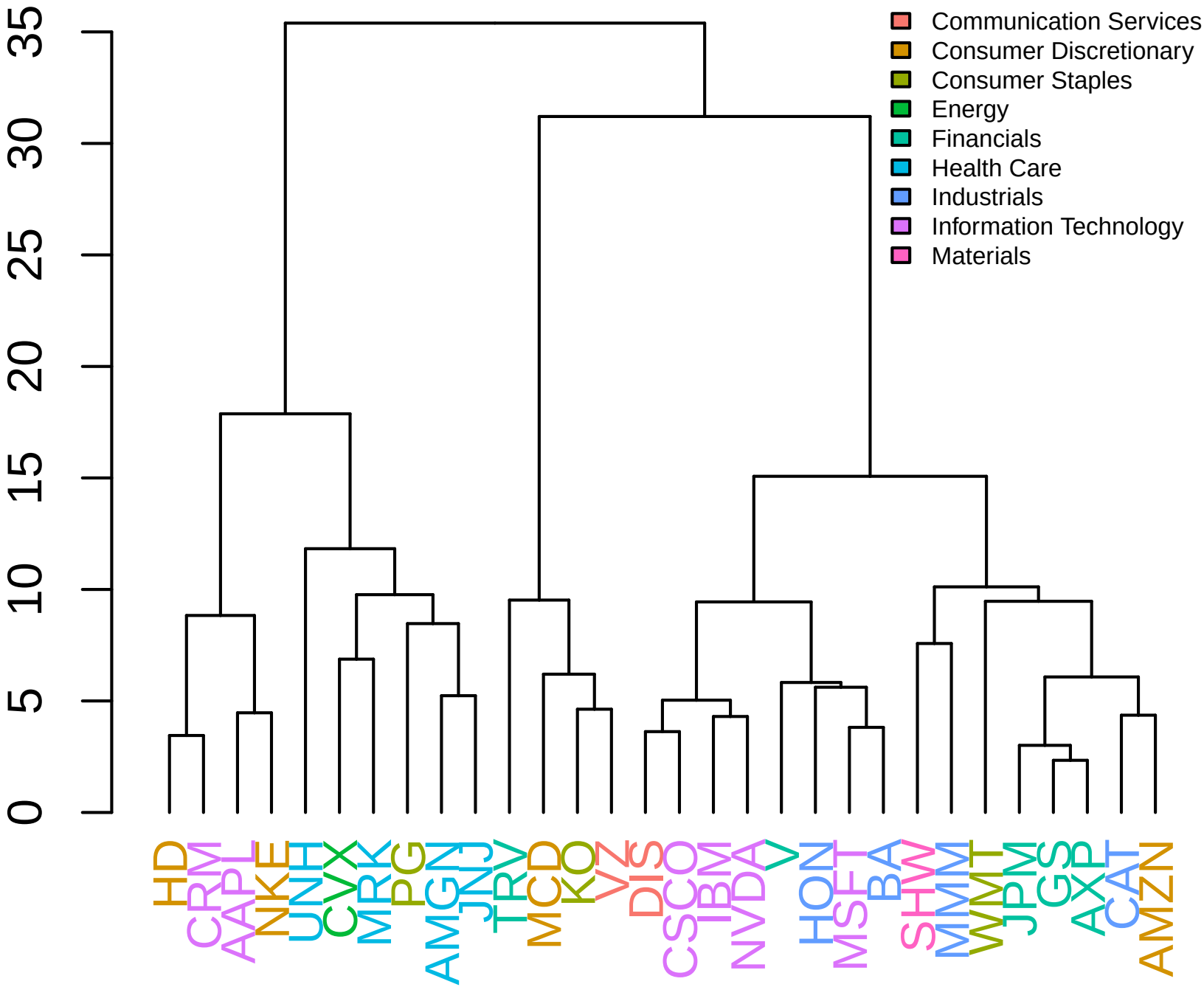
Table 3: Clustering Performance Evaluation

Metric	Value
Rand Index (RI)	0.8207
Adjusted Rand Index (ARI)	0.1492

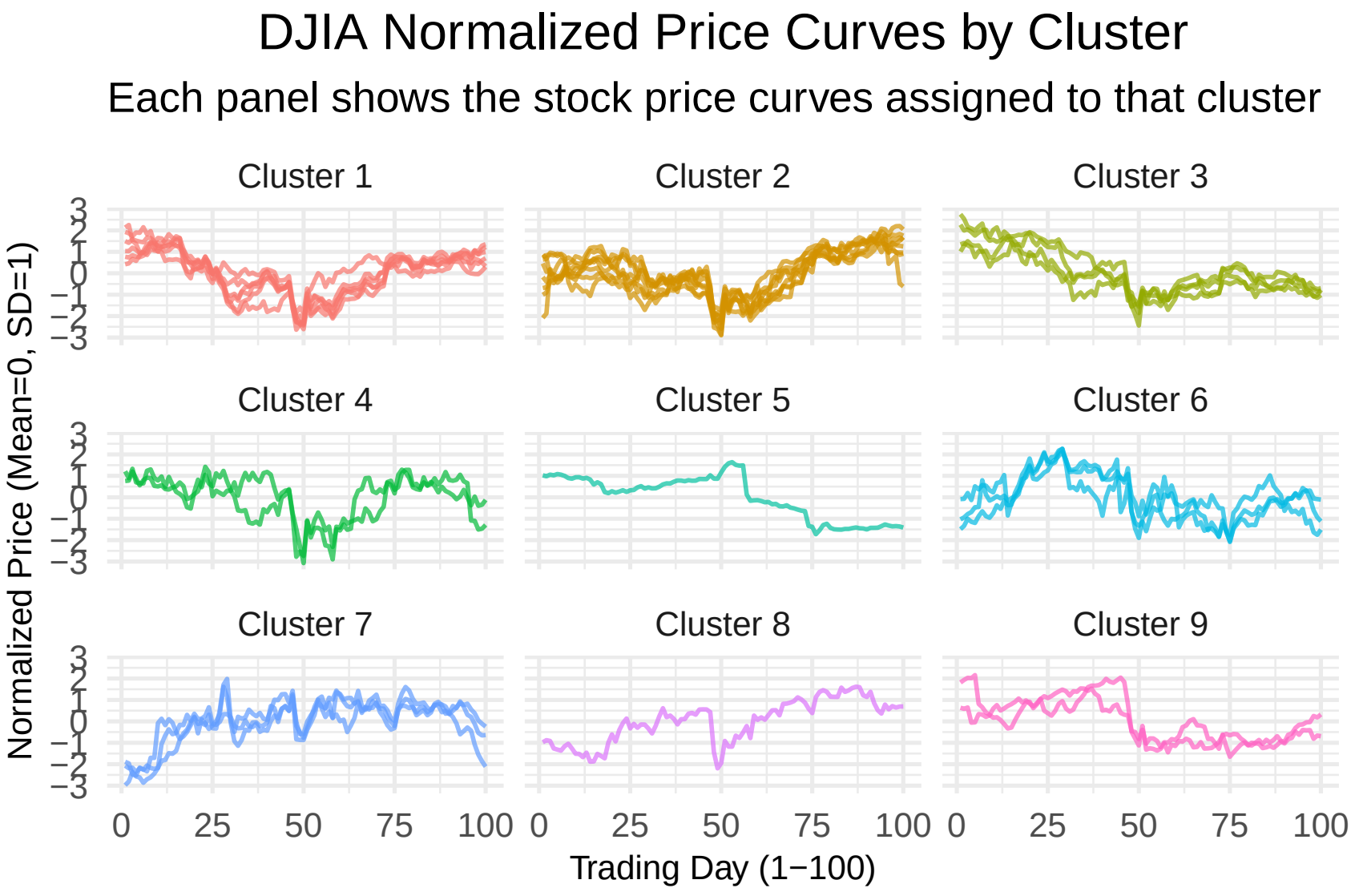
Slide 5g: DJIA by Sector before HAC



Slide 5h: HAC Clustering of DJIA Stocks Dendrogram



Slide 5i: DJIA by Cluster



Slide 6a: From ODEs to Basis Functions

The core of the method is to move away from numerically solving the differential equation system at each step. Instead, the unknown solution trajectory for each variable $x_i(t)$ is approximated by a flexible **basis function expansion**, such as a spline.

$$\hat{x}_i(t) = \sum_{k=1}^{K_i} c_{ik}\phi_{ik}(t) = \mathbf{c}_i^T \phi_i(t)$$

This transforms the problem and creates two distinct classes of parameters to be estimated:

- **Structural Parameters (θ):** These are the parameters of primary interest within the differential equation, such as reaction rates or physical constants.
- **Nuisance Parameters (c):** These are the coefficients for the basis functions. They are essential for defining the shape of the solution trajectory but are not of direct concern.

The main challenge is to estimate the structural parameters θ accurately, without them being overwhelmed by the potentially vast number of nuisance parameters **c**.

Slide 6b: The Two-Part Criterion for Least Squares

The method ingeniously balances two objectives by combining them into a single, penalized fitting criterion.

1. **Fidelity to Data:** The first objective is that the estimated curve $\hat{\mathbf{x}}(t)$ must closely match the observed data points $\mathbf{y}$. This is a standard sum of squared errors term.

$$H(\mathbf{c}, \theta) = \sum_{i \in \mathcal{I}} w_i ||\mathbf{y}_i - \hat{\mathbf{x}}_i(t_i)||^2$$

2. **Fidelity to the Equation:** The second objective is that the curve $\hat{\mathbf{x}}(t)$ must satisfy the dynamics of the system. A penalty term is added to penalize deviations from the differential equation itself.

$$PEN_i(\hat{\mathbf{x}}) = \int [\dot{\hat{x}}_i(t) - f_i(\hat{\mathbf{x}}, \mathbf{u}, t|\theta)]^2 dt$$

These are combined into an **inner criterion**, J, which is minimized with respect to the nuisance coefficients $\mathbf{c}$ for a *fixed* θ.

$$J(\mathbf{c}|\theta, \lambda) = H(\mathbf{c}, \theta) + \sum_{i=1}^{d} \lambda_i PEN_i(\hat{\mathbf{x}})$$

The smoothing parameters $\lambda_i \geq 0$ control the trade-off.

Slide 6c: The "Parameter Cascade": Generalized Profiling

Instead of a joint optimization over both θ and $\mathbf{c}$, the method uses a hierarchical, multi-criterion approach called **Generalized Profiling (GP)** or a **parameter cascade** (J. O. Ramsay et al. 2007).

Inner Optimization: For any given set of structural parameters θ, the optimal nuisance coefficients $\hat{\mathbf{c}}$ are found by minimizing the inner criterion J.

$$\hat{\mathbf{c}}(\theta) = \arg\min_{\mathbf{c}} J(\mathbf{c}|\theta, \lambda)$$

This critical step defines the nuisance parameters as an *implicit function* of the structural parameters.

Outer Optimization: This implicit function $\hat{\mathbf{c}}(\theta)$ is then substituted back into the data-fitting criterion H, effectively eliminating the nuisance parameters.

$$H(\theta) = \sum_{i \in \mathcal{I}} w_i ||\mathbf{y}_i - \hat{\mathbf{x}}_i(t_i; \hat{\mathbf{c}}(\theta))||^2$$

The final estimate for the structural parameters is then found by minimizing this outer criterion, which now depends only on θ.

$$\hat{\theta} = \arg\min_{\theta} H(\theta)$$

Slide 6d: The FitzHugh-Nagumo Model and Simulation Results

The goal is to estimate parameters for the **FitzHugh-Nagumo** equations (Postnikov and Titkova 2016), which model the electrical potential across a neuron's membrane.

The system is described by two coupled differential equations:

$$\dot{V} = c\left(V - \frac{V^3}{3} + R\right)$$
$$\dot{R} = -\frac{1}{c}\left(V - a + bR\right).$$

Where V is the membrane potential and R is a recovery variable.

Simulation Details:

- Synthetic data for the voltage (V) was generated by numerically solving the system with known parameters ($a = 0.2,\ b = 0.2,\ c = 3.0$).
- To simulate experimental conditions, Gaussian noise with a standard deviation of σ=0.25 was added to the true voltage data.
- The smoothing parameters ($\lambda_V, \lambda_R, \lambda_{IC}$) required for the GP estimation were selected automatically via Bayesian Optimization.

Slide 6e(Results): Generalized Profiling Solution

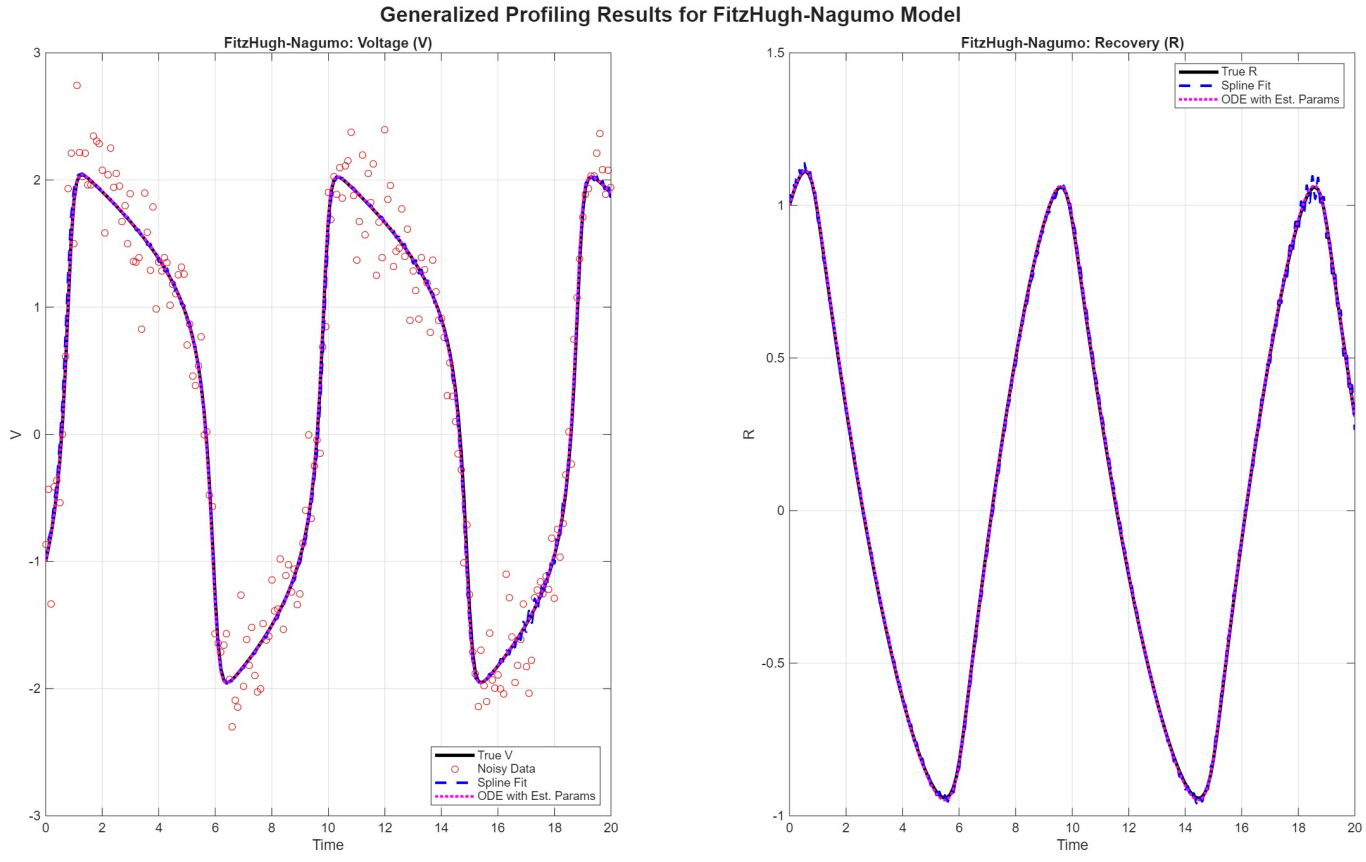


Figure 1: The FitzHugh-Nagumo Model Solution via Generalized Profiling

Slide 6f (Results): Parameter and Lambda Estimation

The optimization process successfully recovered the model parameters from the noisy data.

Table 4: Recovered model parameters from noisy data

Parameter	True.Value	Estimated.Value
a	0.2	0.198
b	0.2	0.179
c	3.0	3.000

Table 5: Optimal smoothing parameters via Bayesian optimization

Penalty.Term	Estimated.Value
λ_V (Voltage ODE)	10.30
λ_R (Recovery ODE)	10.48
λ_{IC} (Initial Conditions)	7505.70

Appendix

Slide A1a: The Euler Method: A Naive Approach

Euler's method is the most basic way to solve an initial value problem, $y'(t) = f(t, y),\ y(t_0) = y_0$. It approximates the solution by taking small steps along the tangent line.

The Formula:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

Problems:

- **Low Order:** It is a **first-order** method. The error over an interval is proportional to h. Extremely small step size h is required for acceptable accuracy.
- **Systematic Drift:** It only uses the slope at the **beginning** of each step. If the curve bends away, the Euler method will consistently "miss" the true solution in the direction of that initial tangent, leading to rapid error accumulation.

Slide A1b: Runge-Kutta: Using Multiple Slopes

Runge-Kutta (RK) methods (Shampine 1986) are a family of algorithms that dramatically improve accuracy by using a **weighted average of multiple slopes** within each step.

The Core Idea:

Instead of just one slope, we compute several "trial" slopes and combine them. A simple second-order RK method looks like this:

1. Calculate slope at the start: $k_1 = f(t_n, y_n)$
2. Use k_1 to estimate the solution at the midpoint of the interval: $k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$
3. Use the midpoint slope k_2 to take the full step: $y_{n+1} = y_n + h \cdot k_2$

Where does performance improvement come from:

- By evaluating the slope inside the interval, RK methods get a much better approximation of the solution's path.
- RK are **higher-order** methods, meaning they achieve high accuracy with a much larger step size h.

Slide A1c: ODE Solvers

Popular MATLAB's solver ode45 relies on **Runge-Kutta 4(5)** method which is also known as the **Dormand-Prince (4,5) pair**. The slopes are calculated in the following manner (Shampine 1986):

$$k_1 = f(t_n, y_n)$$

$$k_2 = f\left(t_n + \frac{1}{5}h, y_n + \frac{1}{5}hk_1\right)$$

$$k_3 = f\left(t_n + \frac{3}{10}h, y_n + h\left(\frac{3}{40}k_1 + \frac{9}{40}k_2\right)\right)$$

$$k_4 = f\left(t_n + \frac{4}{5}h, y_n + h\left(\frac{44}{45}k_1 - \frac{56}{15}k_2 + \frac{32}{9}k_3\right)\right)$$

$$k_5 = f\left(t_n + \frac{8}{9}h, y_n + h\left(\frac{19372}{6561}k_1 - \frac{25360}{2187}k_2 + \frac{64448}{6561}k_3 - \frac{212}{729}k_4\right)\right)$$

$$k_6 = f\left(t_n + h, y_n + h\left(\frac{9017}{3168}k_1 - \frac{355}{33}k_2 + \frac{46732}{5247}k_3 + \frac{49}{176}k_4 - \frac{5103}{18656}k_5\right)\right)$$

$$k_7 = f\left(t_n + h, y_n + h\left(\frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6\right)\right)$$

Slide A1d:The Dormand-Prince (4,5) Pair

The solutions are constructed in the following way:

- **5th-Order Solution (more accurate, used for the final result):**

$$y_{n+1}^{(5)} = y_n + h\left(\frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6\right)$$

- **4th-Order Solution (less accurate, used for error estimation):**

$$y_{n+1}^{(4)} = y_n + h\left(\frac{5179}{57600}k_1 + \frac{7571}{16695}k_3 + \frac{393}{640}k_4 - \frac{92097}{339200}k_5 + \frac{187}{2100}k_6 + \frac{1}{40}k_7\right)$$

The key is that both formulas reuse the same k_i values, making this very computationally efficient. The RK coefficients are chosen to be identical to the terms in the true solution's expansion up to a certain power of h.

Slide A1e: Adaptive Step-Size Control

The solver uses the two Dormand-Prince solutions to control the step size h automatically.

1. Estimate the Error: The error E is the difference between the two solutions.

$$E = ||y_{n+1}^{(5)} - y_{n+1}^{(4)}||$$

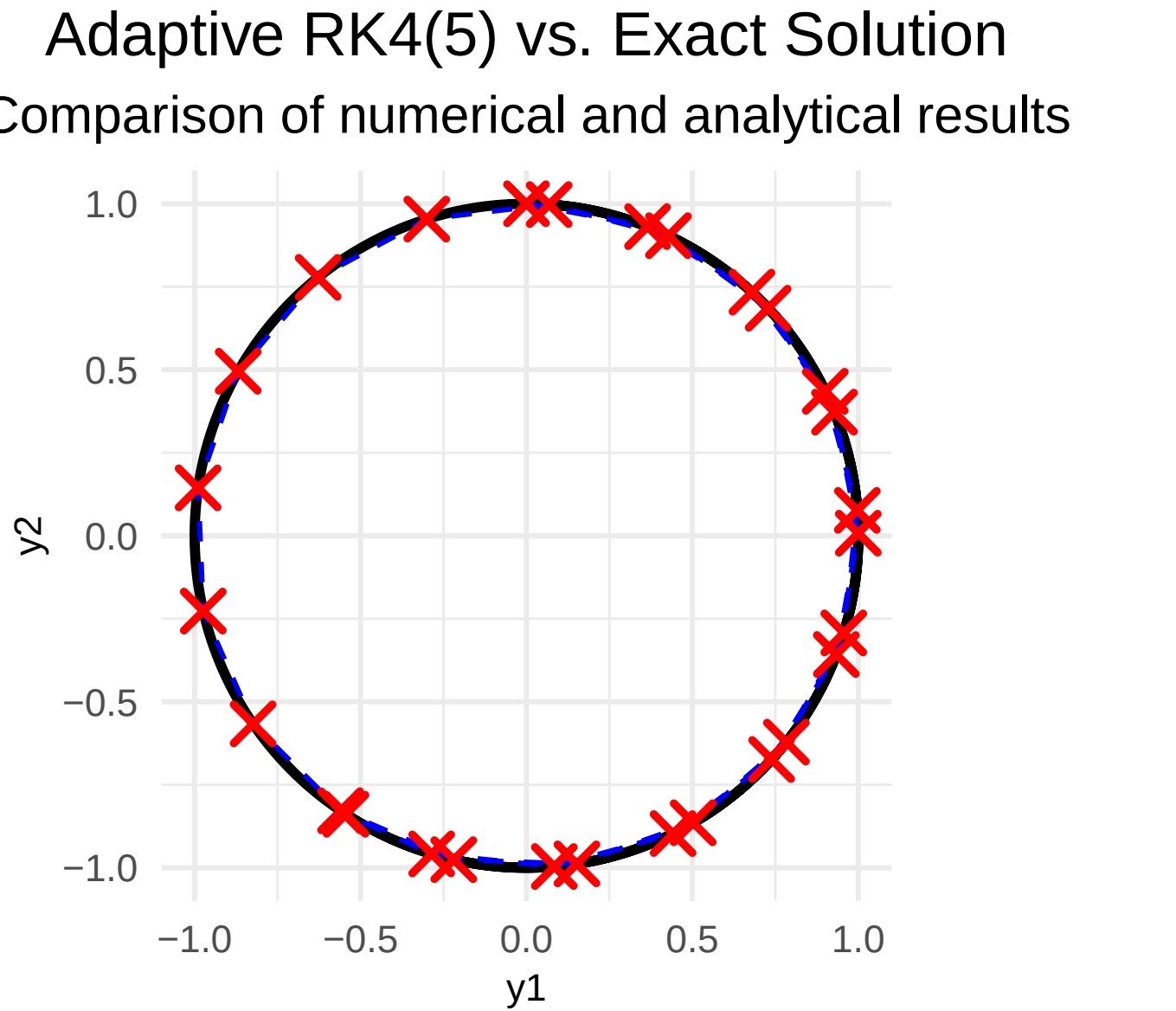
2. Define the Tolerance: `AbsTol` is the acceptable error when the solution is near zero, while `RelTol` is the acceptable fractional error when the solution is large.

$$\text{Threshold} = \text{AbsTol} + \text{RelTol} \cdot ||y_{n+1}^{(5)}||$$

3. Adjust the Step:

- If $E \leq$ Threshold, the step is **accepted**. The solver may try a larger step next time.
- If $E >$ Threshold, the step is **rejected**. The solver retries from the same point with a smaller step size.

Slide A1f (Example): Linear Equations $y_1' = y_2$ and $y_2' = -y_1$



Slide A1g (Example): Non-linear Equations

Consider the following system of non-linear ODEs:

$$\begin{cases} y_1' = y_2^2, \\ y_2' = y_1 y_2. \end{cases}$$

Table 6: Detailed Intermediate Calculations of the RK4(5) Solver

			Stage Slopes							Predictions		Control			Truth
Step	Start Time	Start Point	k1	k2	k3	k4	k5	k6	k7	y5 pred	y4 pred	Error Term	Threshold	Decision	y true
1	0.00000000	[0.00000000, 1.00000000]	[1.00000000, 0.00000000]	[1.00000000, 0.02000000]	[1.00090020, 0.03001350]	[1.00641987, 0.08057748]	[1.00790975, 0.08994006]	[1.01002369, 0.10126789]	[1.01006705, 0.10083844]	[0.10033467, 1.00502092]	[0.10033471, 1.00502092]	3.83e-08	1.01e-05	Accept	[0.10033467, 1.00502092]
2	0.10000000	[0.10033467, 1.00502092]	[1.01006705, 0.10083844]	[1.01819090, 0.14201165]	[1.02601507, 0.16338897]	[1.07100306, 0.27654447]	[1.08186329, 0.30043915]	[1.09597517, 0.32740677]	[1.09568895, 0.32379821]	[0.30933624, 1.04675162]	[0.30933762, 1.04675162]	1.38e-06	1.09e-05	Accept	[0.30933625, 1.04675160]
3	0.30000000	[0.30933624, 1.04675162]	[1.09568895, 0.32379821]	[1.13289604, 0.39272742]	[1.16080973, 0.43211824]	[1.32717296, 0.66022724]	[1.37007207, 0.71721662]	[1.42045823, 0.77892685]	[1.41474021, 0.76599719]	[0.64400439, 1.18942852]	[0.64401526, 1.18943219]	1.15e-05	1.35e-05	Accept	[0.64400462, 1.18942925]
4	0.57214899	[0.64400439, 1.18942852]	[1.41474021, 0.76599719]	[1.49155172, 0.85838689]	[1.54143568, 0.91361661]	[1.83413939, 1.23708013]	[1.90802601, 1.31745772]	[1.99199718, 1.40756380]	[1.97913963, 1.39207062]	[0.98951653, 1.40681897]	[0.98952491, 1.40682398]	9.77e-06	1.72e-05	Accept	[0.98951764, 1.40682094]
5	0.78012942	[0.98951653, 1.40681897]	[1.97913963, 1.39207062]	[2.11222469, 1.53425749]	[2.19620976, 1.62089721]	[2.69025682, 2.13239608]	[2.81482422, 2.26061392]	[2.95560621, 2.40515686]	[2.93293323, 2.38100401]	[1.39030165, 1.71258087]	[1.39031150, 1.71258824]	1.23e-05	2.21e-05	Accept	[1.39030479, 1.71258500]
6	0.94725635	[1.39030165, 1.71258087]	[2.93293323, 2.38100401]	[3.15331422, 2.60702986]	[3.29093175, 2.74583186]	[4.10236634, 3.56739717]	[4.30692985, 3.77408573]	[4.53787597, 4.00744230]	[4.50014279, 3.96877523]	[1.87086890, 2.12135400]	[1.87088084, 2.12136407]	1.56e-05	2.83e-05	Accept	[1.87087583, 2.12136191]
7	1.07992412	[1.87086890, 2.12135400]	[4.50014279, 3.96877523]	[4.85998815, 4.33233676]	[5.08368451, 4.55638436]	[6.40495146, 5.88361248]	[6.73815025, 6.21807861]	[7.11425679, 6.59581027]	[7.05244975, 6.53335042]	[2.46017472, 2.65564488]	[2.46018948, 2.65565820]	1.99e-05	3.62e-05	Accept	[2.46018828, 2.65565931]
8	1.19472230	[2.46017472, 2.65564488]	[7.05244975, 6.53335042]	[7.63721976, 7.12072147]	[7.99995668, 7.4833231]	[10.14498102, 9.63186562]	[10.68611619, 10.17376883]	[11.29693929, 10.78575100]	[11.19626769, 10.68458138]	[3.19316143, 3.34608244]	[3.19317989, 3.34609978]	2.53e-05	4.63e-05	Accept	[3.19318626, 3.34610796]
9	1.26730408	[3.19316143, 3.34608244]	[11.19626769, 10.68458138]	[12.14494122, 11.63519561]	[12.73275182, 12.22258103]	[16.21138032, 15.70326880]	[17.08921081, 16.58157017]	[18.08012103, 17.57335891]	[17.91653793, 17.40936684]	[4.11297368, 4.23279316]	[4.11299692, 4.23281553]	3.23e-05	5.90e-05	Accept	[4.11301741, 4.23283737]
10	1.33229345	[4.11297368, 4.23279316]	[17.91653793, 17.40936684]	[19.45461768, 18.94899018]	[20.40704538, 19.90081772]	[26.04617840, 25.54112803]	[27.46951001, 26.96475122]	[29.07623647, 28.57218577]	[28.81073419, 28.30632713]	[5.27358996, 5.36756315]	[5.27361935, 5.36759186]	4.11e-05	7.53e-05	Accept	[5.27366509, 5.36763854]
11	1.38339977	[5.27358996, 5.36756315]	[28.81073419, 28.30632713]	[29.82843512, 29.32434281]	[30.38581461, 29.88164346]	[33.33175085, 32.82791503]	[33.92282572, 33.41898962]	[34.64793801, 34.14420074]	[34.61438832, 34.11073252]	[5.79779313, 5.88339938]	[5.79779346, 5.88339971]	4.68e-07	8.26e-05	Accept	[5.79788372, 5.88349008]

Slide A2a: The Non-Linear Least Squares (NLS) Problem

Given a set of m data points (x_i, y_i) and a non-linear model function $F(x, \beta)$ with a vector of n unknown parameters $\beta \in \mathbb{R}^n$, the objective is to find the optimal parameter vector that best fits the model to the data.

This is framed as an optimization problem by defining a **residual** for each data point:

$$r_i(\beta) = y_i - F(x_i, \beta)$$

In our research, x_i are days, y_i are daily known cases and $F(x_i, \beta)$ is a numeric solution of system of ODEs for C_k, i.e. known cases.

The goal is to minimize the sum of the squares of these residuals. The objective function $S(\beta)$ is therefore the squared Euclidean norm (L_2-norm) of the residual vector $\mathbf{r}(\beta)$:

$$\min_{\beta} S(\beta) = \min_{\beta} \sum_{i=1}^{m} [r_i(\beta)]^2 = \min_{\beta} ||\mathbf{r}(\beta)||_2^2$$

Slide A2b: The Challenge and The Iterative Approach

A closed-form analytical solution for the optimal parameters β is generally unobtainable. Setting the gradient of the objective function, $\nabla S(\beta)$, to zero yields a system of non-linear equations which cannot be solved directly.

Therefore, the solution must be found using an **iterative numerical method**. The process is as follows:

1. Begin with an initial estimate for the parameters, β_0.
2. At iteration k, compute an **update step**, $\Delta\beta$, which seeks to improve the current estimate.
3. Update the parameter vector: $\beta_{k+1} = \beta_k + \Delta\beta$.
4. Repeat the process until a convergence criterion is met (e.g., the norm of the update step $||\Delta\beta||$ is below a threshold).

The fundamental challenge lies in determining an optimal update step at each iteration.

Slide A2c: Local Linearization using the Jacobian

The standard methodology to make the problem tractable is to form a local linear approximation of the non-linear model at each iterate β_k. This is achieved using a first-order Taylor expansion.

The **Jacobian matrix**, J, is central to this linearization. It is an $m \times n$ matrix containing the partial derivatives of each residual component with respect to each parameter, evaluated at β_k:

$$J_{ij}(\beta_k) = \left.\frac{\partial r_i}{\partial \beta_j}\right|_{\beta_k} = -\left.\frac{\partial F(x_i, \beta)}{\partial \beta_j}\right|_{\beta_k}$$

The Jacobian describes the sensitivity of the residuals to infinitesimal changes in the parameters. By default, the Jacobian is approximated using finite differences.

The residual vector for a small step $\Delta\beta$ can now be approximated linearly:

$$\mathbf{r}(\beta_k + \Delta\beta) \approx \mathbf{r}(\beta_k) + J_k \Delta\beta$$

This transforms the original problem into solving a sequence of linear least-squares subproblems for the update step $\Delta\beta$.

Slide A2d: The Levenberg-Marquardt Update Step

At each iteration, the update step $\Delta\beta$ is found by solving the linear least-squares problem:

$$\min_{\Delta\beta} ||\mathbf{r}(\beta_k) + J_k \Delta\beta||_2^2$$

The **Gauss-Newton method** solves the corresponding normal equations:

$$(J_k^T J_k)\Delta\beta = -J_k^T \mathbf{r}(\beta_k)$$

This method can be unstable if the matrix $J_k^T J_k$ is singular or ill-conditioned.

The **Levenberg-Marquardt (LM) algorithm** (Moré 1978) introduces a regularization or damping parameter $\lambda \geq 0$ to ensure stability:

$$(J_k^T J_k + \lambda I)\Delta\beta = -J_k^T \mathbf{r}(\beta_k)$$

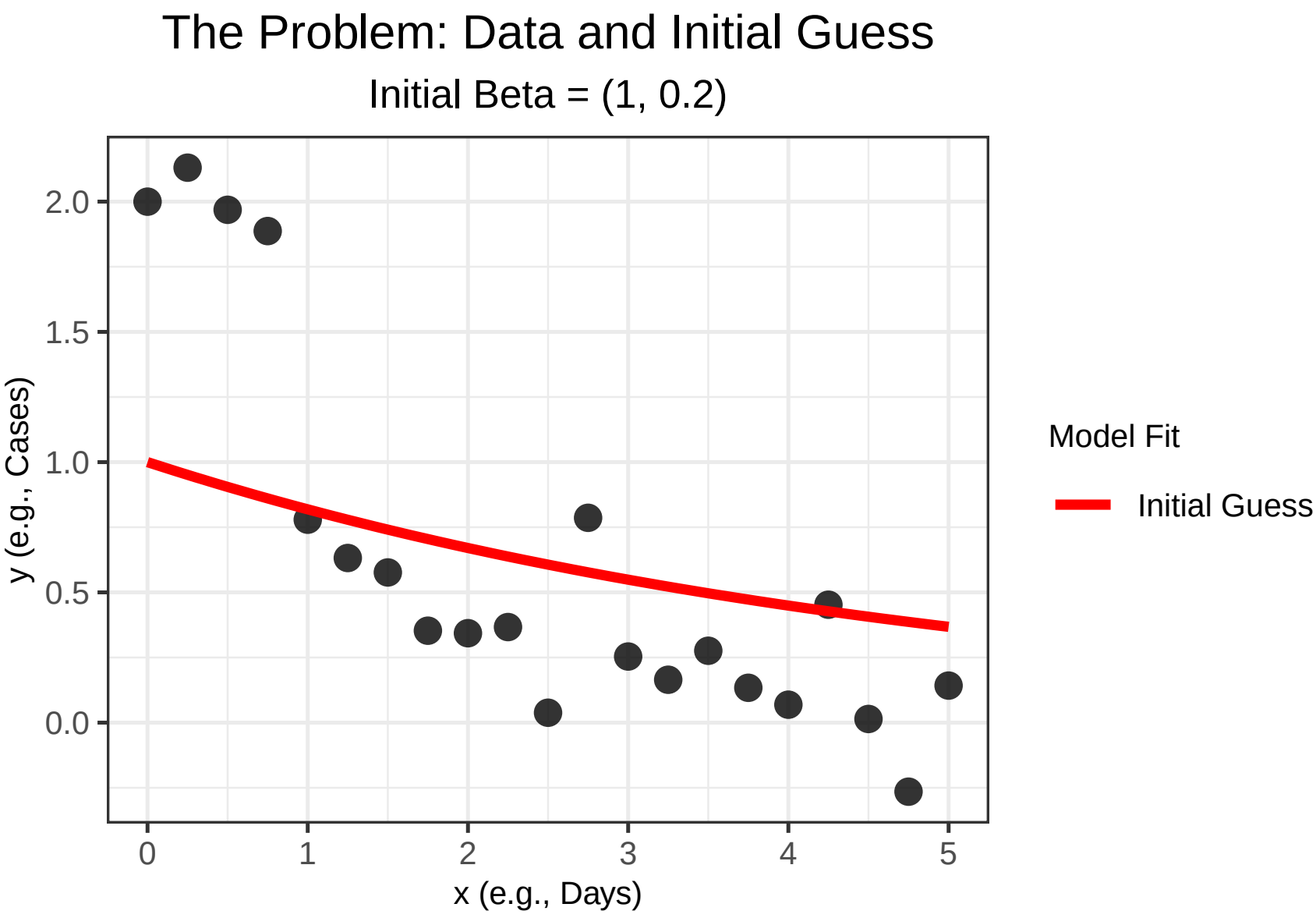
The parameter λ is adjusted at each iteration.

- For small λ, the LM method approximates the fast-converging Gauss-Newton method.
- For large λ, the LM method approximates the robust but slower method of gradient descent.

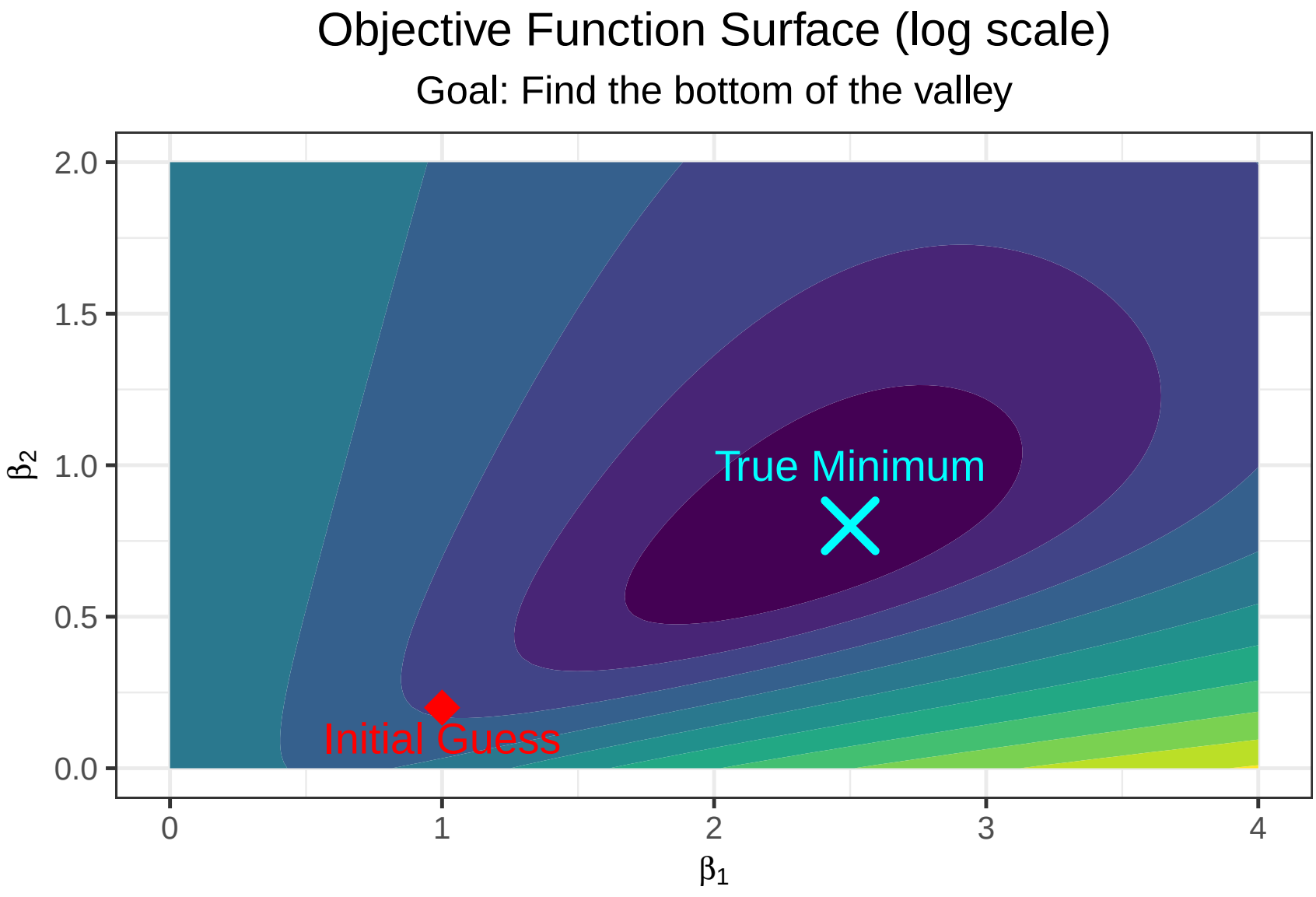
Slide A2e (Example): Data Simulation and Model Definition

- Simulate data for a non-linear least squares problem.
  - Define an exponential decay model ($y = f(x) = \beta_1 e^{-\beta_2 x}$) and generate a "true" dataset: $y = 2.5e^{-0.8x}$.
  - Add random Gaussian noise ($\sigma = 0.25$) to this data to mimic a real-world experimental measurement.
- Define the objective function to be minimized: the sum of squared residuals (SSR), which measures the total squared difference between the noisy data and the model's predictions.
- The convergence characteristics of the algorithm are investigated through contour plots of the SSR landscape. In these visualizations, isochromatic mapping denotes different elevations of the error function.

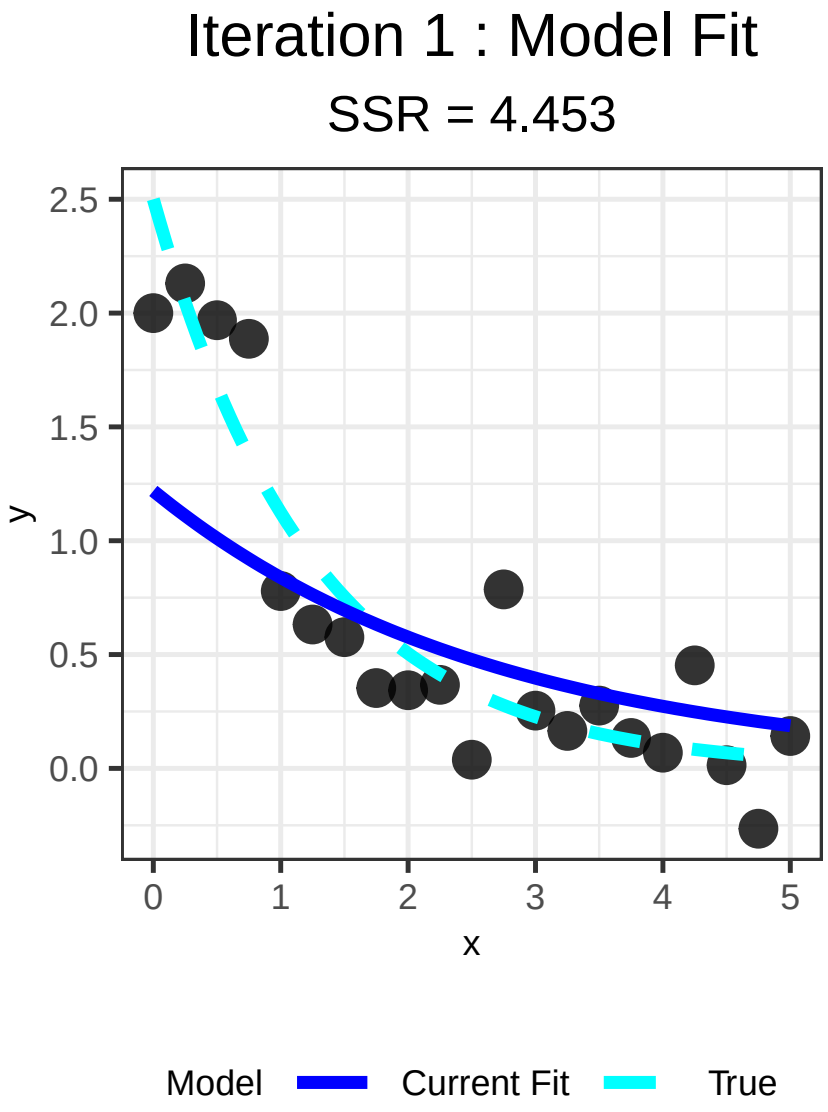
Slide A2f (Example): Initial Guess

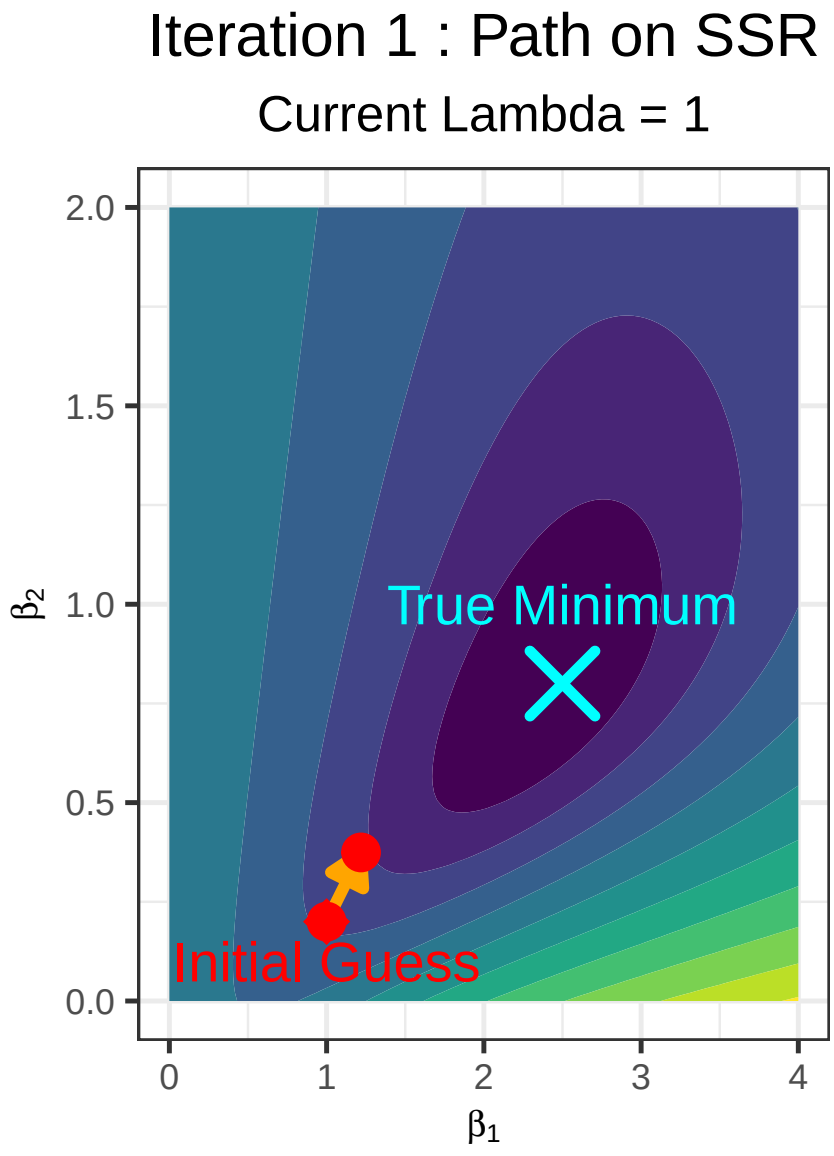


Slide A2g (Example): Objective Function Surface

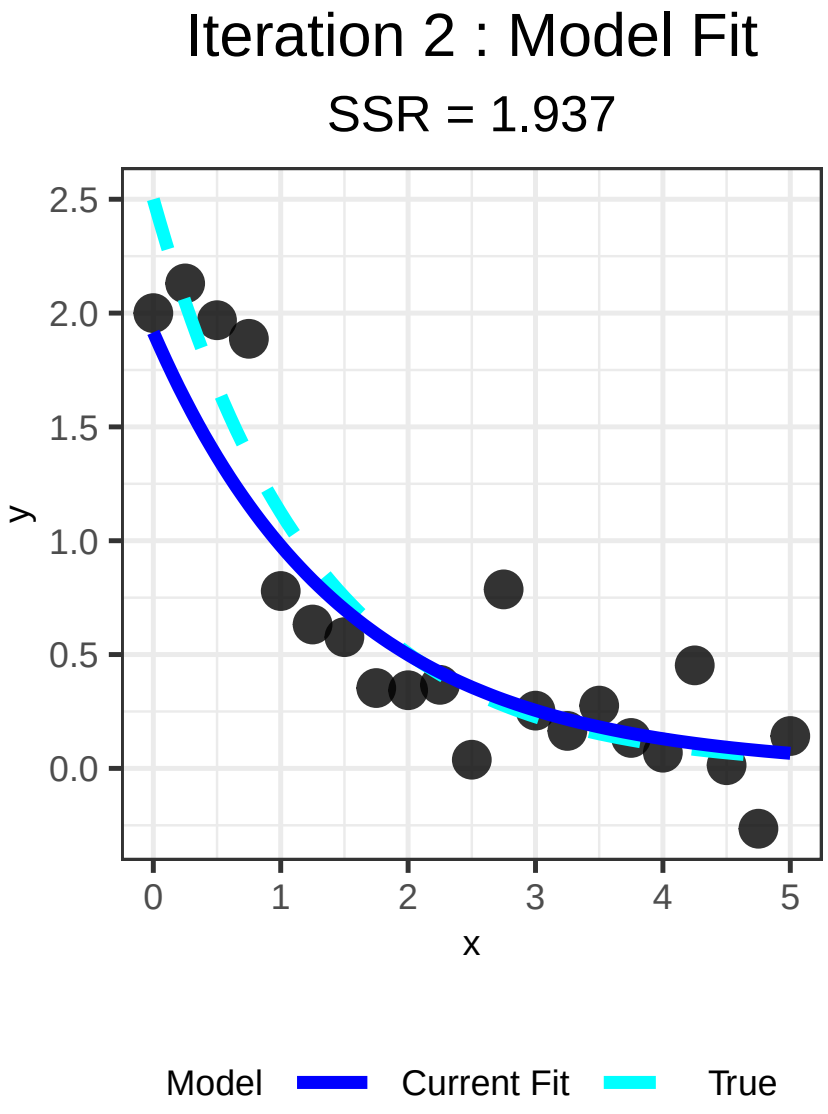


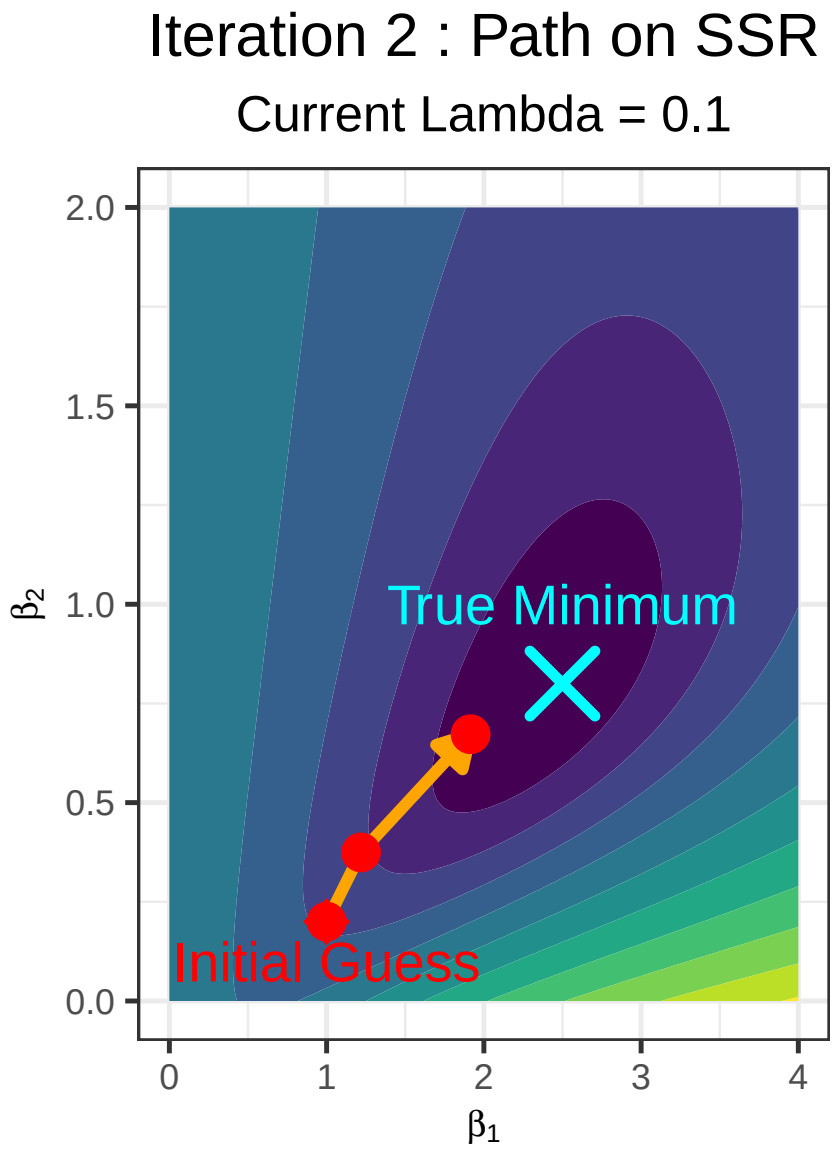
Slide A2h (Example): LM Iteration 1



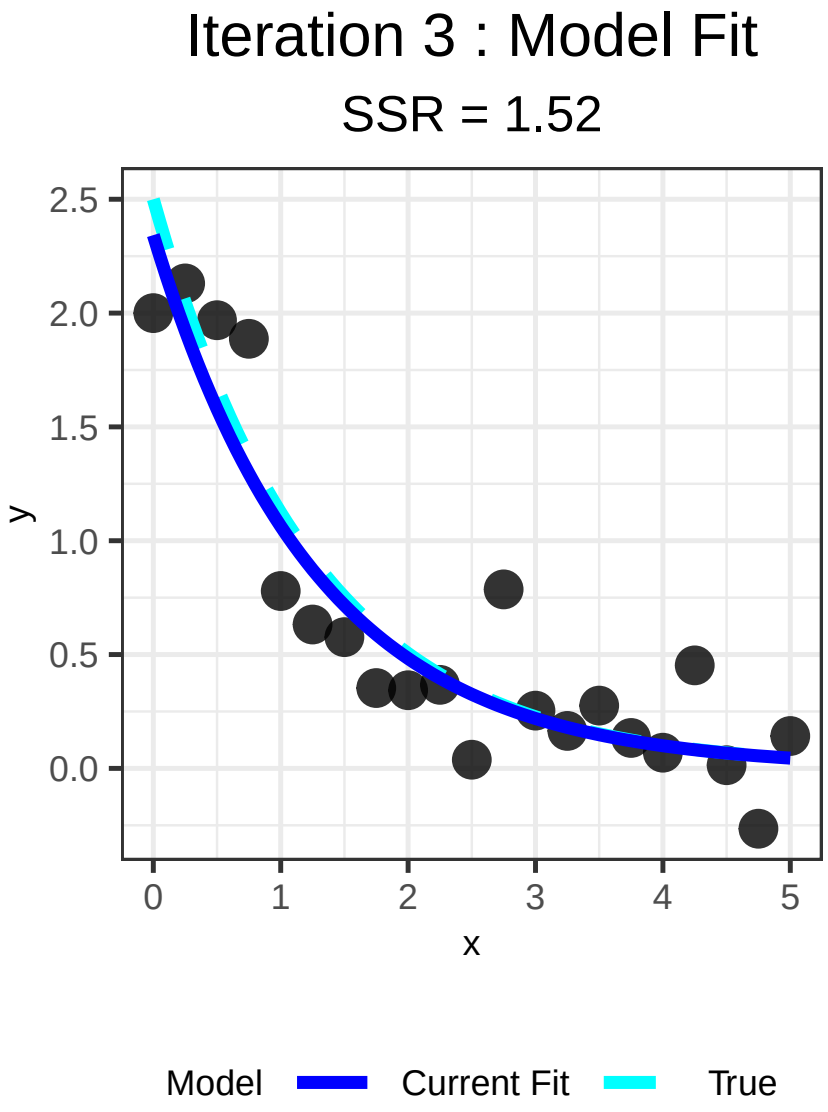


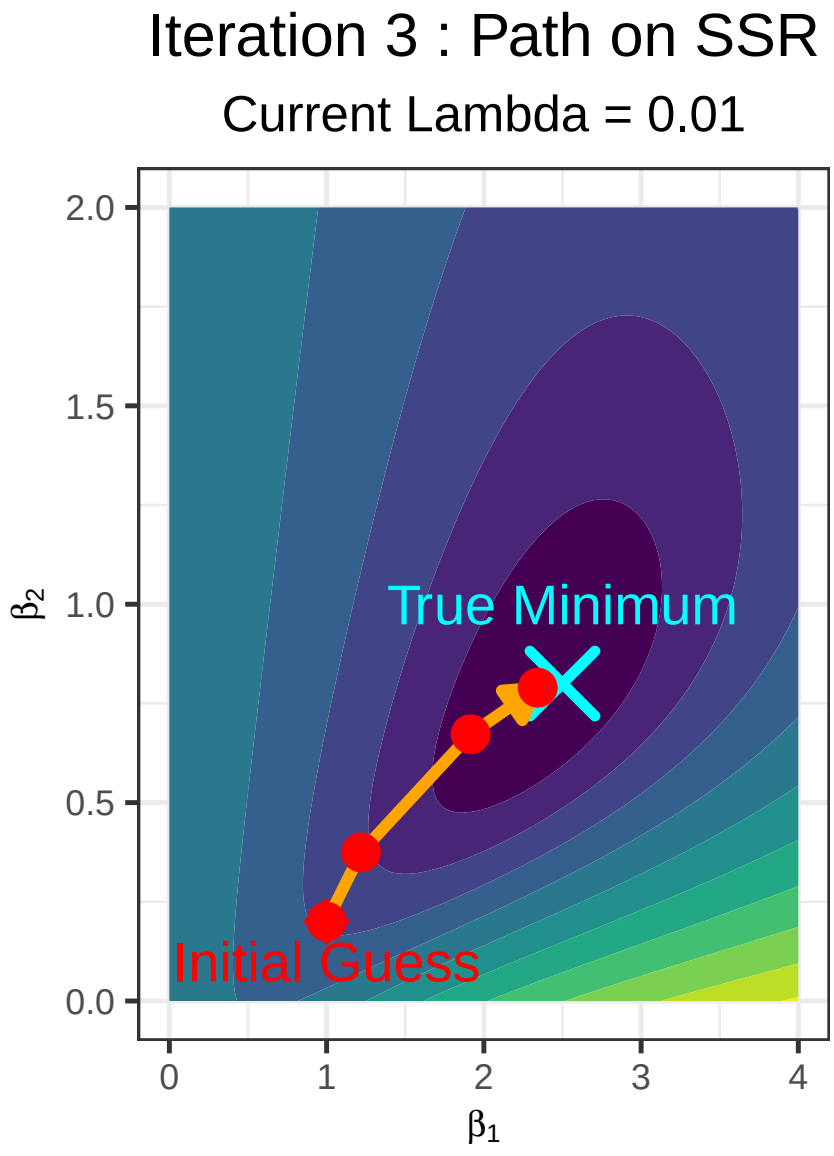
Slide A2i (Example): LM Iteration 2



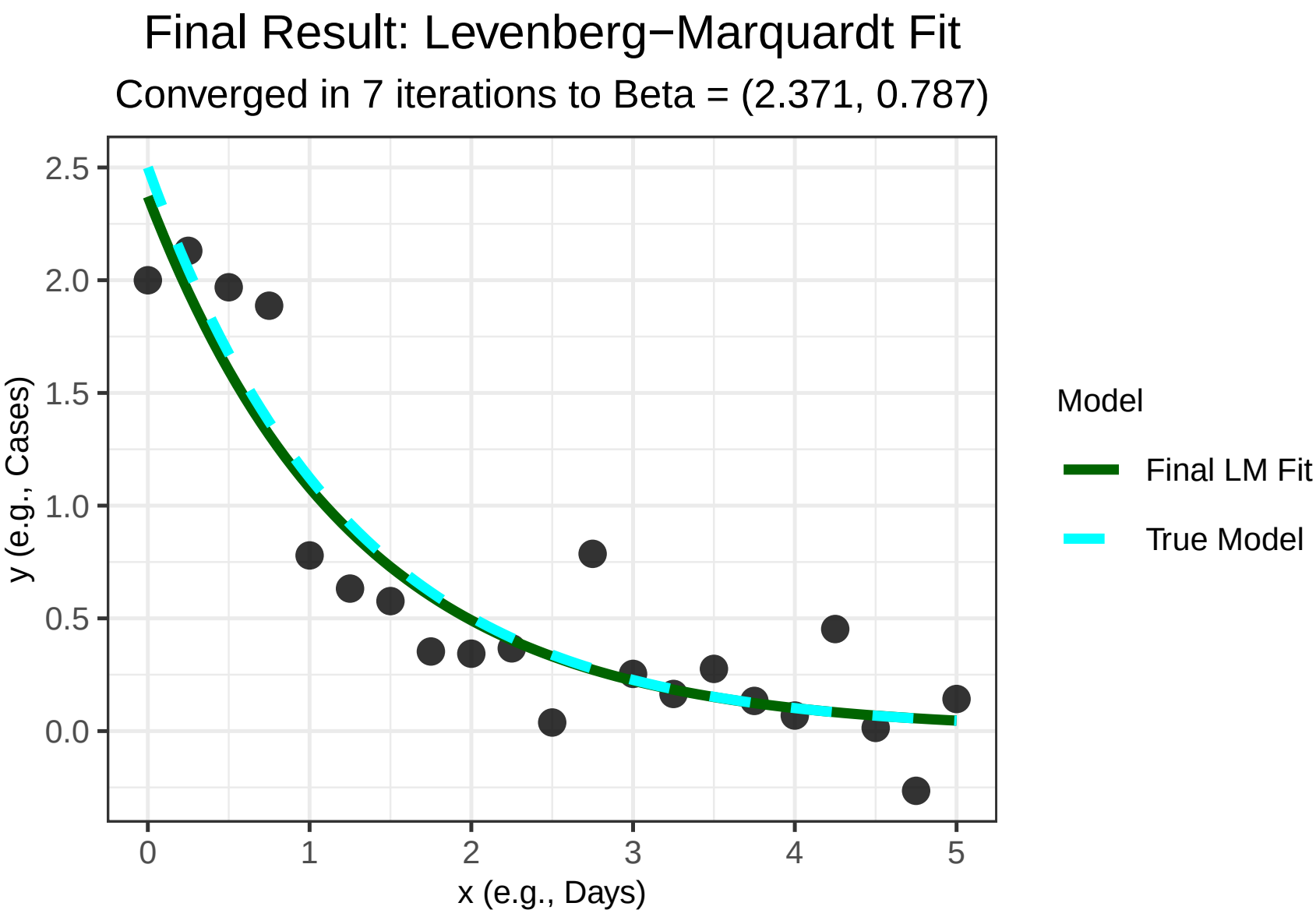


Slide A2j (Example): LM Iteration 3





Slide A2k (Example): Final Fit



References I

Blanchard, Philippe, and Erwin Brüning. 2015. "Inner Product Spaces and Hilbert Spaces." In *Mathematical Methods in Physics: Distributions, Hilbert Space Operators, Variational Methods, and Applications in Quantum Physics*, 213–25. Springer International Publishing. https://doi.org/10.1007/978-3-319-14045-2_15.

Dash, M., H. Liu, P. Scheuermann, and K. Tan. 2003. "Fast Hierarchical Clustering and Its Validation." *Data & Knowledge Engineering* 44: 109–38. https://doi.org/10.1016/S0169-023X(02)00138-6.

Ferreira, Laura, and David B. Hitchcock. 2009. "A Comparison of Hierarchical Methods for Clustering Functional Data." *Communications in Statistics - Simulation and Computation* 38 (9): 1925–49. https://doi.org/10.1080/03610910903168603.

Gautschi, Walter. 2011. *Numerical Analysis*. Birkhäuser Basel. https://link.springer.com/book/10.1007/978-0-8176-8259-0.

Gradshteyn, I. S., and I. M. Ryzhik. 2014. *Table of Integrals, Series, and Products*. Edited by Daniel Zwillinger. 8th ed. Amsterdam: Academic Press. https://www.sciencedirect.com/book/9780123849335/table-of-integrals-series-and-products.

Hubert, Lawrence, and Phipps Arabie. 1985. "Comparing Partitions." *Journal of Classification* 2 (1): 193–218. https://doi.org/10.1007/BF01908075.

Moré, Jorge J. 1978. "The Levenberg-Marquardt Algorithm: Implementation and Theory." In *Numerical Analysis*, edited by G. A. Watson, 630:105–16. Lecture Notes in Mathematics. Berlin, Heidelberg: Springer. https://doi.org/10.1007/BFb0067700.

References II

Müller, Hans-Georg. 2022. "Special Issue on 'Functional and Object Data Analysis': Guest Editor's Introduction." *Canadian Journal of Statistics* 50. https://doi.org/10.1002/cjs.11690.

Park, Y., L. Reichel, G. Rodriguez, and X. Yu. 2018. "Parameter Determination for Tikhonov Regularization Problems in General Form." *Journal of Computational and Applied Mathematics* 343: 12–25. https://doi.org/10.1016/j.cam.2018.04.017.

Patrikalakis, Nicholas M., Shin Maekawa, and Takashi Cho. 2009. "Section 1.4: Bspline Curves and Surfaces." https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/.

Postnikov, E. B., and O. V. Titkova. 2016. "A Correspondence Between the Models of Hodgkin-Huxley and FitzHugh-Nagumo Revisited." *The European Physical Journal Plus* 131: 411. https://doi.org/10.1140/epjp/i2016-16411-1.

Ramsay, J. O., Giles Hooker, D. Campbell, and J. Cao. 2007. "Parameter Estimation for Differential Equations: A Generalized Smoothing Approach." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69 (5): 741–96. https://doi.org/10.1111/j.1467-9868.2007.00610.x.

Ramsay, J. O., and B. W. Silverman. 2001. "Functional Data Analysis." In *International Encyclopedia of the Social & Behavioral Sciences*, edited by Neil J. Smelser and Paul B. Baltes, 5822–28. Oxford: Elsevier. https://doi.org/10.1016/B0080430767/00434-4.

———. 2005. *Functional Data Analysis, 2nd*. New York: Springer. https://doi.org/10.1002/0471667196.ess3138.

References III

Ramsay, James O., Giles Hooker, and Spencer Graves. 2009. *Functional Data Analysis with r and MATLAB*. Use r! New York: Springer. https://doi.org/10.1007/978-0-387-98185-7.

Rand, William M. 1971. "Objective Criteria for the Evaluation of Clustering Methods." *Journal of the American Statistical Association* 66 (336): 846–50. https://doi.org/10.2307/2284239.

Shahriari, Bobak, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. "Taking the Human Out of the Loop: A Review of Bayesian Optimization." *Proceedings of the IEEE* 104 (1): 148–75. https://doi.org/10.1109/JPROC.2015.2494218.

Shampine, Lawrence F. 1986. "Some Practical Runge–Kutta Formulas." *Mathematics of Computation* 46 (173): 135–50. https://doi.org/10.2307/2008219.

Weisstein, Eric W. 2002. "Fourier Series." https://mathworld.wolfram.com/FourierSeries.html.

Xu, Hongyan. 2020. "Chapter 7 - Big Data Challenges in Genomics." In *Principles and Methods for Data Science*, edited by Arni S. R. Srinivasa Rao and C. R. Rao, 43:337–48. Handbook of Statistics. Elsevier. https://doi.org/https://doi.org/10.1016/bs.host.2019.08.002.