Lecture 5: Roughness Penalties and Fitting Interpolating Splines

MATH5824 Generalised Linear and Additive Models

Richard P Mann

MATH5824 Generalised Linear and Additive Models

Reading

Course notes: Chapter 3, Sections 3.4–3.5

`www.richardpmann.com/MATH5824`

Why Natural Splines?

We showed that natural splines give a unique interpolant with m degrees of freedom.

There is a deeper justification: natural splines are **optimal** in a precise sense.

Idea: Among all smooth functions that interpolate the data, the natural spline is the *least wiggly*.

The Roughness Penalty

Roughness of a function f is measured by:

$$J_\nu(f) = \int_{-\infty}^{\infty} \left[f^{(\nu)}(t) \right]^2 dt$$

where $f^{(\nu)}$ is the νth derivative and $\nu \geq 1$.

Interpretation:

- $J_1(f)$: penalises variation in slope (total squared first derivative)
- $J_2(f)$: penalises curvature (total squared second derivative)

A "smooth" function has small $J_\nu(f)$; a "wiggly" function has large $J_\nu(f)$.

The Optimality Result

Proposition (3.2)

Among all functions f satisfying the interpolation conditions $f(t_i) = y_i$, the unique minimiser of $J_\nu(f)$ is a pth-order **natural spline**, where $p = 2\nu - 1$.

Key cases:

ν	$p = 2\nu - 1$	Spline type
1	1	Natural linear spline
2	3	Natural cubic spline

The cubic interpolating spline minimises integrated squared curvature — it is the smoothest interpolant in this sense.

Consequences of the Optimality Result

- **Linear interpolating splines** are shortest-path (piecewise linear) solutions — they minimise $J_1(f)$
- **Cubic interpolating splines** produce curves where knot locations are "invisible" — they minimise $J_2(f)$
- Higher-order splines (quintic, etc.) are rarely motivated in practice

Computational note: Solving the interpolation problem requires inverting an $n \times n$ matrix, which is $O(n^3)$ in general. Specialised algorithms exploit the banded structure to achieve $O(n)$.

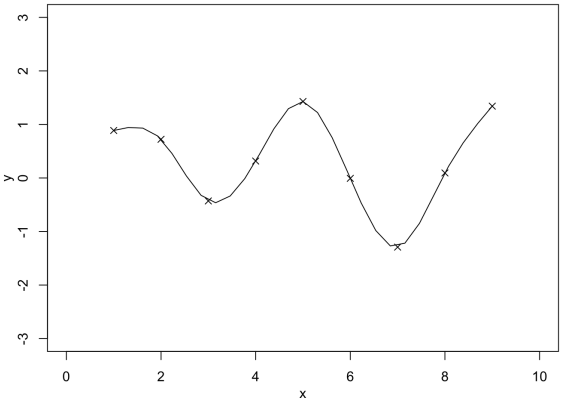
Fitting Interpolating Splines in R

Two functions:

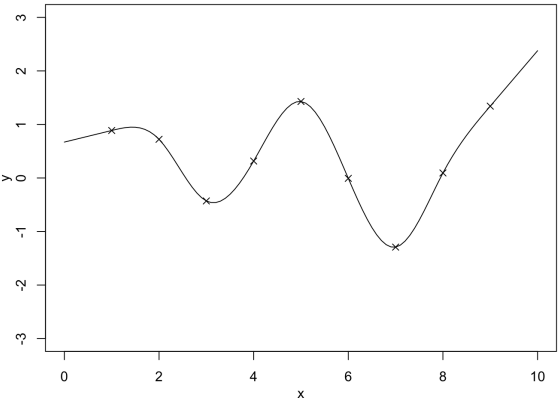
- `spline(x, y, method="natural")`: returns fitted values
- `splinefun(x, y, method="natural")`: returns a *function*

```r
# Using spline: returns values at a grid
myfit1 <- spline(x, y, method = "natural")
plot(x, y)
lines(myfit1)

# Using splinefun: returns a function object
myfit2 <- splinefun(x, y, method = "natural")
curve(myfit2, from = 0, to = 10, add = TRUE)
```

R Output: `spline` vs. `splinefun`



`spline()` — returns values



`splinefun()` — returns a function

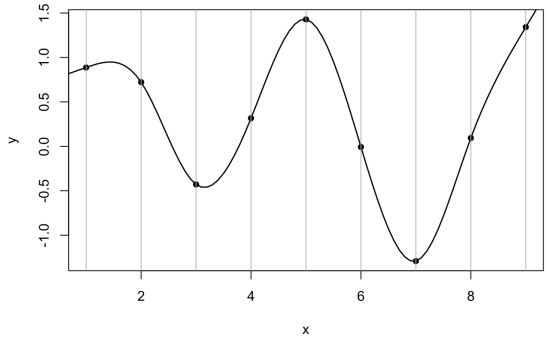
Key difference:

- `spline()` outputs a list of (x, y) pairs
- `splinefun()` outputs a function that can be evaluated anywhere, passed to `curve()`, ...

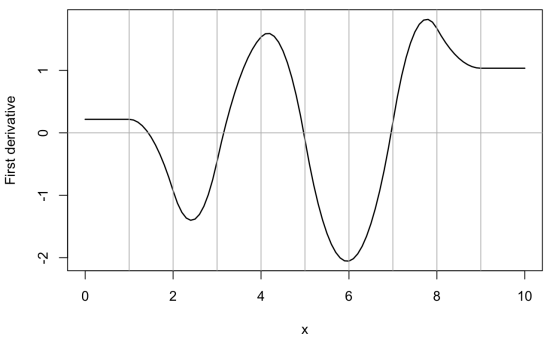
Prediction and Derivatives

```r
# Predict at specific locations
spline(x, y, xout = c(2.5, 7.5), method = "natural")

# Or using the function object
myfit2(c(2.5, 7.5))

# Compute derivatives
myfit2(x, deriv = 1)  # First derivative
myfit2(x, deriv = 2)  # Second derivative
myfit2(x, deriv = 3)  # Third derivative
```

Spline Derivatives: Function and First Derivative



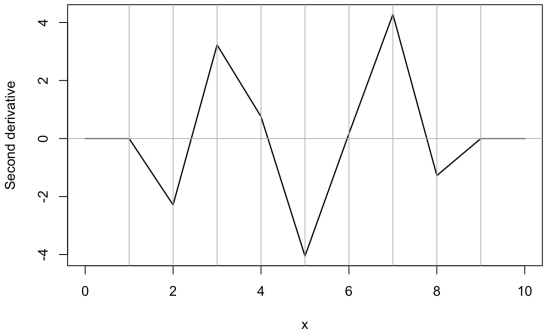
(a) Fitted spline $f(t)$



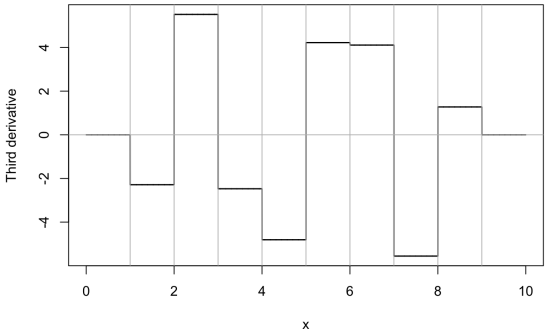
(b) First derivative $f'(t)$

Both f and f' are continuous everywhere, as expected for a cubic spline.

Spline Derivatives: Second and Third Derivative



(c) Second derivative $f''(t)$ — continuous



(d) Third derivative $f'''(t)$ — jumps at knots

The second derivative is continuous (piecewise linear). The third derivative has **jumps at knot locations** — this is where the cubic pieces join. Beyond the boundary knots, derivatives are constant or zero (natural boundary conditions).

Complete Example: Coal Seam Data

```r
# Coal seam data (missing value at x=6)
x <- c(0, 1, 2, 3, 4, 5, 7, 8, 9, 10)
y <- c(35, 41, 48, 47, 65, 67, 52, 51, 68, 75)

# Fit natural cubic interpolating spline
fit <- splinefun(x, y, method = "natural")

# Plot data and fitted curve
plot(x, y, pch = 19, xlab = "Location (km)",
     ylab = "Depth")
curve(fit, from = 0, to = 10, add = TRUE, col = "blue")

# Predict at missing location
fit(6)  # Predicted depth at x = 6
```

Summary

Key points:

- The roughness penalty $J_\nu(f) = \int [f^{(\nu)}]^2 \, dt$ measures "wiggliness"
- Natural splines minimise roughness among all interpolating functions
- $\nu = 2$ gives cubic natural splines (minimise curvature)
- In R: `spline()` for values, `splinefun()` for function objects
- `splinefun` can compute derivatives via `deriv` argument
- Cubic spline derivatives are continuous up to order 2; the third derivative jumps at knots

Next lecture: Smoothing splines — what to do when data have measurement error.