Lecture 2: Analytical Methods

Introduction to Data Analysis Techniques

1 Data Representation

Definition 1.1 (Finite Index Set Generator). [def:finite-index-set]
Finite index sets; real-analysis, pending transfer

The finite index set generator is the function

$$\mathrm{Fin} : \mathbb{N} \to \mathcal{P}(\mathbb{N}) \equiv n \mapsto \{i : \mathbb{N} \mid i < n\}$$

Thus $\mathrm{Fin}(n)$ contains the first n natural numbers, starting at 0; in particular, $\mathrm{Fin}(0) = \emptyset$ and $\mathrm{Fin}(3) = \{0, 1, 2\}$

LaTeX: `\Fin (n)`

Definition 1.2 (Pairing Function Generator). [def:pairing-function-generator]
Pairing indexed families; real-analysis, pending transfer

The pairing function generator takes $n : \mathbb{N}_1$ and returns an operation combining two indexed families $x, y : \mathrm{Fin}(n) \to \mathbb{R}$ into one indexed family with values in $\mathbb{R}\times\mathbb{R}$, representing n observations

$$\begin{aligned}\mathrm{pairGen} : \Pi n : \mathbb{N}_1.\, (\mathrm{Fin}(n) \to \mathbb{R}) &\to (\mathrm{Fin}(n) \to \mathbb{R}) \\ &\to \mathrm{Fin}(n) \to (\mathbb{R} \times \mathbb{R}) \\ &\equiv n \mapsto \big(x \mapsto \big(y \mapsto \big(i \mapsto (x(i), y(i))\big)\big)\big)\end{aligned}$$

LaTeX: `\mathrm {pairGen}(n)(x)(y)(i)`

Definition 1.3 (Pairing of Functions). [def:pairing-functions]
Pairing indexed families; real-analysis, pending transfer

For a fixed $n : \mathbb{N}_1$, define the pairing function by applying the pairing function generator to n

$$\begin{aligned}\mathrm{pair}_n : (\mathrm{Fin}(n) \to \mathbb{R}) &\to (\mathrm{Fin}(n) \to \mathbb{R}) \\ &\to \big(\mathrm{Fin}(n) \to (\mathbb{R} \times \mathbb{R})\big) \\ &\equiv \mathrm{pairGen}(n)\end{aligned}$$

For data fitting, $x(i)$ represents the input coordinate of observation i, and $y(i)$ represents its observed output, so

$$\mathrm{pair}_n(x)(y)(i) = (x(i), y(i))$$

LaTeX: `\mathrm {pair}_n(x)(y)(i)`

Definition 1.4 (Distinct Input Families). [def:distinct-input-families]
Injective indexed families; real-analysis, pending transfer

The function DistinctFamily assigns to each $n : \mathbb{N}$ the set of indexed families $x : \mathrm{Fin}(n) \to \mathbb{R}$ with pairwise distinct values

$$\begin{aligned}\mathrm{DistinctFamily} &: \Pi n : \mathbb{N}.\, \mathcal{P}(\mathrm{Fin}(n) \to \mathbb{R}) \\ &\equiv n \mapsto \left\{ x : \mathrm{Fin}(n) \to \mathbb{R} \;\middle|\; \begin{array}{c} \forall i, k : \mathrm{Fin}(n), \\ i \neq k \implies x(i) \neq x(k) \end{array} \right\}\end{aligned}$$

Thus $x : \mathrm{DistinctFamily}(n)$ means that x is injective

LaTeX: `\DistinctFamily (n)`

Convention 1.1 (Finite Index Ranges). [conv:finite-index-sum-product-notation] Real Analysis prerequisite: generalize finite summation and finite products to families defined only on a finite interval of natural numbers, allowing index 0 and empty intervals. Only indices in the interval should be evaluated; retain 0 for an empty sum and 1 for an empty product

Convention 1.2 (Data Fitting Indices). [conv:data-fitting-indices] In data fitting, $n : \mathbb{N}_1$ is the number of observations, indexed by $\mathrm{Fin}(n)$. The parameter $m : \mathbb{N}$ is the maximum polynomial degree, so there are $m+1$ coefficients, indexed by $\mathrm{Fin}(m+1)$
Finite data, vector coordinates, and matrix rows and columns are indexed from 0. A family of length k has indices $0, \ldots, k-1$

2 Lagrange Interpolation

Definition 2.1 (Interpolation). [def:interpolation] Methods to find a smooth function $f(x)$ that passes through n given points $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$

Definition 2.2 (Lagrange Interpolation). [def:lagrange-interpolation] For $n : \mathbb{N}_1$, take

$$x : \mathrm{DistinctFamily}(n), \qquad y : \mathrm{Fin}(n) \to \mathbb{R}$$

Lagrange interpolation for the paired observations $\mathrm{pair}_n(x)(y)$ seeks a polynomial $f : \mathbb{R} \to \mathbb{R}$ of degree at most $n-1$ such that

$$\forall i : \mathrm{Fin}(n), \quad f(x(i)) = y(i)$$

The values $y(i)$ may repeat

Theorem 2.1 (Unique Existence of Lagrange Interpolation). [thm:lagrange-existence-uniqueness] *For any $n : \mathbb{N}_1$ data points $(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})$ with pairwise distinct input coordinates, there exists a unique polynomial of degree at most $n-1$ that passes through all these points*

Convention 2.1 (Lagrange Interpolation Method). [conv:lagrange-interpolation-method] For data satisfying the Lagrange interpolation conditions, write $x_i \equiv x(i)$ and $y_i \equiv y(i)$ and construct the following system of equations:

$$\begin{cases} \sum_{0}^{n-1} (j \mapsto a_j \cdot x_0^j) = y_0 \\ \sum_{0}^{n-1} (j \mapsto a_j \cdot x_1^j) = y_1 \\ \vdots \\ \sum_{0}^{n-1} (j \mapsto a_j \cdot x_{n-1}^j) = y_{n-1} \end{cases} \implies \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \ldots & x_{n-1}^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix}$$

According to the Theorem in Linear Algebra , which states that a system of n linear equations with n unknowns has a unique solution if $\det A \neq 0$, where A is the coefficient matrix.
Since $x_0, \ldots, x_{n-1}$ are pairwise distinct, $\det A \neq 0$. Therefore, there exists a unique solution $\vec{a}$
So, we can find a unique polynomial $f(x) = \sum_{0}^{n-1} (j \mapsto a_j \cdot x^j)$ that passes through all the given points

Definition 2.3 (Lagrange Basis Polynomial Generator). [def:lagrange-base-polynomials-generator]

$$\begin{aligned}\text{LagrangeTerm} : \Pi n : \mathbb{N}_1.\, \text{DistinctFamily}(n) \to \text{Fin}(n) \to (\mathbb{R} \to \mathbb{R}) \\ \equiv n \mapsto x \mapsto j \mapsto \Bigg(t \mapsto \\ \prod_{0}^{j-1} \left(i \mapsto \frac{t - x(i)}{x(j) - x(i)} \right) \cdot \prod_{j+1}^{n-1} \left(i \mapsto \frac{t - x(i)}{x(j) - x(i)} \right) \Bigg)\end{aligned}$$

Here n is the number of observations, j selects the basis polynomial, and x supplies the input coordinates. The output is a function of $t : \mathbb{R}$, using the finite sum and product notation for the two index ranges

`LaTeX: \LagrangeTerm (n)(x)(j)(t)`

Redirect (Lagrange Basis Notation) [def:lagrange-base-polynomials]
Moved to convention 2.2.

Convention 2.2 (Lagrange Basis Notation). [conv:lagrange-base-polynomials] For fixed $n : \mathbb{N}_1$ and $x : \text{DistinctFamily}(n)$, write

$$L \equiv \text{LagrangeTerm}(n)(x), \qquad L_j \equiv L(j) \quad (j : \text{Fin}(n))$$

`LaTeX: \LagrangeFamily _j(t)`

Definition 2.5 (Lagrange Polynomial Generator). [def:lagrange-polynomial-generator]

$$\begin{aligned}\text{LagrangePoly} : \Pi n : \mathbb{N}_1.\, \text{DistinctFamily}(n) \to (\text{Fin}(n) \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R}) \equiv \\ n \mapsto x \mapsto y \mapsto t \mapsto \sum_{0}^{n-1} (j \mapsto y(j) \cdot \text{LagrangeTerm}(n)(x)(j)(t))\end{aligned}$$

`LaTeX: \LagrangePoly (n)(x)(y)(t)`

Definition 2.6 (Lagrange Interpolating Polynomial). [def:lagrange-interpolating-polynomial] For $n : \mathbb{N}_1$, $x : \text{DistinctFamily}(n)$, and $y : \text{Fin}(n) \to \mathbb{R}$, the Lagrange interpolating polynomial is

$$f : \mathbb{R} \to \mathbb{R} \equiv t \mapsto \text{LagrangePoly}(n)(x)(y)(t)$$

Theorem 2.2 (Interpolation at the Data Points). [thm:lagrange-interpolation-values] *Let $n : \mathbb{N}_1$, $x : \text{DistinctFamily}(n)$, and $y : \text{Fin}(n) \to \mathbb{R}$, and let f be the Lagrange interpolating polynomial for these inputs. Then*

$$\forall i : \text{Fin}(n), \quad f(x(i)) = y(i)$$

Proof. **TODO**(omitted) □

Remark 2.1 (Limitations of Exact Interpolation). [rem:lagrange-interpolation-limitations] When observations contain measurement errors, exact interpolation also fits those errors; this may cause *overfitting*

Remark 2.2 (Lagrange Construction Summary). [rem:lagrange-construction-summary] Fix $n : \mathbb{N}_1$ and $x : \text{DistinctFamily}(n)$. Each basis polynomial $L_j : \mathbb{R} \to \mathbb{R}$ uses the same input coordinates. For each $i : \text{Fin}(n)$ with $i \neq j$, form the factor

$$\frac{t - x(i)}{x(j) - x(i)}$$

Multiplying these $n-1$ factors gives $L_j(t)$. It equals 1 at $t = x(j)$ and 0 at the other input coordinates
Multiply each basis polynomial by its observed value $y(j)$ and sum

$$f(t) = \sum_{0}^{n-1} (j \mapsto y(j) \cdot L_j(t))$$

The resulting interpolating polynomial passes through all n observations, as stated in interpolation at the data points

3 Linear Least Squares Fit

Convention 3.1 (Finite Real Vectors). [conv:finite-real-vectors]
Finite real vectors; real-analysis, pending transfer
For $k : \mathbb{N}$, write

$$\mathbb{R}^k \equiv (\mathrm{Fin}(k) \to \mathbb{R})$$

For a coefficient vector $c : \mathbb{R}^k$, its coordinate indexed by $j : \mathrm{Fin}(k)$ is written $c(j)$
LaTeX: \RealVectors {k}

Definition 3.1 (Polynomial Model Generator). [def:linear-model-generator]
Polynomial functions from coefficient vectors; real-analysis, pending transfer
The polynomial model generator takes a maximum degree $m : \mathbb{N}$ and a coefficient vector $c : \mathbb{R}^{m+1}$, and returns a polynomial function

$$\begin{aligned}\mathrm{PolynomialModelGen} &: \Pi m : \mathbb{N}.\, \mathbb{R}^{m+1} \to (\mathbb{R} \to \mathbb{R}) \\ &\equiv m \mapsto c \mapsto \left(t \mapsto \sum_{0}^{m} \left(j \mapsto c(j) \cdot t^j \right) \right)\end{aligned}$$

LaTeX: \PolynomialModelGen (m)(c)(t)

Remark 3.1. The output has degree at most m. The model is linear in its coefficients

Example 3.1. [eg:straight-line-model] A straight-line model has two coefficients. For $c : \mathbb{R}^2$,

$$\mathrm{PolynomialModelGen}(1)(c) = t \mapsto c(0) + c(1) \cdot t$$

Definition 3.2 (Squared Error Generator). [def:squared-error] For $n : \mathbb{N}_1$ and indexed families $x, y : \mathrm{Fin}(n) \to \mathbb{R}$, the squared error generator returns an operator that assigns to each candidate function $f : \mathbb{R} \to \mathbb{R}$ the sum of the squared distances between its predicted outputs and the observations

$$\begin{aligned}\mathrm{SquaredError} &: \Pi n : \mathbb{N}_1.\, (\mathrm{Fin}(n) \to \mathbb{R}) \to (\mathrm{Fin}(n) \to \mathbb{R}) \\ &\to \big((\mathbb{R} \to \mathbb{R}) \to \mathbb{R} \big) \\ &\equiv n \mapsto x \mapsto y \mapsto f \mapsto \sum_{0}^{n-1} \left(i \mapsto d(f(x(i)), y(i))^2 \right)\end{aligned}$$

LaTeX: \SquaredError (n)(x)(y)(f)

Definition 3.3 (Polynomial Error Generator). [def:polynomial-error-generator] For n observations and a maximum polynomial degree m, the polynomial error generator assigns to each coefficient vector the sum of squared errors of its generated polynomial

$$\begin{aligned}\text{PolyError} : \Pi n : \mathbb{N}_1.\, \Pi m : \mathbb{N}.\, (\text{Fin}(n) \to \mathbb{R}) \to (\text{Fin}(n) \to \mathbb{R}) \to (\mathbb{R}^{m+1} \to \mathbb{R}) \\ \equiv n \mapsto m \mapsto x \mapsto y \mapsto c \mapsto \\ \text{SquaredError}(n)(x)(y)\big(\text{PolynomialModelGen}(m)(c)\big)\end{aligned}$$

Thus, after supplying the counts and observations,

$$\text{PolyError}(n)(m)(x)(y) : \mathbb{R}^{m+1} \to \mathbb{R}$$

`LaTeX: \PolyError (n)(m)(x)(y)(c)`

Remark 3.2 (Polynomial Error Intuition). [rem:polynomial-error-intuition] Given a coefficient vector c, $\text{PolyError}(n)(m)(x)(y)(c)$ uses $\text{PolynomialModelGen}(m)(c)$ to predict an output at each observed input $x(i)$. It squares the distance between each prediction and the observed output $y(i)$, then adds these values over all n observations. The result is the total squared error of the generated polynomial on the data

Convention 3.2 (Error Function for Fixed Data). [conv:least-squares-error-function] Fix $n : \mathbb{N}_1$, $m : \mathbb{N}$, and observations $x, y : \text{Fin}(n) \to \mathbb{R}$. Construct the coefficient-error function

$$S : \mathbb{R}^{m+1} \to \mathbb{R} \equiv \text{PolyError}(n)(m)(x)(y)$$

Remark 3.3. The notation S is local to these choices. In each derivation or program, construct S first.

Redirect (Comparing Two Straight-Line Fits) [eg:straight-line-squared-error]
Moved to example straight line squared error.

Definition 3.4 (Coordinate Partial Derivative). [def:coordinate-partial-derivative]
For $k : \mathbb{N}_1$, $j : \text{Fin}(k)$, and differentiable $f : \mathbb{R}^k \to \mathbb{R}$, define

$$\text{D}_j(f) : \mathbb{R}^k \to \mathbb{R} \equiv c \mapsto \lim_{h \to 0} \frac{f(c + h \cdot e_j) - f(c)}{h}$$

Here $e_j(i) \equiv 1$ if $i = j$, and 0 otherwise; vector operations act coordinatewise. Only coordinate j varies

`LaTeX: \PartialD _j(f)(c)`

Definition 3.5 (General Least Squares Fit). [def:linear-least-squares-error] Fix n, m, x, and y, and construct $S = \text{PolyError}(n)(m)(x)(y)$. If we propose $c : \mathbb{R}^{m+1}$ as a least-squares fit, then

$$\forall c' : \mathbb{R}^{m+1},\ S(c') \geq S(c)$$

Theorem 3.1 (Polynomial Least Squares Derivatives). [thm:straight-line-least-squares-equations] *Fix $n : \mathbb{N}_1$, $m : \mathbb{N}$, and observations $x, y : \text{Fin}(n) \to \mathbb{R}$, and construct $S \equiv \text{PolyError}(n)(m)(x)(y)$. For $c : \mathbb{R}^{m+1}$ and $j : \text{Fin}(m+1)$*
The vector c is a least-squares fit if and only if

$$\forall j : \text{Fin}(m+1), \quad \text{D}_j(S)(c) = 0$$

These give $m+1$ linear equations in the $m+1$ coefficients

Remark 3.4 (Proof Intuition). [rem:polynomial-least-squares-proof-intuition]

- The polynomial model is linear in its coefficients, so S is a quadratic function of those coefficients
- Differentiate each squared residual using the chain rule to obtain the displayed partial derivatives
- At a minimum, varying any one coefficient must give zero derivative, so every $\mathrm{D}_j(S)(c)$ vanishes
- Conversely, expand $S(c+h) - S(c)$ for any coefficient change h. If all partial derivatives vanish, the linear terms cancel and only nonnegative squares remain, so c is a global minimum

Theorem 3.2 (Normal Equations). [thm:linear-system-of-sloving-least-squares-fit] *Fix $n : \mathbb{N}_1$, $m : \mathbb{N}$, and $x, y : \mathrm{Fin}(n) \to \mathbb{R}$, and construct $S \equiv \mathrm{PolyError}(n)(m)(x)(y)$. A vector $c : \mathbb{R}^{m+1}$ is a least-squares fit if and only if it satisfies the* normal equations*: for every $j : \mathrm{Fin}(m+1)$,*

$$\sum_0^m \left(k \mapsto \left(\sum_0^{n-1} \big(i \mapsto x(i)^{j+k}\big) \right) \cdot c(k) \right) = \sum_0^{n-1} \big(i \mapsto y(i) \cdot x(i)^j\big)$$

For $r : \mathbb{N}$, abbreviate the known data sums by

$$A_r \equiv \sum_0^{n-1} \big(i \mapsto x(i)^r\big), \qquad b_r \equiv \sum_0^{n-1} \big(i \mapsto y(i) \cdot x(i)^r\big)$$

Since $A_0 = n$, the expanded system has the pattern

$$\begin{cases} n \cdot c(0) + \ldots + A_m \cdot c(m) = b_0 \\ A_1 \cdot c(0) + \ldots + A_{m+1} \cdot c(m) = b_1 \\ \qquad \vdots \\ A_m \cdot c(0) + \ldots + A_{2m} \cdot c(m) = b_m \end{cases}$$

Take rows $j = 0, \ldots, m$, with coefficients indexed by $k = 0, \ldots, m$. This is a linear system for $c(0), \ldots, c(m)$, solvable using Gaussian elimination

Pf. Derivation of the Normal Equations.
For a coefficient vector $a : \mathbb{R}^{m+1}$, write $p_a \equiv \mathrm{PolynomialModelGen}(m)(a)$. Unfolding the error generators gives

$$\begin{aligned} \mathrm{D}_j(S)(c) &= \mathrm{D}_j(\mathrm{PolyError}(n)(m)(x)(y))(c) \\ &= \mathrm{D}_j\big(a \mapsto \mathrm{SquaredError}(n)(x)(y)(p_a)\big)(c) \\ &= \mathrm{D}_j\left(a \mapsto \sum_0^{n-1} \left(i \mapsto |y(i) - p_a(x(i))|^2\right)\right)(c) \\ &= \mathrm{D}_j\left(a \mapsto \sum_0^{n-1} \left(i \mapsto \big(y(i) - p_a(x(i))\big)^2\right)\right)(c) \end{aligned}$$

The last equality uses $|u|^2 = u^2$ for real u
Only coefficient $a(j)$ varies when taking D_j, so

$$\mathrm{D}_j\big(a \mapsto p_a(x(i))\big)(c) = x(i)^j$$

The chain rule therefore gives

$$\begin{aligned}&\mathrm{D}_j\left(a \mapsto \big(y(i) - p_a(x(i))\big)^2\right)(c)\\ &\quad= 2 \cdot \big(y(i) - p_c(x(i))\big) \cdot \big(-x(i)^j\big)\\ &\quad= 2 \cdot \big(p_c(x(i)) - y(i)\big) \cdot x(i)^j\end{aligned}$$

The minus sign comes from differentiating $y(i) - p_a(x(i))$. Differentiating the finite sum term by term now gives

$$\mathrm{D}_j(S)(c) = 2 \cdot \sum_0^{n-1} \big(i \mapsto \big(p_c(x(i)) - y(i)\big) \cdot x(i)^j\big)$$

Set every partial derivative equal to zero and divide each equation by 2. For every $j : \mathrm{Fin}(m+1)$, this gives

$$\sum_0^{n-1}\left(i \mapsto \left(\sum_0^{m}\big(k \mapsto c(k) \cdot x(i)^k\big) - y(i)\right) \cdot x(i)^j\right) = 0$$

Expanding and collecting the coefficients gives the *normal equations*, one for each $j : \mathrm{Fin}(m+1)$:

$$\sum_0^{m}\left(k \mapsto \left(\sum_0^{n-1}\big(i \mapsto x(i)^{j+k}\big)\right) \cdot c(k)\right) = \sum_0^{n-1}\big(i \mapsto y(i) \cdot x(i)^j\big)$$

These are the normal equations stated above □

Corollary 3.1 (Straight-Line Fit). [cor:straight-line-normal-equations] *Fix $n : \mathbb{N}_1$ and $x, y : \mathrm{Fin}(n) \to \mathbb{R}$, and construct $S \equiv \mathrm{PolyError}(n)(1)(x)(y)$. The vector $c : \mathbb{R}^2$ gives a least-squares line $t \mapsto c(0) + c(1) \cdot t$ if and only if*

$$\begin{cases} n \cdot c(0) + \left(\sum_0^{n-1}\big(i \mapsto x(i)\big)\right) \cdot c(1) = \sum_0^{n-1}\big(i \mapsto y(i)\big) \\ \left(\sum_0^{n-1}\big(i \mapsto x(i)\big)\right) \cdot c(0) + \left(\sum_0^{n-1}\big(i \mapsto x(i)^2\big)\right) \cdot c(1) = \sum_0^{n-1}\big(i \mapsto x(i) \cdot y(i)\big) \end{cases}$$

Calculate the sums from the observations, then solve these two linear equations for $c(0)$ and $c(1)$

Proof. Substitute $m = 1$ and $j = 0, 1$ into the normal equations □

Corollary 3.2 (Line Through the Origin). [cor:origin-line-least-squares] *Fix $n : \mathbb{N}_1$ and $x, y : \mathrm{Fin}(n) \to \mathbb{R}$. For lines $t \mapsto a \cdot t$, construct*

$$S : \mathbb{R} \to \mathbb{R} \equiv a \mapsto \mathrm{SquaredError}(n)(x)(y)(t \mapsto a \cdot t)$$

If at least one $x(i)$ is nonzero, the unique minimizing slope is

$$a^* = \frac{\sum_0^{n-1}\big(i \mapsto x(i) \cdot y(i)\big)}{\sum_0^{n-1}\big(i \mapsto x(i)^2\big)}$$

In the straight-line model, this fixes $c(0) = 0$ and solves only for the slope $c(1) = a^*$

Proof. Set the derivative of S equal to zero and solve for a. The coefficient of a^2 is $\sum_0^{n-1}\big(i \mapsto x(i)^2\big) > 0$, so this stationary point is the unique minimum □

4 General Least Square Fit

Definition 4.1 (Model Family)**.** [def:model-family] For $k : \mathbb{N}_1$, a model family with k parameters is a function

$$F : \mathbb{R}^k \to (\mathbb{R} \to \mathbb{R})$$

A vector $c : \mathbb{R}^k$ supplies the settings; $F(c)$ is the resulting function, and $F(c)(t)$ is its output at input t

Remark 4.1. Fix $n : \mathbb{N}_1$ and observations $x, y : \mathrm{Fin}(n) \to \mathbb{R}$, and construct

$$S : \mathbb{R}^k \to \mathbb{R} \equiv c \mapsto \mathrm{SquaredError}(n)(x)(y)(F(c))$$

If $F(c)(t)$ is linear in the coefficients, the same derivative method produces a linear system whose solutions minimize S. For fixed functions $g : \mathrm{Fin}(k) \to (\mathbb{R} \to \mathbb{R})$, this includes $F(c)(t) = \sum_{0}^{k-1} \big(j \mapsto c(j) \cdot g(j)(t)\big)$, even when the functions $g(j)$ are not polynomials. For a general differentiable model family, setting all partial derivatives to zero finds stationary candidates; the equations may be nonlinear and minimality needs a separate check