Interpolation and extrapolation

Alexander Khanov

PHYS6260: Experimental Methods is HEP
Oklahoma State University

October 4, 2023

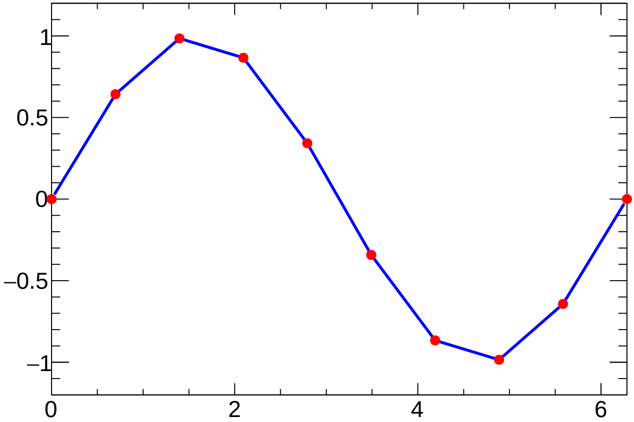
Interpolation/extrapolation vs fitting

- Formulation of the problem:
  - there are two quantities x and y related by some (generally unknown) functional dependence $y = \varphi(x)$
  - we have a set of N independent measurements of y: $\mathbf{y} = y_1, \ldots, y_N$ taken at N values of x: $\mathbf{x} = x_1, \ldots, x_N$
  - we want to evaluate y at arbitrary x
- Data fit:
  - $\mathbf{y} = y_1, \ldots, y_N$ have known variances $\sigma_1^2, \ldots, \sigma_N^2$
  - we want to find a function $f(x, \mathbf{p})$ that depends on $n < N$ parameters $\mathbf{p} = p_1, \ldots, p_n$ where $\mathbf{p}$ are chosen in some optimal way, e.g. minimize the overall difference between y_i and $f(x_i)$
  - methods: least squares, maximum likelihood
- Data interpolation/extrapolation:
  - we want to find a function $f(x)$ such that for all $i = 1, \ldots, N$: $f(x_i) = y_i$
  - the function must be “smooth” and not far away from $\varphi(x)$

Linear interpolation

- For each i, define $F_i(x) = \dfrac{(x - x_i)y_{i+1} + (x_{i+1} - x)y_i}{x_{i+1} - x_i}$
  - $F_i(x)$ is linear
  - $F_i(x_i) = y_i$, $F_i(x_{i+1}) = y_{i+1}$
- Simple, easy to calculate, well behaved
- No good if need to estimate derivatives

Linear interpolation (2)



Lagrange interpolation

- Since all "good" functions can be expanded in a Taylor series, it makes sense to take $f(x)$ as a polynomial
  - unless it's a degenerate case, having N coefficients is necessary and sufficient to satisfy N conditions $f(x_i) = y_i$
- How to construct such a polynomial?

Lagrange interpolation (2)

- Consider polynomial $P(x) = (x - x_1) \dots (x - x_N)$ of degree N with N real different roots $x_1, \dots, x_N$, then

$$P_i(x) = \frac{P(x)}{x - x_i} = (x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_N)$$

is a polynomial of degree $N - 1$ that is equal to zero at all $x = x_k \neq x_i$

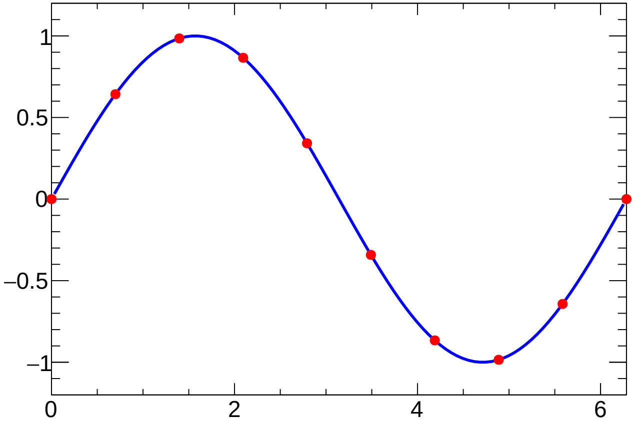
$$F_i(x) = \frac{P_i(x)}{P'(x_i)} = \frac{(x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_N)}{(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_N)}$$

is a polynomial of degree $N - 1$ that is equal to zero at all $x = x_k \neq x_i$ and equal to one at $x = x_i$

$$f(x) = \sum_{i=1}^{N} y_i F_i(x)$$

is a polynomial of degree $N - 1$ that satisfies $f(x_i) = y_i$ for all i

Lagrange interpolation (3)



How good is polynomial interpolation?

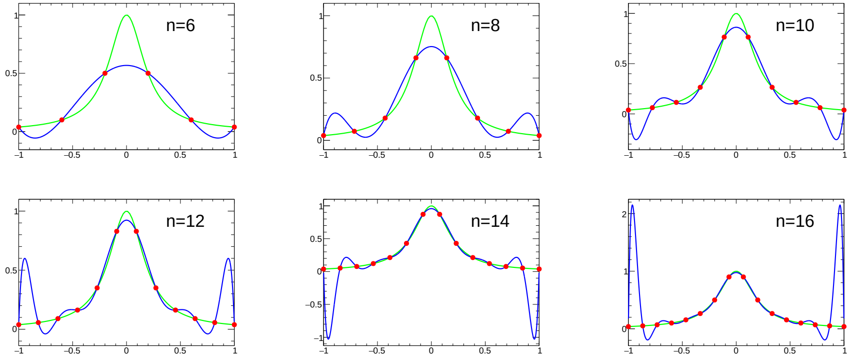
- Since we don't know $\varphi(x)$ we can't say for certain, but in general for any x there is ξ such that

$$\varphi(x) - f(x) = \frac{\varphi^{(N)}(\xi)}{N!} \prod_{i=1}^{N} (x - x_i)$$

- If we want to optimize the interpolation by adjusting the points x_i where we take the measurements, then we should try to minimize the maximum $\prod_{i=1}^{N} (x - x_i)$ (so called minimax principle)

Runge's phenomenon

- Depending on $\varphi(x)$, the polynomial interpolation can exhibit oscillating behavior as the number of points increases
- Typical example: $\varphi(x) = \dfrac{1}{1+25x^2}$



How to improve the interpolation?

- If the choice of x_i is up to us, one can mitigate bad interpolation behaviour by picking more points in the problematic region
  - Lagrange interpolation does not require that the roots x_i of the interpolating polynomial are equidistant
- Chebyshev polynomials:

$$T_n(x) = \cos(n \cos^{-1} x) \qquad -1 \leq x \leq 1$$

$$T_1(x) = 1 \qquad T_2(x) = x \qquad T_3(x) = 2x^2 - 1 \qquad T_4(x) = 4x^3 - 3x \qquad \ldots$$

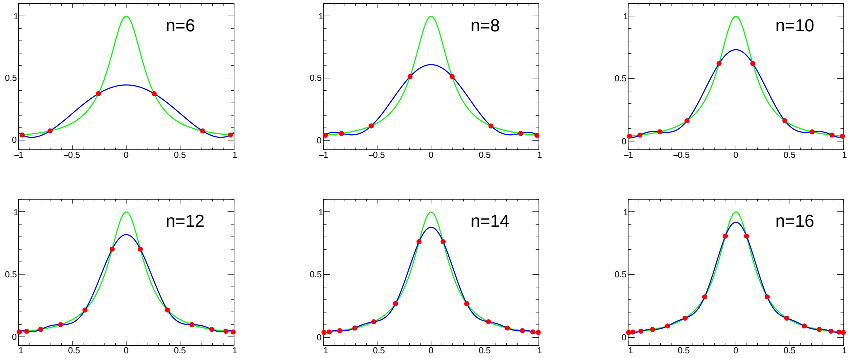
- The measurement points are taken to be the T_n roots:

$$x_i = \cos\left(\pi \frac{2i-1}{2N}\right),\ i = 1, \ldots, N$$

One can show that this choice of x_i is optimal in the minimax sense

Runge's phenomenon cured

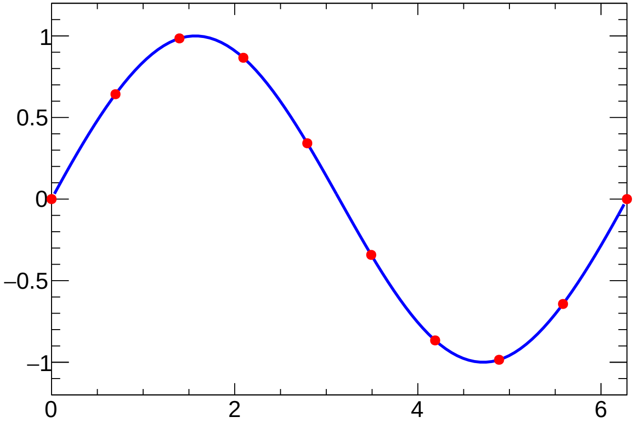
- By picking the sampling points at the roots of Chebyshev polynomials, the oscillating behavior is gone
- Same example: $\varphi(x) = \dfrac{1}{1+25x^2}$



Cubic splines

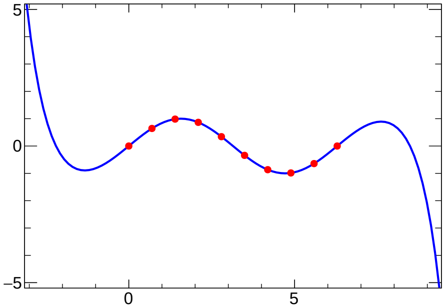
- The polynomial interpolation provides a "single formula" solution with nice mathematical properties but it's often an overkill
  - a simpler solution: a piecewise low degree polynomial that provides a smooth curve
- Cubic spline: for each interval $x_i < x < x_{i+1}$, define a cubic polynomial $F_i(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$
  - four unknowns $\Rightarrow$ need four equations
- Two conditions per point come from $F_i(x_i) = y_i$, $F_i(x_{i+1}) = y_{i+1}$
- Two more conditions per point make sure that first and second derivatives are continuous at join points
- One also needs two more conditions at the end points
  - e.g. "natural spline" $F_1''(x_1) = 0$, $F_{N-1}''(x_N) = 0$
- Given the x_i, y_i, the spline coefficients are found from a system of linear equations
- It's also possible to do a combination of splines and fitting when the spline join points ("knots") are chosen different from the actual measurement points

Cubic spline interpolation



Extrapolation

- The problem: want to know $\varphi(x)$ outside the data range (for $x < x_1$ or $x > x_N$)
- Can use polynomial extrapolation, but no error estimate is available!



Multivariate interpolation

- Linear interpolation can be easily extended to any number of dimensions
- 1-d: for each cell (x_0, x_1) define

$$F(x) = \frac{x_1 - x}{x_1 - x_0} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1)$$

- 2-d: for each cell (x_0, x_1, y_0, y_1) define

$$\begin{aligned} F(x, y) = & \frac{(x_1 - x)(y_1 - y)}{(x_1 - x_0)(y_1 - y_0)} f(x_0, y_0) + \frac{(x_1 - x)(y - y_0)}{(x_1 - x_0)(y_1 - y_0)} f(x_0, y_1) \\ & + \frac{(x - x_0)(y_1 - y)}{(x_1 - x_0)(y_1 - y_0)} f(x_1, y_0) + \frac{(x - x_0)(y - y_0)}{(x_1 - x_0)(y_1 - y_0)} f(x_1, y_1) \end{aligned}$$

- ...
- Note that the multilinear interpolation is not linear anymore!

Multivariate interpolation (2)

- RBF (radial basis function) interpolation

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \, \psi(|\mathbf{x} - \mathbf{x}_i|)$$

  - $\psi(r)$ is "radial basis function" (a function that depends on the distance between the points but not on the points themselves)
  - typical choice is $\psi(r) = \sqrt{1 + (\epsilon r)^2}$, where ϵ is a small number

- weights w_i can be found from a system of linear equations

$$\begin{bmatrix} \psi(|\mathbf{x}_1 - \mathbf{x}_1|) & \dots & \psi(|\mathbf{x}_N - \mathbf{x}_1|) \\ \dots & & \dots \\ \psi(|\mathbf{x}_1 - \mathbf{x}_N|) & \dots & \psi(|\mathbf{x}_N - \mathbf{x}_N|) \end{bmatrix} \begin{bmatrix} w_1 \\ \dots \\ w_N \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_1) \\ \dots \\ f(\mathbf{x}_N) \end{bmatrix}$$

- The method works well for randomly placed points, no need for a grid

Multivariate interpolation (3)

- Many other methods
  - splines can be extended to > 1 dimensions
  - Bézier surfaces: kind of splines widely used in computer graphics and CAD
- A popular technique in high energy physics: "morphing"
  - data analyses typically rely on shapes on kinematical distributions obtained with Monte Carlo that may depend on various model parameters
  - since Monte Carlo production is resource consuming, there is a need to interpolate between the limited number of available templates
- arXiv:1410.7388 [physics.data-an]: Interpolation between multi-dimensional histograms using a new non-linear moment morphing method

Data smoothing

- "Fitting" data without using any explicit fitting function
- The idea is to get rid of noise while preserving the important pattern properties
- Usual smoothing techniques:
  - running medians, e.g. $z_i = \mathrm{median}(y_{i-1}, y_i, y_{i+1})$
  - running means, e.g. $z_i = (y_{i-1} + y_i + y_{i+1})/3$, $z_i = y_{i-1}/4 + y_i/2 + y_{i+1}/4$
  - Savitzky-Golay filter: running weighted means based on least squares fit of points inside a window (so that the number of points inside the window is greater than the degree of the polynomial)
  - 353QH: an algorithm developed in the 70's and implemented in PAW's (and then ROOT's) histogram smoothing, a combination of running medians, quadratic interpolation, and Hanning (weighted) means
- Usually smoothing is neither desirable nor permissible ("Numerical methods in C")