Approximation of discrete data: Interpolation/Extrapolation

(離散データの近似: 補間/補外）

Interpolation

Pattern 1: Reproduce all sample points (標本点を必ず通る)

n sample points are reproduced by (*n* – 1) order polynomial.

- Interpolated data might be scattered largely in particular for orders higher than 3 (Runge's phenomenon/oscillation **ルンゲの現象**).

補間点が大きく振動する問題がでる。特に3次以上の多項式

=> To suppress the Runge's phenomenon:

Make the *n*-th order differentiations continuous at the boundaries between neighboring regions

=> Spline function

n sample points are reproduced by (*n* + *N* – 1) order polynomial.

Pattern 2: Smoothing (平滑化)

Scattering of data will be reduced

Pattern 3: Does not reproduce sample points exactly, but the deviation will be minimized

(標本点を通らないが、補間データは標本点から大きく外れない)

- Least-squares method (LSQ, 最小二乗法)
- Minimax approximation (ミニマックス近似)

Polynomial that reproduces sample points
(標本点を通る多項式)

n sample points (x_i, y_i) $(i = 1, \cdots, n)$ are reproduced by $(n - 1)$ order polynomial.

$$y_i = \sum_{k=0}^{n-1} a_k {x_i}^k \qquad (i = 1, \cdots, n)$$

$$\begin{pmatrix} 1 & x_1 & {x_1}^2 & \cdots & {x_1}^{n-1} \\ 1 & x_2 & {x_2}^2 & & {x_2}^{n-1} \\ 1 & x_3 & {x_3}^2 & & {x_3}^{n-1} \\ \vdots & & & \ddots & \\ 1 & x_n & {x_n}^2 & & {x_n}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

$|x_i| > 1$ might cause overflow,
$|x_i| < 1$ might cause underflow errors.

=> Normalize (正規化) the x range e.g. to [-1, 1] : $x_i' = 2\frac{x_i - x_{\mathrm{mid}}}{x_{\max} - x_{\min}}$

Standardize (標準化) by average and standard deviation: $x_i' = 2\frac{x_i - x_{\mathrm{average}}}{\sigma_x}$

Lagrange interpolation formula

(ラグランジの補間公式)

戸田英雄, 小野令美, 入門 数値計算, オーム社 (昭和58年)

(n – 1) order polynomial that reproduces n sample points (x_i, y_i) (i = 0, ･･･, n – 1) is determined uniquely.

Lagrange interpolation formula

$$P_{n-1}(x) = f(x_0)\phi_0(x) + f(x_1)\phi_1(x) + \cdots f(x_{n-1})\phi_{n-1}(x)$$

$$\phi_i(x) = \frac{\prod_{k\neq i}^{n-1}(x-x_k)}{\prod_{k\neq i}^{n-1}(x_i-x_k)} = \prod_{k\neq i}^{n-1}\frac{(x-x_k)}{(x_i-x_k)}$$

n = 2:

$$P_1(x) = f(x_0)\frac{(x-x_1)}{(x_0-x_1)} + f(x_1)\frac{(x-x_0)}{(x_1-x_0)}$$

n = 3:

$$P_2(x) = f(x_0)\frac{(x-x_1)}{(x_0-x_1)}\frac{(x-x_2)}{(x_0-x_2)} + f(x_1)\frac{(x-x_0)}{(x_1-x_0)}\frac{(x-x_2)}{(x_1-x_2)} + f(x_2)\frac{(x-x_0)}{(x_2-x_0)}\frac{(x-x_1)}{(x_2-x_1)}$$

Lagrange interpolation formula: Practical implementation

(実装の場合)

interpolate_lagrange.py: Lagrange基底を定義式どおり計算

実用的実装 (効率的・安定に計算):

interpolate_lagrange_barycentric.py

$$w_i = \frac{1}{\prod_{k=0,k\neq i}^{n-1}(x_i - x_k)}$$

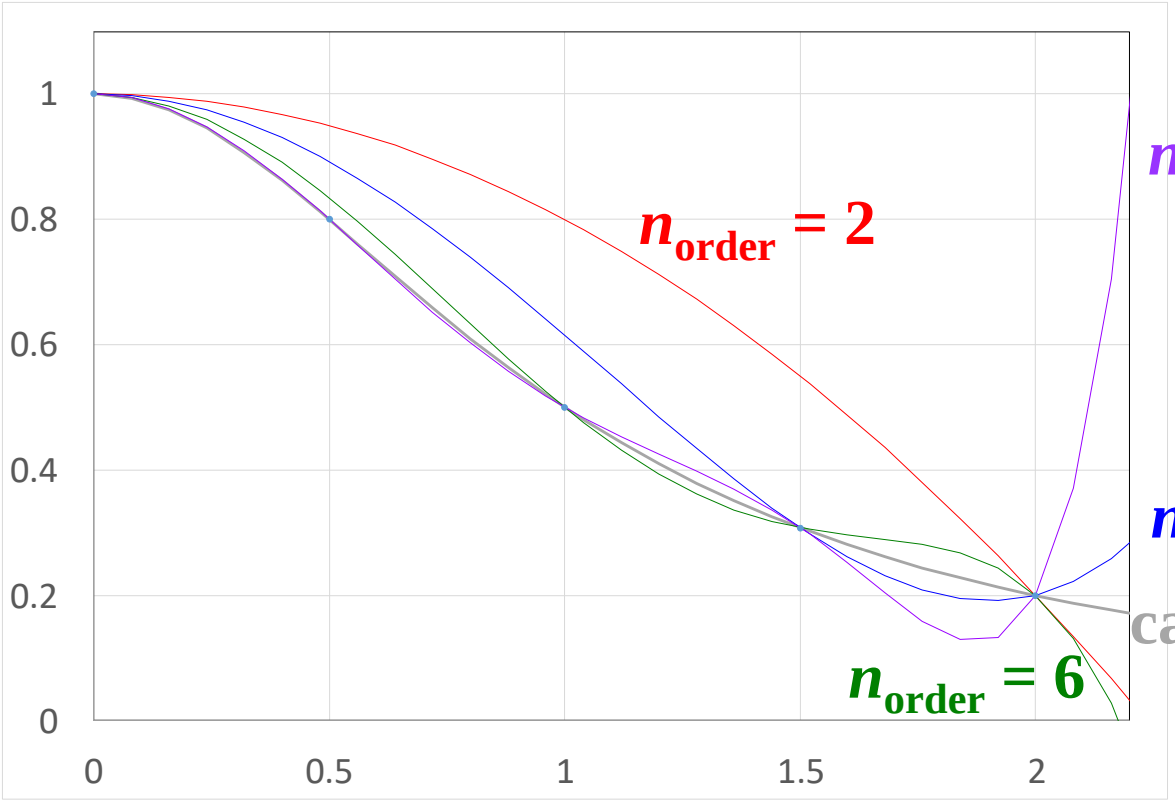
$$P_{n-1}(x) = \sum_i \frac{w_i y_i}{x - x_i} \Big/ \sum_i \frac{w_i}{x - x_i}$$

Problem of such polynomials

- Increasing the sample points will change the coefficients of polynomial completely.
- **Runge's phenomenon / oscillation (ルンゲの現象)**

  High order (e.g. >3) polynomial will cause large oscillations at points other than the sample points (高次の多項式では標本点以外で大きく振動することがある)

***Ex.* Interpolate $f(x) = 1 / (1 + x^2)$ for $(n_{order}+1)$ points in the range $x = [-2, 2]$**



In the machine learning (機械学習):

Overfitting (過適合), Overlearning (過学習)

Interpolation: Piecewise polynomial interpolation

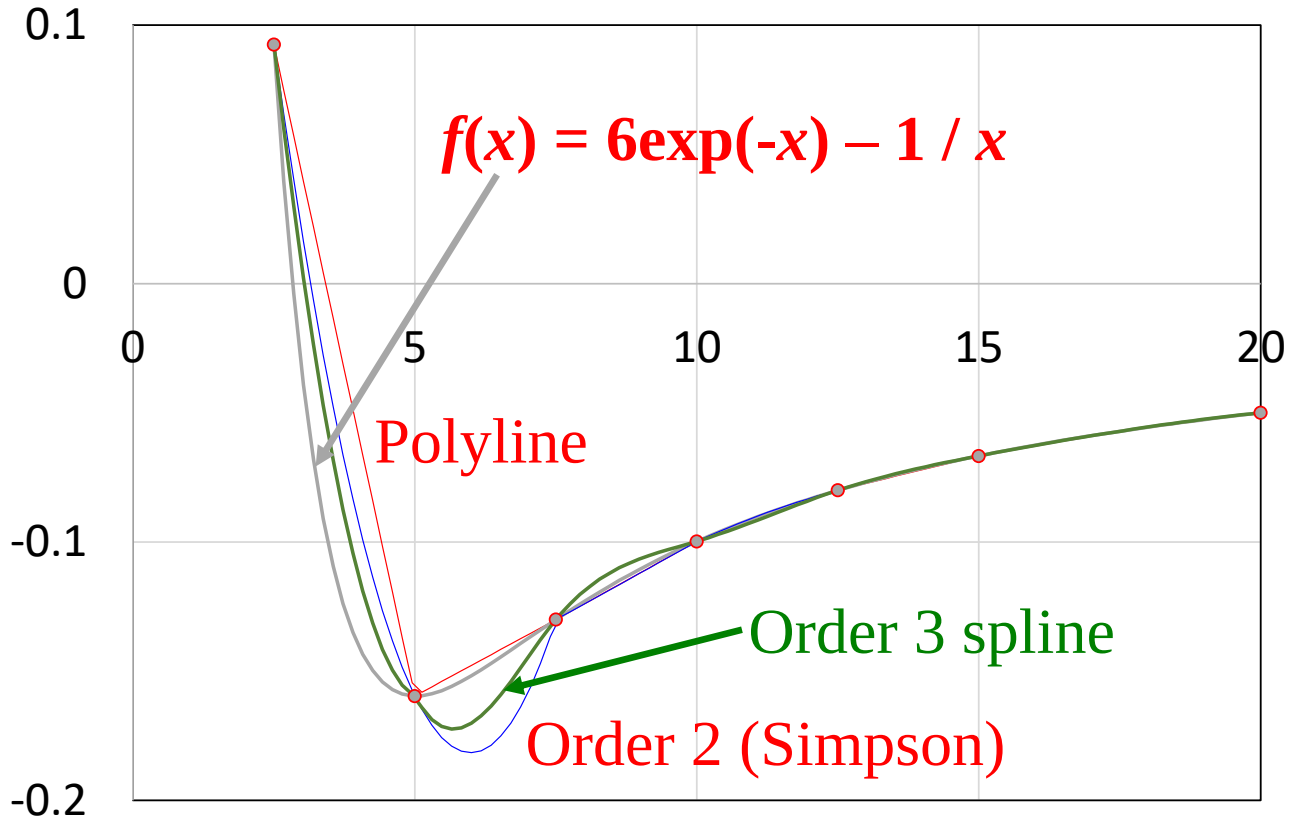
(区分多項式補間)

Connect divided sections by polylines (折れ線)

=> First derivatives will be discontinuous at the boundaries

=> (n – 1)-th derivatives are continuous for whole range:

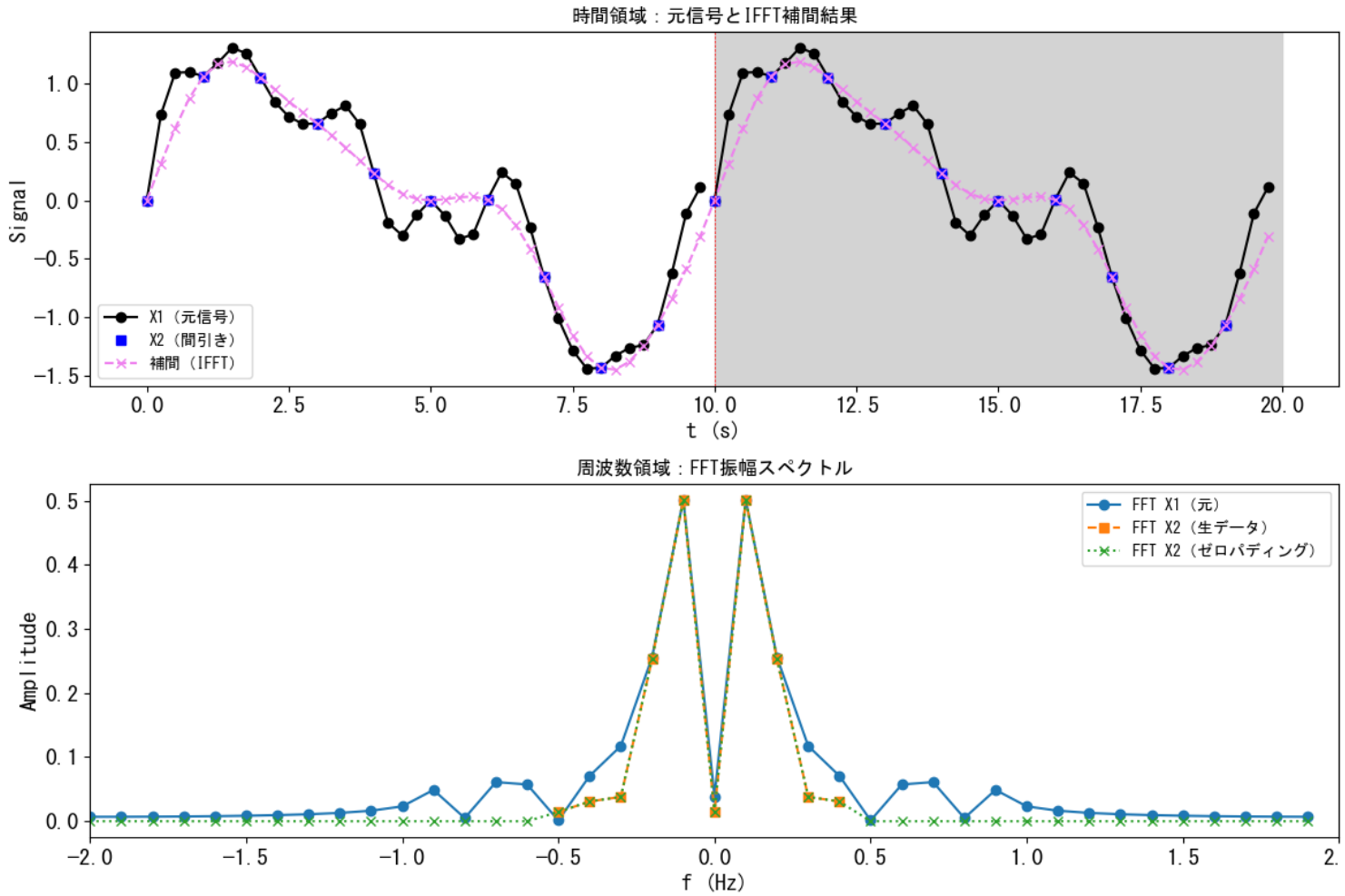
Order *n* spline functions (n次のスプライン関数)



Interpolation by FFT (Fast Fourier Transformation)

compare_FFT.py

1. Fourier transform $S(t)$ to $S^*(f)$
2. Increase the number of data by adding zeros in the high $|f|$ region (zero padding)
3. Inverse Fourier transform



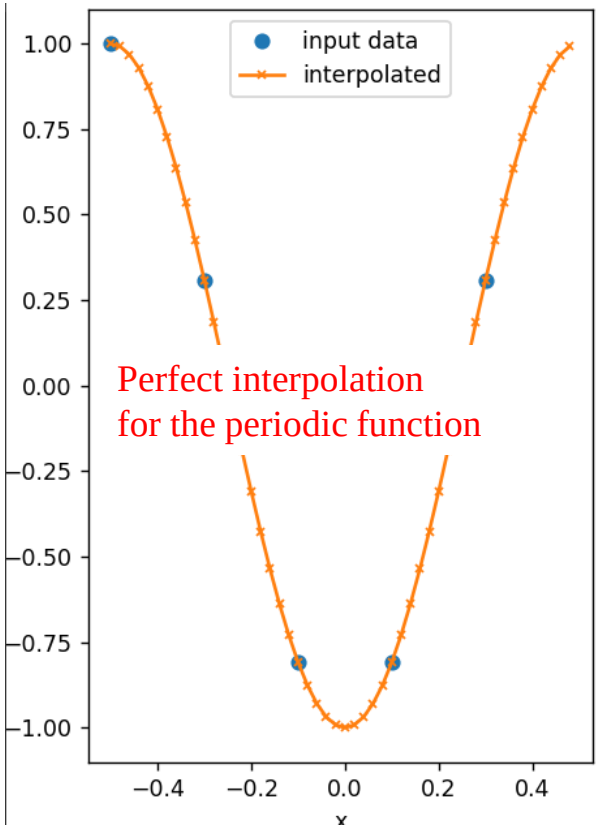
Interpolation by FFT (Fast Fourier Transform)

For a periodic function E(k) (like an energy function $E(k)$ of crystal), interpolation can be carried out by:

1. Fourier transform $E(k)$ to $E^*(x)$
2. Increase the number of data by adding zeros in the high $|x|$ region (padding)
3. Inverse Fourier transform

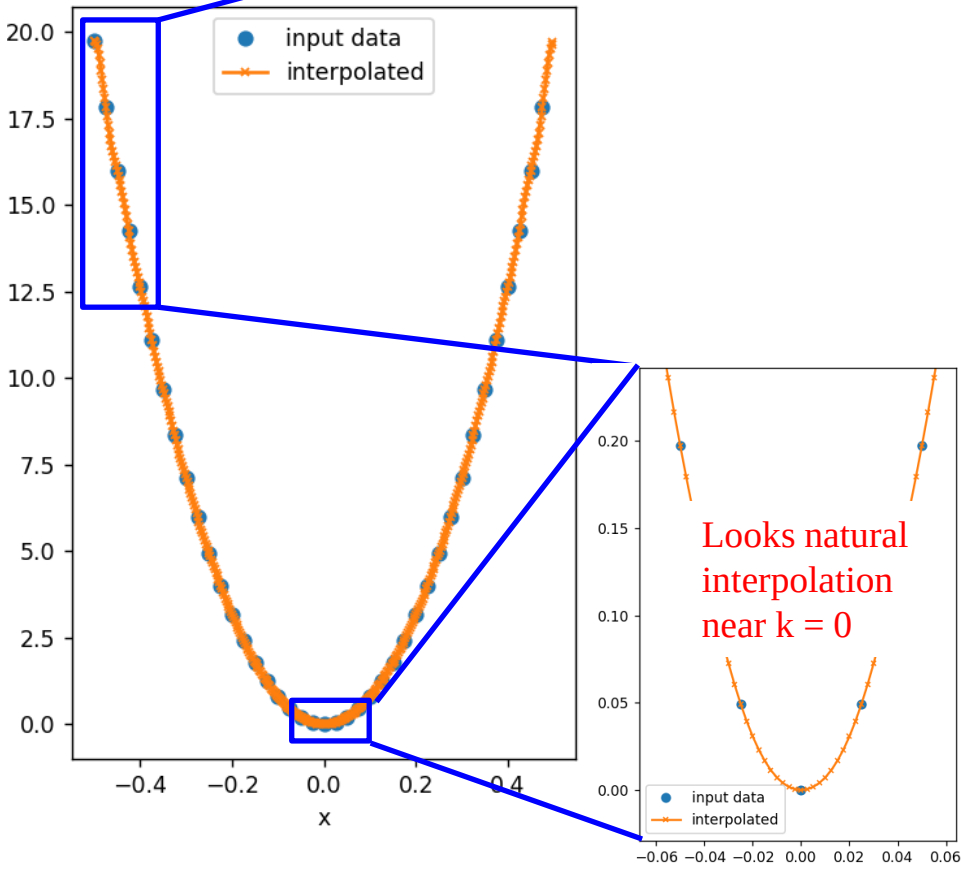
> python interpolate_fft.py generate

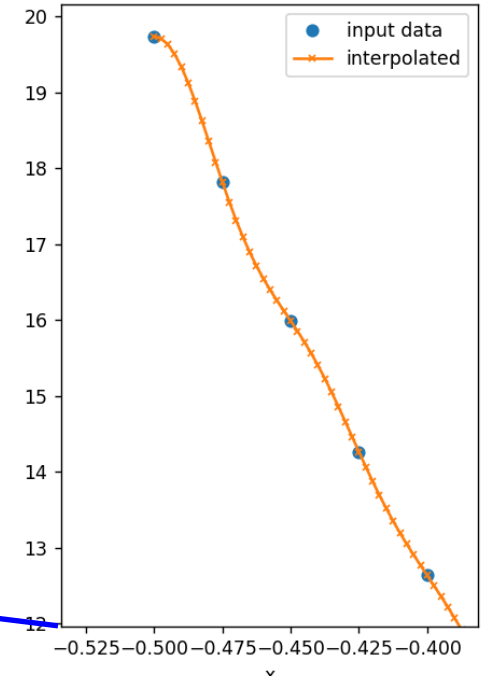
data: interpolate_fft.test.xlsx: TB band



> python interpolate_fft.py band_free_e.xlsx

data: band_free_e.xlsx: Free electron band





Interpolated data become a periodic function, producing unnatural deformation near the periodic boundaries