Numerical Methods for Computational Science and Engineering

Autumn Semester 2018, Week 6

Prof. Rima Alaifari, SAM, ETH Zurich

Question from last week: Periodic convolution with different periods

4-periodic & 6-periodic convolution of

$$\underline{x} = [2,1,3,2]^T \quad \text{with} \quad \underline{h} = [1,1,2]^T$$

$$p^4 = (\dots, 1,1,2,0,1,1,2,0,1,1,2,0,\dots)$$

$$p^6 = (\dots, 1,1,2,0,0,0,1,1,2,0,0,0,\dots)$$

$$x^4 = (\dots, 2,1,3,2,2,1,3,2,2,1,3,2,\dots)$$

$$x^6 = (\dots, 2,1,3,2,0,0,2,1,3,2,0,0,\dots)$$

- period 4:

$$y^4 = x^4 *_4 p^4, \quad y_k^4 = \sum_{j=0}^{3} p_{k-j}^4 \, x_j^4$$

$$y^4 = (\dots, 10,7,8,7,10,7,8,7,\dots)$$

$$y_0^4 = x_0 p_0^4 + x_1 p_{-1}^4 + x_2 p_{-2}^4 + x_3 p_{-3}^4$$

$$= x_0 h_0 + x_1 h_3 + x_2 h_2 + x_3 h_1 = 10$$

$$y_1^4 = 7$$

$$y_2^4 = 8, \quad y_3^4 = 7$$

• period 6:

$$y_k^6 = \sum_{j=0}^{5} p_{k-j}^6 \, x_j^6$$

$$y_0^6 = x_0^6 p_0^6 + x_1^6 p_{-1}^6 + x_2^6 p_{-2}^6 + x_3^6 p_{-3}^6 + \underbrace{x_4^6 p_{-4}^6 + x_5^6 p_{-5}^6}_{=0}$$

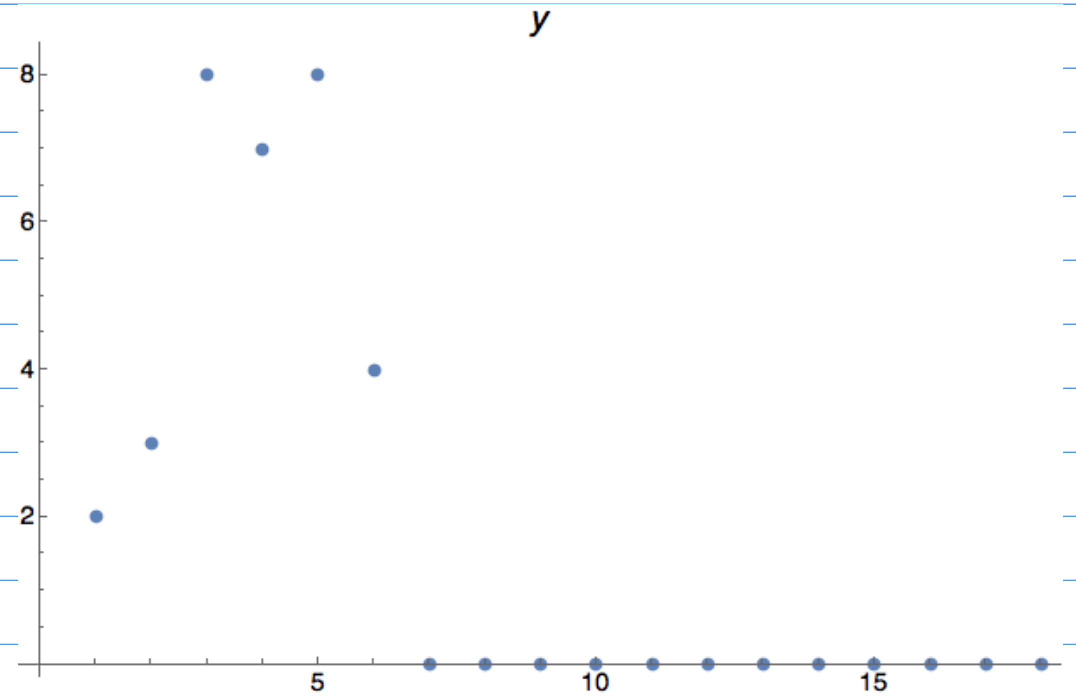
$$= x_0^6 h_0 + \underbrace{x_1^6 h_5 + x_2 h_4 + x_3 h_3}_{=0}$$

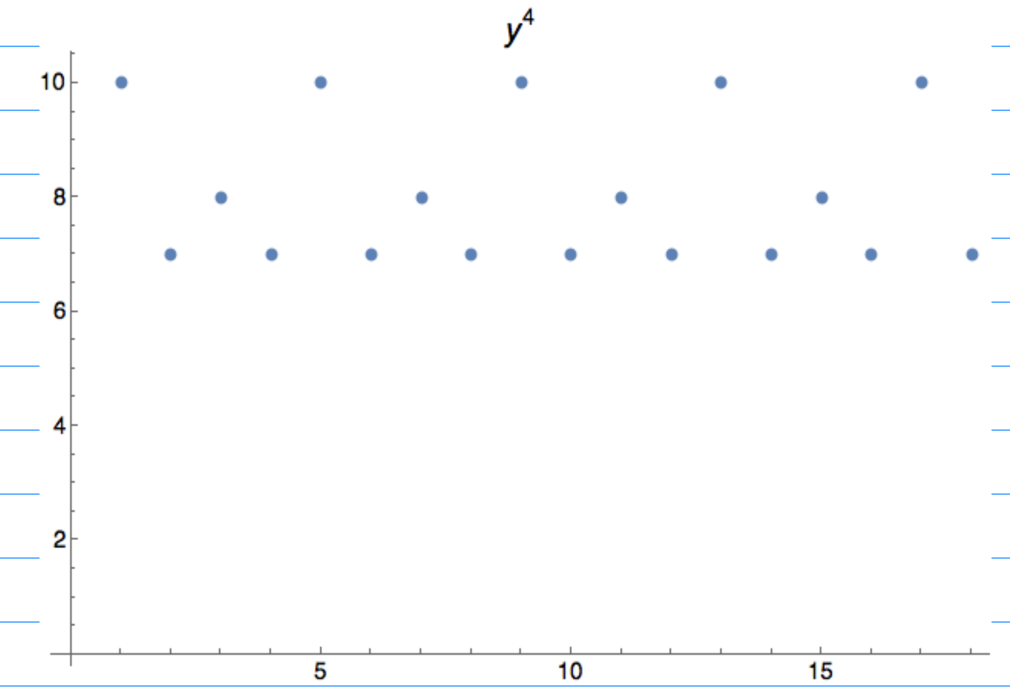
$$= 2$$

$$y^6 = (\ldots, 2, 3, 8, 7, 8, 4, 2, 3, 8, 7, 8, 4, \ldots)$$

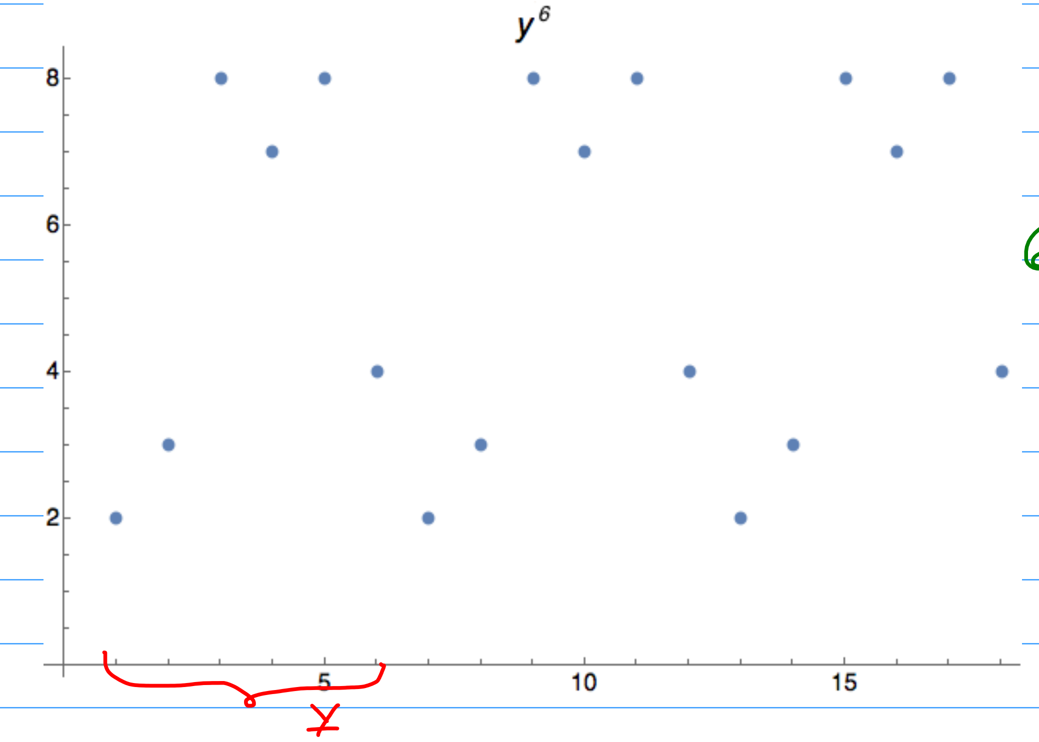
Note: linear convolution of $\underline{x} * \underline{h}$ would give $\underline{y} = [2, 3, 8, 7, 8, 4]^T$

Linear discrete convolution





4-periodic convolution



6-periodic convolution

Confirm: linear discr. convolution can be expressed as periodic convolution if the period is suff. long.

Thus: linear convolution can be written as multiplication by a circulant matrix

The Discrete Fourier Transform

Circulant matrices of a fixed size $n \times n$ have their eigenvectors in common:

Define the vectors $v_k := \left[\omega_n^{-jk}\right]_{j=0}^{n-1} \in \mathbb{C}^n$

$k \in \{0, \dots, n-1\}$

where $\omega_n := e^{-2\pi i/n}$

$$\omega_n^{\ell} = e^{-2\pi i \ell/n}, \quad \omega_n^{n} = e^{-2\pi i} = 1$$

$$\omega_n^{-\ell} = \overline{\omega_n^{\ell}} \quad \text{(complex conjugate)}$$

$$\omega_n^{jk} = (\omega_n^{j})^{k} = (\omega_n^{k})^{j}$$

$$v_k := \begin{bmatrix} 1 \\ e^{2\pi i k/n} \\ e^{4\pi i k/n} \\ \vdots \\ e^{2\pi i k(n-1)/n} \end{bmatrix}$$

One can show: For any circulant matrix $\in \mathbb{C}^{n,n}$ its eigenvectors are given by $\{v_0, \dots, v_{n-1}\}$

Thus: Eigenvectors are independent of the circulant matrix itself

[Two circulant matrices $C_1, C_2 \in \mathbb{C}^{n,n}$

- have the same set of eigenvectors
- only their eigenvalues differ

$$C_1 v_k = \lambda_k v_k$$
$$C_2 v_k = \tilde{\lambda}_k v_k$$

$\{v_0, \dots, v_{n-1}\}$ is called trigonometric basis of $\mathbb{C}^n$.

$\Rightarrow$ Eigenvalue decomposition of C with $Cv_k = \lambda_k v_k$ can be written compactly as:

$$\begin{bmatrix} | & | & & | \\ Cv_0 & Cv_1 & \cdots & Cv_{n-1} \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \lambda_0 v_0 & \lambda_1 v_1 & \cdots & \lambda_{n-1} v_{n-1} \\ | & | & & | \end{bmatrix} \quad (*)$$

Define $U_n = \begin{bmatrix} | & | & & | \\ v_0 & v_1 & \cdots & v_{n-1} \\ | & | & & | \end{bmatrix}$

(*) becomes:

$$C U_n = \underbrace{U_n \operatorname{diag}(\lambda_0, \dots, \lambda_{n-1})}_{= \begin{bmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_{n-1} \\ | & | & & | \end{bmatrix} \begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{n-1} \end{bmatrix}}$$

One can compute the eigenvalues $\lambda_0, \dots, \lambda_{n-1}$ as follows:

C : $C_{ij} = p_{i-j}$ vector p is defined periodically

(i.e. vector p representing the circulant matrix C)

Then: $\lambda_k = v_k^H p$.

$$\Rightarrow \quad C = U_n \operatorname{diag}(U_n^H p) U_n^{-1}$$

$$= U_n \operatorname{diag}(\overline{U_n} p) U_n^{-1}$$

U_n is symm.

The Fourier matrix is defined as:

$$\begin{array}{c}
\mathbf{F}_n = \begin{bmatrix} \omega_n^0 & \omega_n^0 & \cdots & \omega_n^0 \\ \omega_n^0 & \omega_n^1 & \cdots & \omega_n^{n-1} \\ \omega_n^0 & \omega_n^2 & \cdots & \omega_n^{2n-2} \\ \vdots & \vdots & & \vdots \\ \omega_n^0 & \omega_n^{n-1} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} = [\omega_n^{lj}]_{l,j=0}^{n-1} \in \mathbb{C}^{n,n} \\
\begin{array}{cc} \uparrow & \uparrow \\ \overline{v}_0 & \overline{v}_1 \end{array}
\end{array}$$

$$\Rightarrow \quad F_n = \overline{U_n}$$

and $\quad C = \overline{F_n} \, \mathrm{diag}\,(F_n p) \left(\overline{F_n}\right)^{-1}$

Inversion of F_n is simple:

$$\boxed{F_n^{-1} = \frac{1}{n} F_n^H = \frac{1}{n} \overline{F_n} \left(= \frac{1}{n} U_n \right)}$$

$$\Rightarrow \quad U_n^{-1} = \left(\overline{F_n}\right)^{-1} = \frac{1}{n} F_n$$

Equivalent decomposition for C:

$$C = n F_n^{-1} \, \mathrm{diag}\,(F_n p) \, \frac{1}{n} F_n$$

$$= F_n^{-1} \, \mathrm{diag}\,(F_n p) \, F_n .$$

Back to discrete convolution:

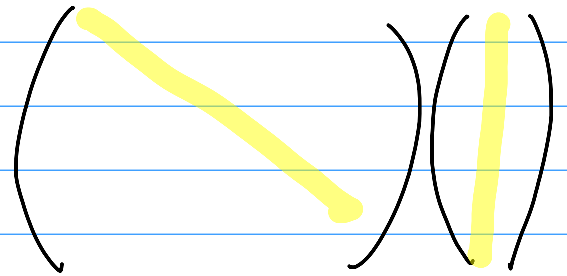
$$\underline{y} = \underline{x} * \underline{h} \quad \Leftrightarrow \quad y^L = x^L *_L h^L$$

$$\underline{y}^L = C \underline{x}^L$$

(where C is the circulant matrix constructed

by $\quad (h_0, h_1, \ldots, h_{m-1}, 0, \ldots, 0) = \underline{h}^L$

$$\underline{y}^L = C\underline{x}^L = F_L^{-1} \underbrace{\mathrm{diag}(F_L p^L)\, F_L \underline{x}^L}$$



pointwise multiplication of $F_L p^L$ and $F_L \underline{x}^L$

Theorem 4.2.1 (Convolution theorem). *The discrete periodic convolution $*_n$ between n-dimensional vectors* **u** *and* **x** *is equal to the inverse DFT of the component-wise product between the DFTs of* **u** *and* **x***; i.e.:*

$$(\mathbf{u}) *_n (\mathbf{x}) := \sum_{j=0}^{n-1} u_{k-j} x_j = \mathbf{F}_n^{-1} \left[(\mathbf{F}_n \mathbf{u})_j (\mathbf{F}_n \mathbf{x})_j \right]_{j=1}^{n} .$$

$$= \mathrm{IDFT}\left[\mathrm{DFT}(u) \underset{\uparrow}{\cdot} \mathrm{DFT}(x)\right]$$

pointwise multipl.

To convolve to signals:

1. zero-pad
2. Compute their DFTs
3. pointwise multiplication
4. take inverse DFT

In other words:

convolution in time $\Leftrightarrow$ multiplication in frequency

The Fast Fourier Transform (FFT)

DFT : $F_n y$ is $\Theta(n^2)$

Not much gain in transition from discrete convolution to DFT

Reason for exploiting the convolution theorem:

fast implementation of DFT is possible

FFT: any algorithm that performs DFT in $\mathcal{O}(n \cdot \log_2 n)$

↳ Divide-and-conquer algorithm

$$c_k := (F_n y)_k = \sum_{j=0}^{n-1} y_j \, \omega_n^{kj} = \sum_{j=0}^{n-1} y_j \, e^{-2\pi i kj/n}$$

Suppose $n = 2^\alpha$ (power of 2)

Then: $n = 2m$

Split the DFT w.r.t. even & odd indices in the sum!

$$c_k = \sum_{j=0}^{m-1} y_{2j} \, \omega_n^{2kj} + \sum_{j=0}^{m-1} y_{2j+1} \, \omega_n^{k(2j+1)}$$

$$= \sum_{j=0}^{m-1} y_{2j} \underbrace{\omega_n^{2kj}}_{=\omega_m^{kj}} + \omega_n^k \sum_{j=0}^{m-1} y_{2j+1} \underbrace{\omega_n^{2kj}}_{\omega_m^{kj}}$$

$$\left(\begin{array}{l} y^1 = [y_0, y_2, \dots, y_{n-2}]^T \\ y^2 = [y_1, y_3, \dots, y_{n-1}]^T \end{array} \right)$$

$$\omega_n^{2kj} = e^{-2\pi i (2kj)/n}$$

$$\omega_m^{kj} = e^{-2\pi i kj/m}$$

$$c_k = \underbrace{\sum_{j=0}^{m-1} (y^1)_j \, \omega_m^{kj}}_{(c^1)_k} + \omega_n^k \cdot \underbrace{\sum_{j=0}^{m-1} (y^2)_j \, \omega_m^{kj}}_{(c^2)_k}$$

y^1, y_2 signals of length $m = \frac{n}{2}$

c_1 is m-DFT of y^1 $\quad (c_1 = F_m y^1)$

c_2 is m-DFT of y^2 $\quad (c_2 = F_m y^2)$

$$(F_n y)_k = c_k = (c^1)_k + \omega_n^k (c^2)_k \qquad k = 0, \dots, m-1$$

$$c_{k+m} = \underbrace{\sum_{j=0}^{m-1} (y^1)_j \underbrace{\omega_m^{(k+m)j}}_{=\omega_m^{kj}}}_{(c^1)_k} + \underbrace{\omega_n^{m+k}}_{=-\omega_n^k} \underbrace{\sum_{j=0}^{m-1} (y^2)_j \underbrace{\omega_m^{(k+m)j}}_{\omega_m^{kj}}}_{(c^2)_k}$$

$\Rightarrow$

$$c_k = (c^1)_k + \omega_n^k (c^2)_k \qquad k = 0, \dots, m-1$$

$$c_{k+m} = (c^1)_k - \omega_n^k (c^2)_k$$

$$\omega_n^{m+k} = \underbrace{\omega_n^m}_{=-1} \cdot \omega_n^k$$

Complexity of computing c from c^1, c^2:

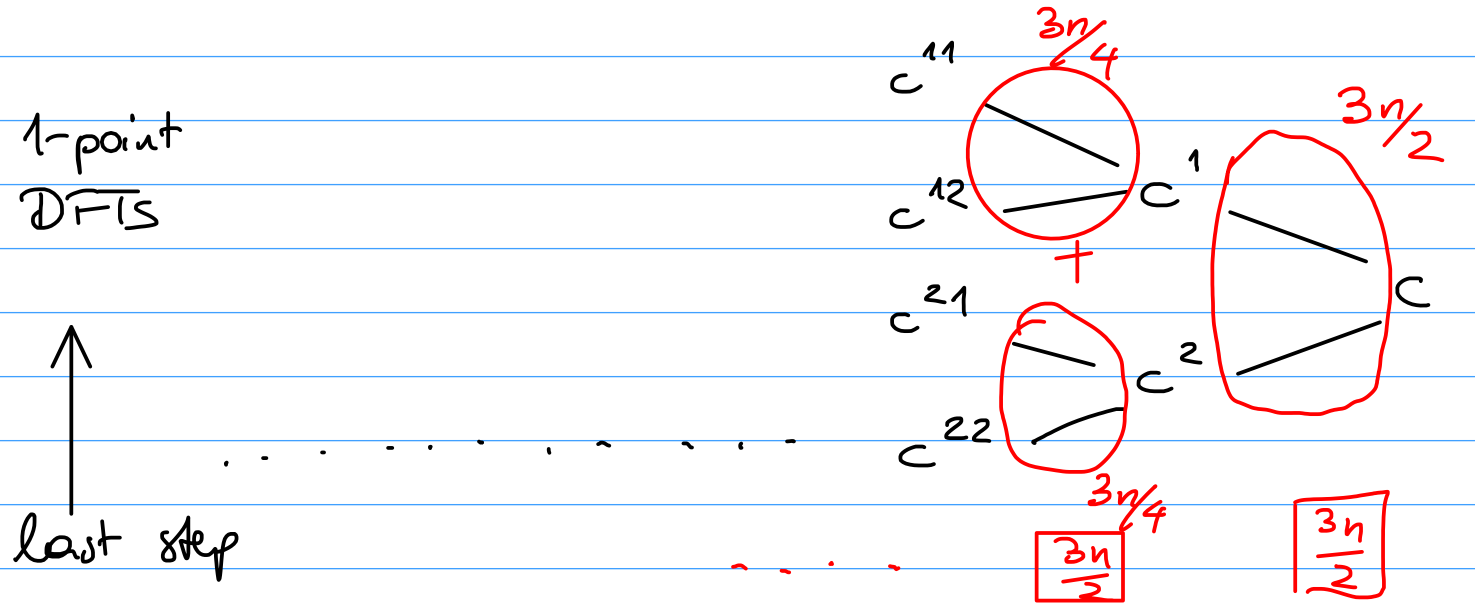
multiplications : $m = \frac{n}{2}$

additions : $2m = n$

$\Rightarrow \frac{3n}{2}$ operations

Proceed by repeatedly splitting the signals:

There are $\log_2 n$ such steps possible $(n = 2^\alpha)$



$\Rightarrow \frac{3n}{2} \cdot \log_2 n$ complexity of FFT

(Note: j-th step: $\left(\frac{3}{2} \cdot \frac{n}{2^j}\right) 2^j = \frac{3n}{2}$)

The Fourier Spectrum

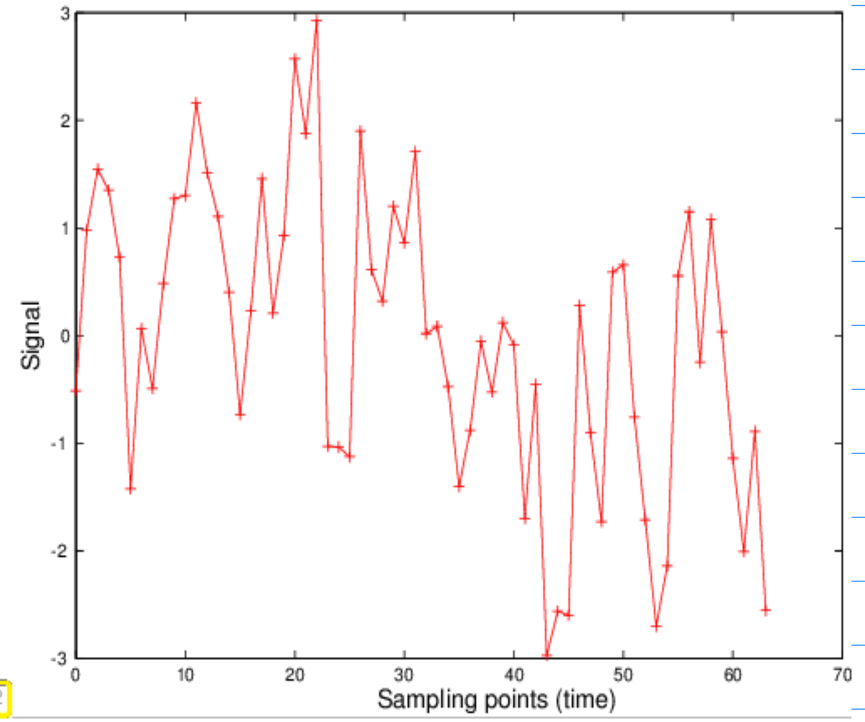


Fig. 142

Signal

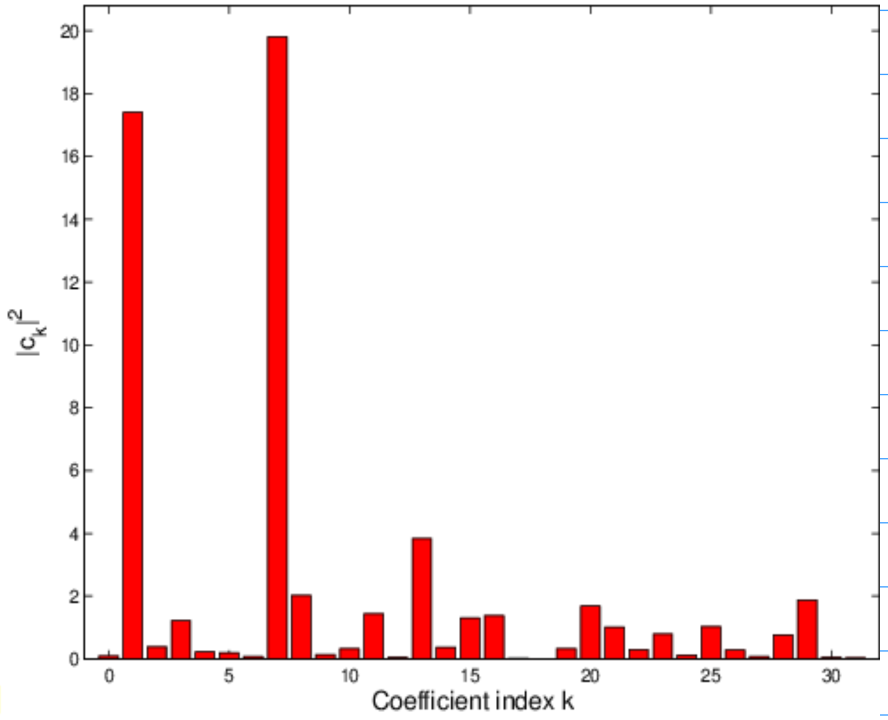


Fig. 143

Fourier magnitudes

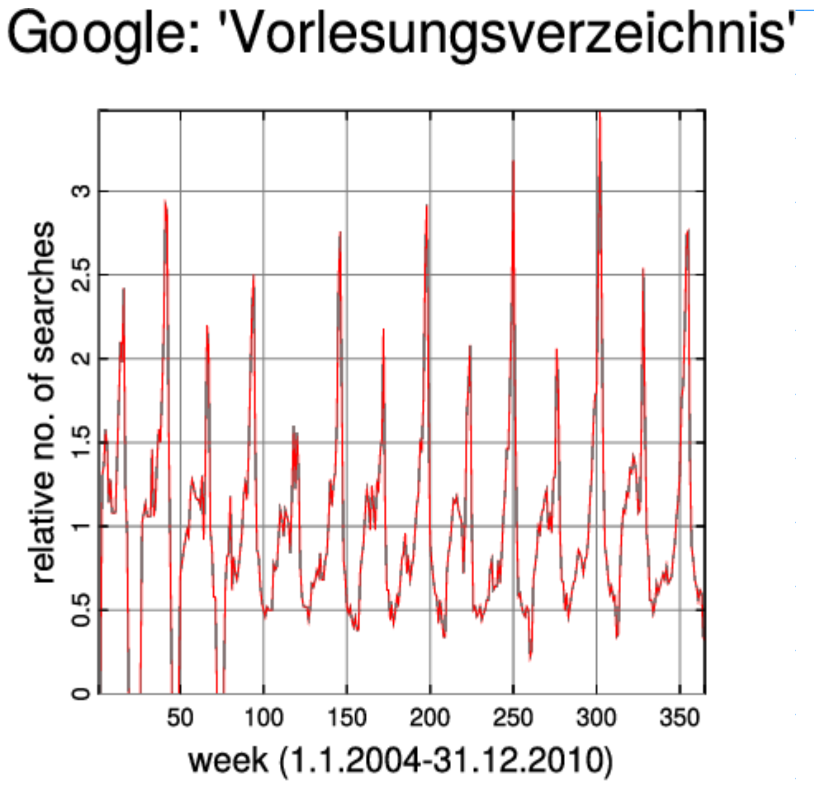


Fig. 144

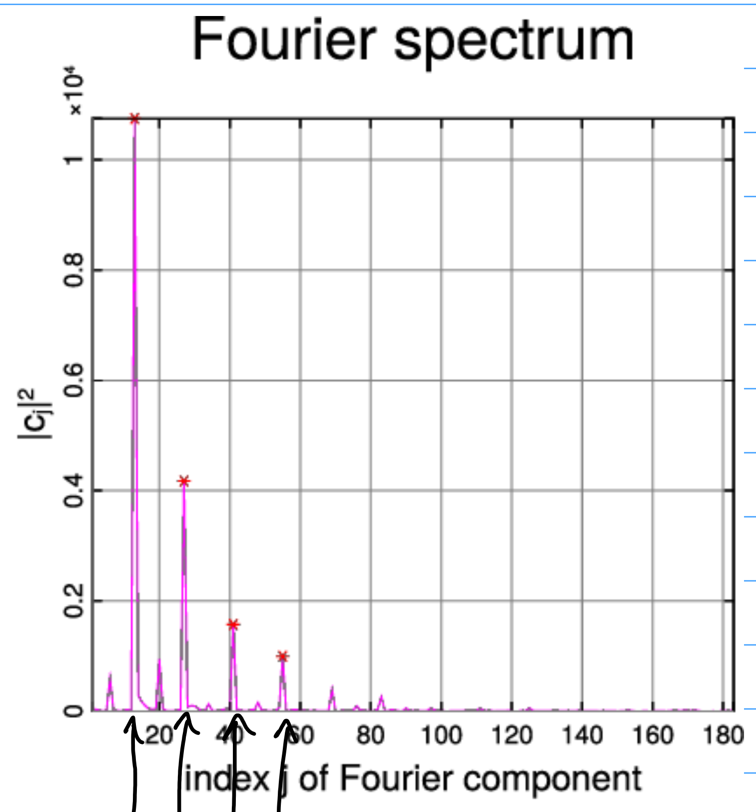


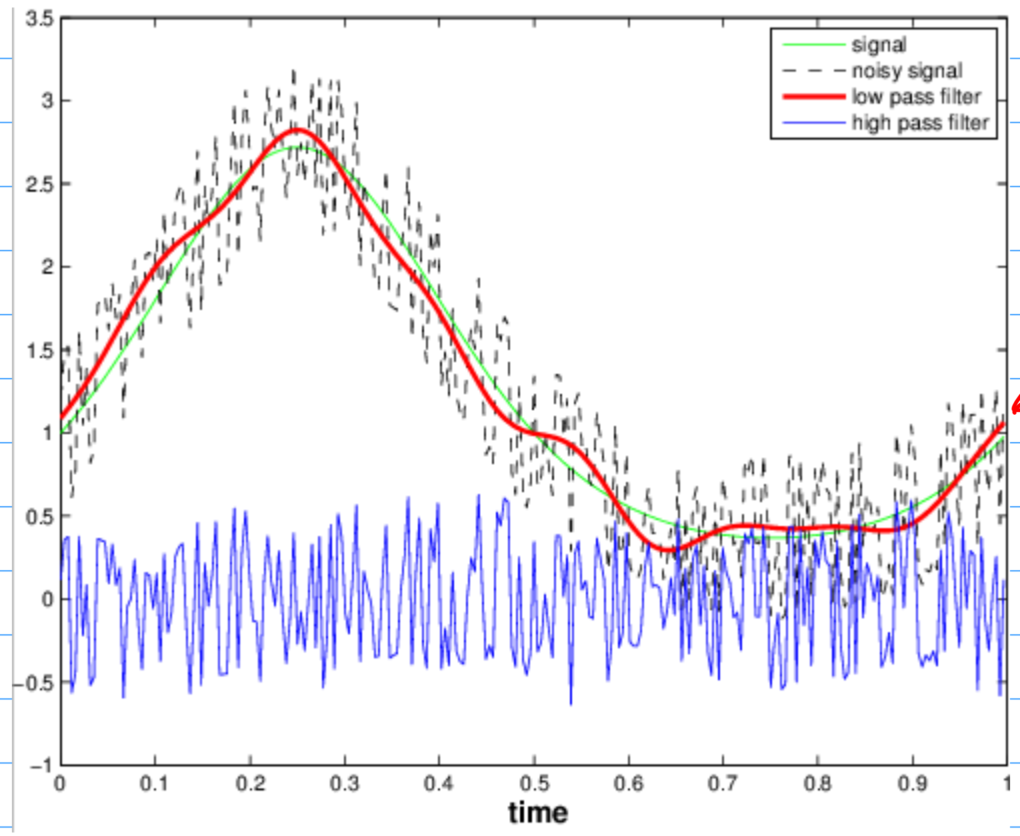
Fig. 145

position ≙ length of period

pronounced peaks

→ structure of data is periodic

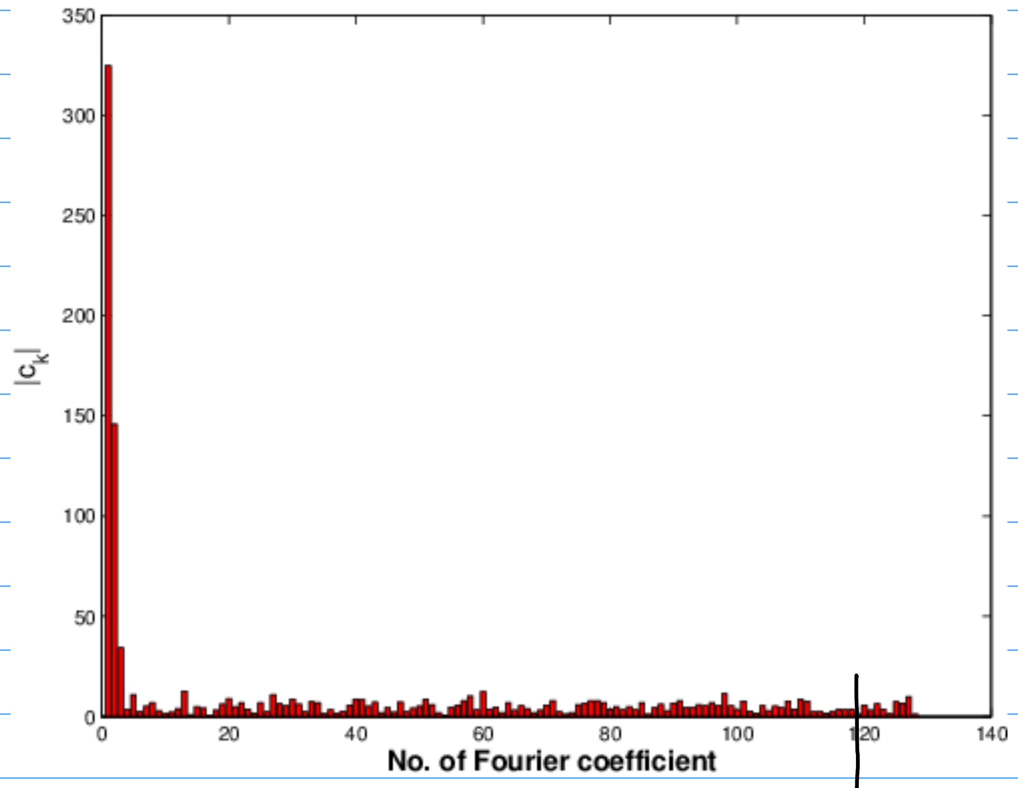
Noise in a signal: high frequent



after cut-off of high-frequencies

→ Denoising

Fourier spectrum



$k = 120$

cut-off high frequencies
to suppress noise

2D DFT

Given a matrix $Y \in \mathbb{C}^{m,n}$

Define its 2D DFT as two nested 1D DFTs

$$(\mathbf{C})_{k_1,k_2} = \sum_{j_1=0}^{m-1}\sum_{j_2=0}^{n-1} y_{j_1,j_2}\,\omega_m^{j_1k_1}\omega_n^{j_2k_2} = \underbrace{\sum_{j_1=0}^{m-1}\omega_m^{j_1k_1}\underbrace{\left(\sum_{j_2=0}^{n-1}\omega_n^{j_2k_2}y_{j_1,j_2}\right)}_{\text{1D DFT}}}_{\text{1D DFT}}\,, \quad 0 \le k_1 < m\,, 0 \le k_2 < n$$

$$(\mathbf{C})_{k_1,k_2} = \sum_{j_1=0}^{m-1}\underbrace{\left(\mathbf{F}_n(\mathbf{Y})^T_{j_1,:}\right)_{k_2}}\,\omega_m^{j_1k_1} \quad\Longrightarrow\quad \mathbf{C} = \mathbf{F}_m(\mathbf{F}_n\mathbf{Y}^T)^T = \mathbf{F}_m\mathbf{Y}\mathbf{F}_n$$

$F_n^T = F_n$

Fourier matrices from 1D DFTs

$$\begin{bmatrix} F_n Y^T_{0,:} & F_n Y^T_{1,:} & \cdots & F_n Y^T_{m-1,:} \end{bmatrix} \leftarrow k_2\text{-th row}$$

F_m is acting on the rows of $F_n Y^T$

F_n is acting on the rows of Y

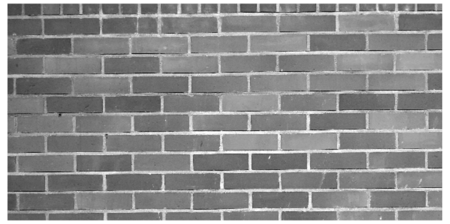
2D inverse DFT:

$$C = F_m Y F_n$$

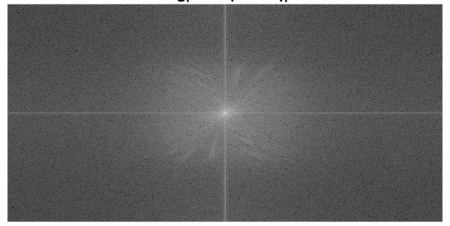
$$F_m^{-1} C F_n^{-1} = Y$$

$$\frac{1}{nm} \overline{F_m} \, C \, \overline{F_n} = Y$$

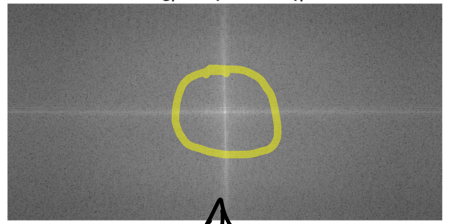
brick wall: "periodic structure"



log|FFT2(rabbit)|



log|FFT2(brick wall)|



more concentrated

Filtering with 2D DFT:

As in 1D:

1. Describe filtering as 2D convolution
2. Describe 2D linear conv. as 2D periodic convolution (through sufficient zero-padding)
3. Compute periodic convolution via FFT/DFT

Given an image:

Original Image

2D convolution theorem:

Let $U, X \in \mathbb{C}^{m,n}$: Define the 2D periodic convolution

$$\left(U *_{m,n} X\right)_{k,l} := \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (U)_{i,j} \, (X)_{\substack{k-i \bmod m \\ l-j \bmod n}}$$

Convolution theorem:

$$\mathbf{U} *_{m,n} \mathbf{X} = \overbrace{\frac{1}{nm}\overline{\mathbf{F}}_m \left[\overbrace{(\mathbf{F}_m \mathbf{U} \mathbf{F}_n)_{i,j}}^{\text{2D DFT}} \underset{\cdot}{\text{component wise}} \overbrace{(\mathbf{F}_m \mathbf{X} \mathbf{F}_n)_{i,j}}^{\text{2D DFT}} \right]_{i=0,\dots,m-1,\ j=0,\dots,n-1} \overline{\mathbf{F}}_n}^{\text{2D inverse DFT}}$$

$$\mathbf{U} *_{m,n} \mathbf{X} = \mathrm{IDFT2}\{[\mathrm{DFT2}(\mathbf{U})]_{i,j} \cdot [\mathrm{DFT2}(\mathbf{X})]_{i,j}\}_{i=0,\dots,m-1,\ j=0,\dots,n-1} \,.$$

5. Data Interpolation in 1D

Given a set of data points $(t_i, y_i) \in \mathbb{R}^2$

t_i: nodes, y_i: data values

$t_i \in I \subset \mathbb{R}$, $i \in \{0, \dots, n\}$

Goal: Find interpolant $f : I \to \mathbb{R}$

$f(t_i) = y_i$ interpolating conditions

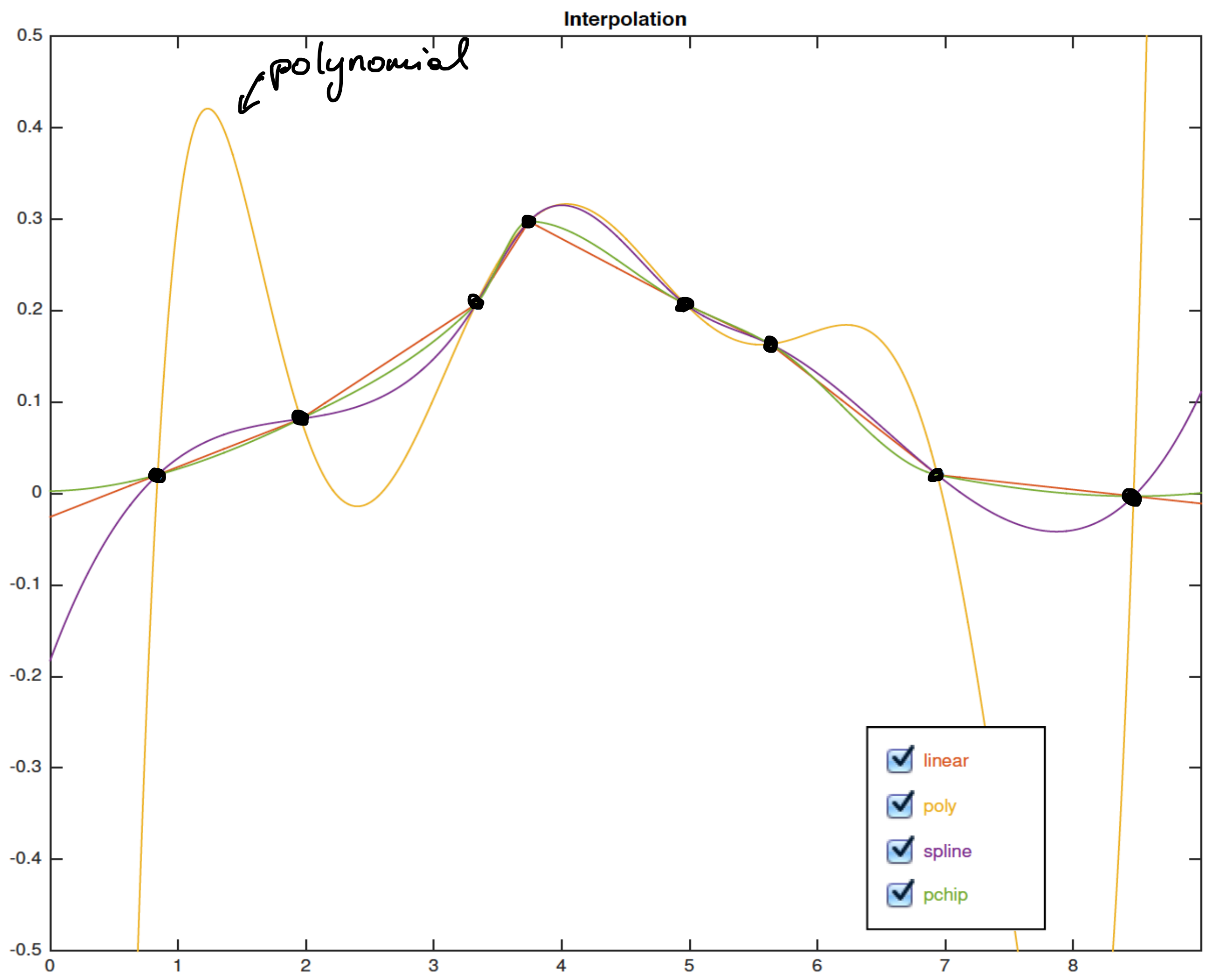
(a continuous function $f \in C^0(I)$)

interpolant: "model" for the data

can estimate the relationship / function

at intermediate points

Problem of "curve fitting"



Infinitely many functions that are candidates for an interpolant.

→ Additional assumptions on f are needed (e.g. smoothness)

Typically: search for interpolant $f \in S \subsetneq C^0(I)$

where $\dim S = m+1$

$S = \text{span}\{b_0, \dots, b_m\}$ $\quad b_j \in C^0(I)$

↑ basis for S

Interpolation would allow us:

- to predict intermediate values
- and estimate derivatives.

Note: (1) Interpolation is used when measurements are suff. accurate (otherwise: data fitting)

(2) We work with discrete quantities

"Finding a function $f: I \to \mathbb{R}$"

$\hat{=}$

"Finding a routine that given any $t \in I \cap M$ can compute $f(t)$."

$$f(t) = \sum_{j=0}^{m} c_j b_j(t)$$

↑ $\{c_j\}_{j=0}^{m}$ fully characterizes f

First method: Piecewise linear interpolation

Simplest way to connect data points continuously

Here: $S = \{ f \in \underline{C^0(I)}$ s.t.

overall cont.

on each $[t_{i-1}, t_i]$: $f(t) = \beta_i t + \gamma_i$

for $i = 1, \dots, n$; $\beta_i, \gamma_i \in \mathbb{R} \}$

For fixed points $t_0, \dots, t_n$:

$$\dim S = 2n - (n-1) = n+1$$

\# β_i, γ_i's — number of interior nodes

Or simpler: count # of data points $Y_0, \dots, Y_n$

$\Rightarrow \dim S = n+1$.

Basis for S: "hat functions"

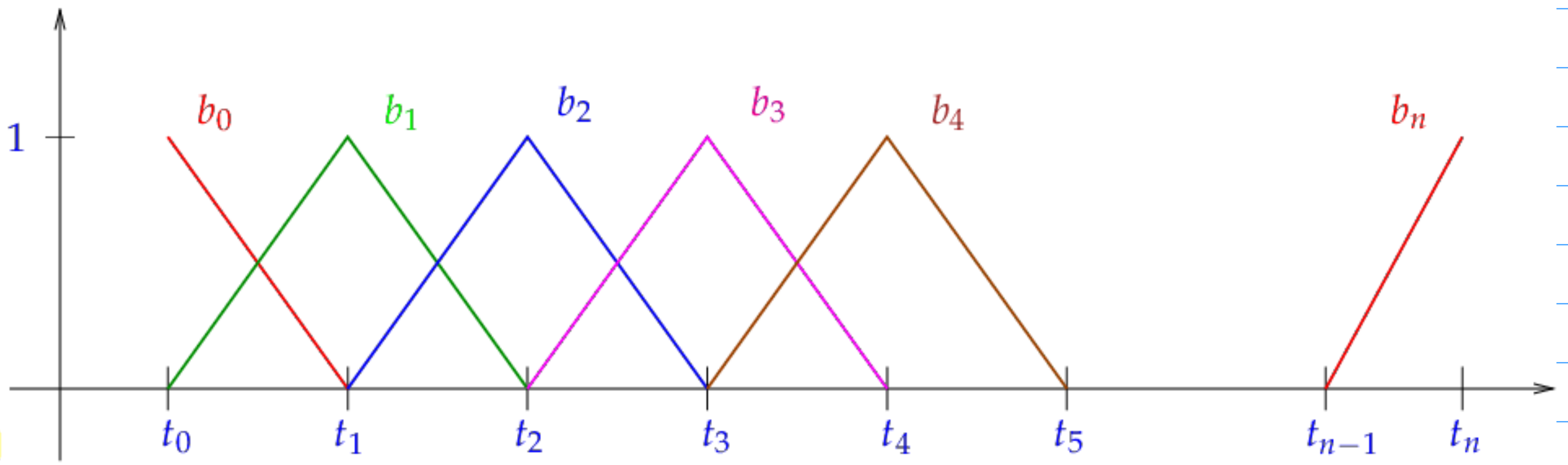


Fig. 169

Equations:

$$b_j(t) := \begin{cases} 1 - \dfrac{t_j - t}{t_j - t_{j-1}} & \text{on } [t_{j-1}, t_j] \\ 1 - \dfrac{t - t_j}{t_{j+1} - t_j} & \text{on } [t_j, t_{j+1}] \\ 0 & \text{outside of } [t_{j-1}, t_{j+1}] \end{cases}$$

$j = 1, \dots, n-1$

$$b_0(t) = \begin{cases} 1 - \frac{t - t_0}{t_1 - t_0} & \text{on } [t_0, t_1] \\ 0 & \text{on } t \geq t_1 \end{cases}$$

$$b_n(t) = \begin{cases} 1 - \frac{t_n - t}{t_n - t_{n-1}} & \text{on } [t_{n-1}, t_n] \\ 0 & \text{on } t \leq t_{n-1} \end{cases}$$

Note $b_j(t_i) = \delta_{ij}$

Find f s.t. $f(t_i) = y_i$

$$f(t) = \sum_{j=0}^{n} c_j b_j(t) \quad \text{(general interp. formula)}$$

Pw linear interp. with hat basis function

$$f(t) = \sum_{j=0}^{n} y_j b_j(t) \Rightarrow f(t_i) = y_i b_i(t_i) = y_i$$

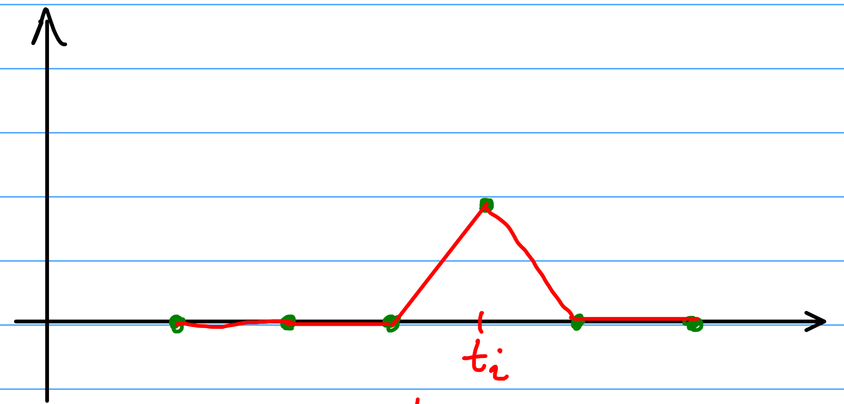
Special property of the basis $\{b_0, \dots, b_n\}$ that $b_j(t_i) = \delta_{ij}$

called : "cardinal basis"

Note : • both S and basis $\{b_j\}_{j=0}^{n}$ depend on the nodes t_i

• infinitely many choices for a basis

• for S in our example (pw. linear fcts): cardinal basis is unique



$\rightarrow$ only one way to construct b_i!

More general interpolation setting

(not necessarily pw linear interp.)

- interpolating conditions $f(t_i) = y_i \quad i = 0, \dots, n$
- for some S and basis $\{b_j\}_{j=0}^{m}$ of S

basis representation:

$$f(t) = \sum_{j=0}^{m} c_j \, b_j(t)$$

$$\Rightarrow \quad f(t_i) = \sum_{j=0}^{m} c_j \, b_j(t_i) = y_i$$

$\Rightarrow$ Amounts to solving an LSE:

$$\mathbf{A}\mathbf{c} := \begin{bmatrix} b_0(t_0) & \dots & b_m(t_0) \\ \vdots & & \vdots \\ b_0(t_n) & \dots & b_m(t_n) \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix} =: \mathbf{y}$$

Solve for coefficient vector $\underline{c}$!

$$c = A^{-1} y$$

Necessary condition for existence & uniqueness:

$m = n$

Interpolation map is linear:

$$\mathcal{I} : \; \underline{y} \mapsto f = \sum_{j=0}^{n} (A^{-1} y)_j \, b_j$$

is a linear map

Property that A is invertible will depend on
- nodes t_i
- space S

but not on the specific choice of the basis $\{b_0, \dots, b_n\}$

[Exercise]

$$A = \begin{bmatrix} b_0(t_0) & \cdots & b_n(t_0) \\ b_0(t_1) & \cdots & b_n(t_1) \\ \vdots & & \\ b_0(t_n) & \cdots & b_n(t_n) \end{bmatrix}$$

cardinal basis $b_i(t_j) = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{if } i \neq j \end{cases}$

For this: $A = \begin{bmatrix} 1 & & & \\ & 1 & & 0 \\ & & 1 & \\ 0 & & & \ddots \\ & & & & 1 \end{bmatrix} = I_{n+1}$

Global Polynomial Interpolation