ETH Lecture 401-0663-00L Numerical Methods for CSE

Numerical Methods for Computational Science and Engineering

Prof. R. Hiptmair, SAM, ETH Zurich

(with contributions from Prof. P. Arbenz and Dr. V. Gradinaru)

Autumn Term 2016

URL: http://www.sam.math.ethz.ch/~hiptmair/tmp/NumCSE/NumCSE16.pdf

IV. Filtering Algorithms

4.1. Discrete Convolutions

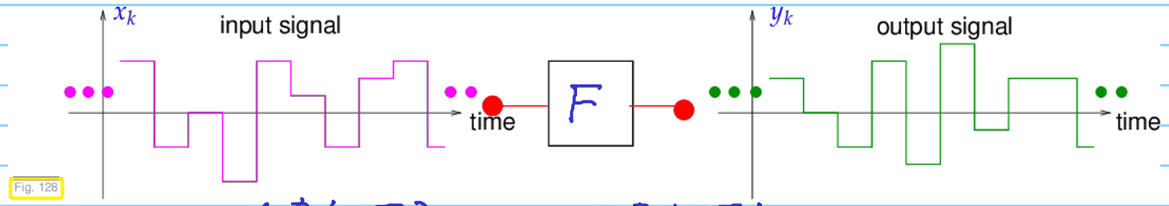


Fig. 128

$F : \ell^\infty(\mathbb{Z}) \to \ell^\infty(\mathbb{Z})$

Impulse response $(h_j) = F((\delta_{0j})_{j\in\mathbb{Z}})$

Definition 4.1.4. Finite channel/filter

A filter $F : \ell^\infty(\mathbb{Z}) \to \ell^\infty(\mathbb{Z})$ is called finite, if every input signal of finite duration produces an output signal of finite duration,

$$\{\, \exists M \in \mathbb{N}: \ |j| > M \ \Rightarrow\ x_j = 0 \,\} \ \Rightarrow\ \exists N \in \mathbb{N}: \ |k| > N \ \Rightarrow\ (F((x_j)_{j\in\mathbb{Z}}))_k = 0\,, \tag{4.1.5}$$

Definition 4.1.11. Causal channel/filter

A filter $F : \ell^\infty(\mathbb{Z}) \to \ell^\infty(\mathbb{Z})$ is called causal (or physical, or nonanticipative), if the output does not start before the input

$$\forall M \in \mathbb{N}: \ (x_j)_{j\in\mathbb{Z}} \in \ell^\infty(\mathbb{Z}),\ x_j = 0 \ \ \forall j \le M \ \Rightarrow\ F((x_j)_{j\in\mathbb{Z}})_k = 0 \ \ \forall k \le M\,. \tag{4.1.12}$$

Impulse response does not depend on t.

$\ell^\infty(\mathbb{Z}) \triangleq$ vector space of bounded signals

"Signals in pictures"

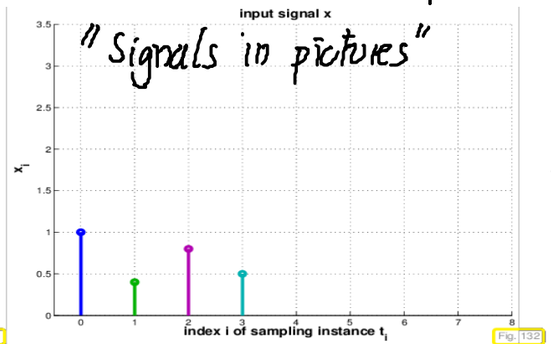


Fig. 131

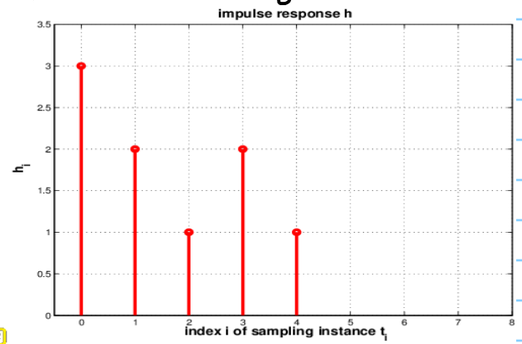


Fig. 132

(2)

Definition 4.1.7. Time-invariant channel/filter

A filter $F : \ell^\infty(\mathbb{Z}) \to \ell^\infty(\mathbb{Z})$ is called time-invariant, if shifting the input in time leads to the same output shifted in time by the same amount; it *commutes with the time shift operator* from (4.1.6):

$$\forall (x_j)_{j\in\mathbb{Z}} \in \ell^\infty(\mathbb{Z}),\ \forall m \in \mathbb{Z}: \quad F(S_m((x_j)_{j\in\mathbb{Z}})) = S_m(F((x_j)_{j\in\mathbb{Z}}))\,. \tag{4.1.8}$$

time shift $S_m((x_j)_{j\in\mathbb{Z}}) = (x_{j-m})_{j\in\mathbb{Z}}$

$\triangleright$ $F((\delta_{jm})_{j\in\mathbb{Z}}) = (h_{j-m})_{j\in\mathbb{Z}}$

Definition 4.1.9. Linear channel/filter

A filter $F : \ell^\infty(\mathbb{Z}) \to \ell^\infty(\mathbb{Z})$ is called linear, if F is a linear mapping:

$$F(\alpha(x_j)_{j\in\mathbb{Z}} + \beta(y_j)_{j\in\mathbb{Z}}) = \alpha F((x_j)_{j\in\mathbb{Z}}) + \beta F((y_j)_{j\in\mathbb{Z}}) \quad \forall (x_j)_{j\in\mathbb{Z}}, (y_j)_{j\in\mathbb{Z}} \in \ell^\infty(\mathbb{Z}),\ \alpha,\beta \in \mathbb{R}\,. \tag{4.1.10}$$

LT - FIR channel

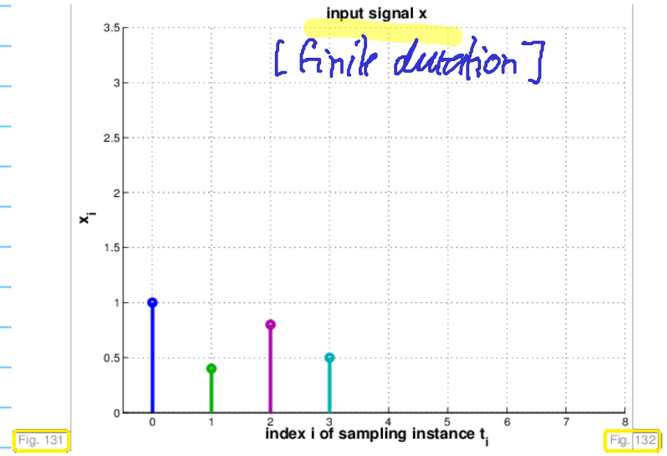
LT-FIR formula, i.e. $(0,\dots,h_0,\dots,h_{n-1},0,\dots)$, $n \in \mathbb{N}$

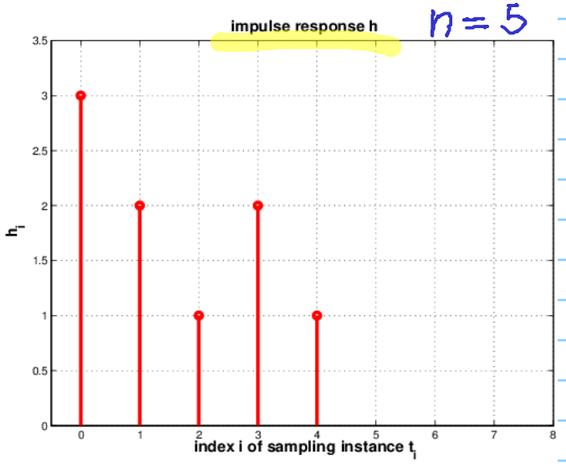
$$F((x_j)) = F\Big(\sum_{k\in\mathbb{Z}} x_k (\delta_{kj})_{j\in\mathbb{Z}}\Big)$$

$$= \sum_{k\in\mathbb{Z}} x_k F((\delta_{kj})_{j\in\mathbb{Z}})$$

$$= \sum_{k\in\mathbb{Z}} x_k (h_{j-k})_{j\in\mathbb{Z}}$$

$$(F((x_j)_{j\in\mathbb{Z}}))_j = \sum_{k\in\mathbb{Z}} x_k h_{j-k} \quad \text{[finite sum]}$$





$(0,\dots 0, x_0 \dots, x_{m-1}, 0 \dots)$ [finite]

$\Rightarrow$ Duration of output: $(m+n-2)\Delta t$!

③

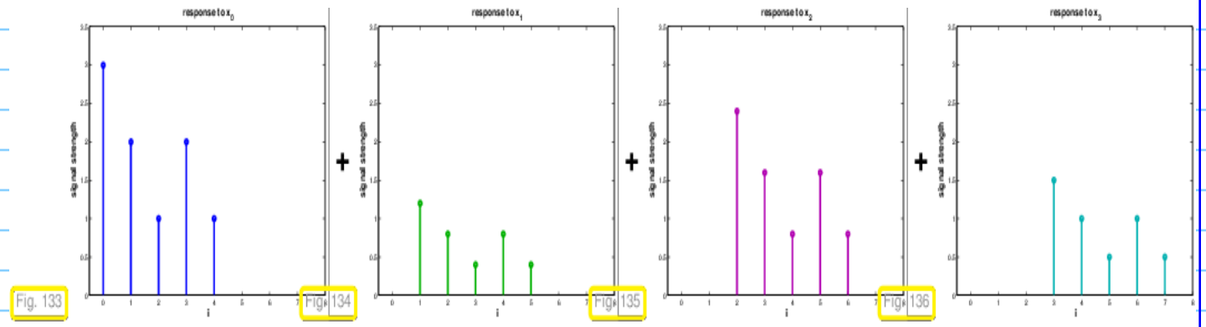


Fig. 133 Fig. 134 Fig. 135 Fig. 136

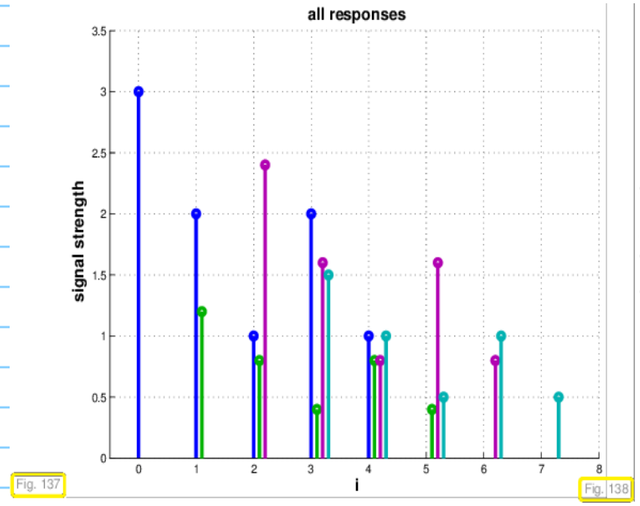


Fig. 137

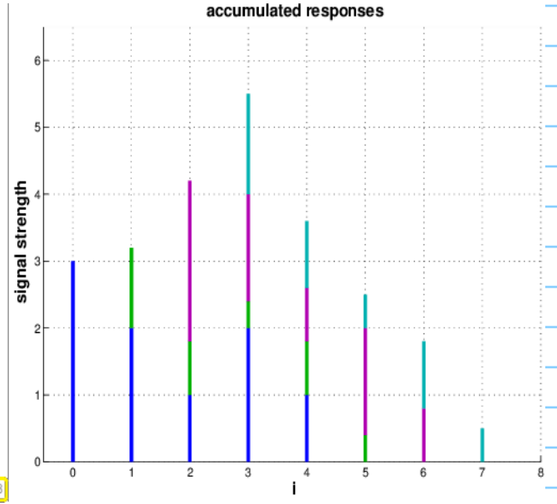


Fig. 138

For finite signals $(\ldots, 0, \ldots 0, x_0, \ldots, x_{n-1}, 0, \ldots) \sim \underline{x} \in \mathbb{R}^n$

LT-FIR channel ~ linear mapping $\mathbb{R}^n \to \mathbb{R}^{2n-1}$

↳ has a matrix representation

$$\begin{bmatrix} y_0 \\ \vdots \\ y_{2n-2} \end{bmatrix} = \begin{bmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & & & 0 \\ & \ddots & & \\ h_{n-1} & & & h_0 \\ 0 & & & h_1 \\ & \ddots & & \\ 0 & & 0 & h_{n-1} \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ x_{n-1} \end{bmatrix} . \tag{4.1.18}$$

$$\Updownarrow$$

$$(F((x_j)_{j\in\mathbb{Z}}))_j = \sum_{k\in\mathbb{Z}} x_k h_{j-k} =: y_j$$

Definition 4.1.22. Discrete convolution

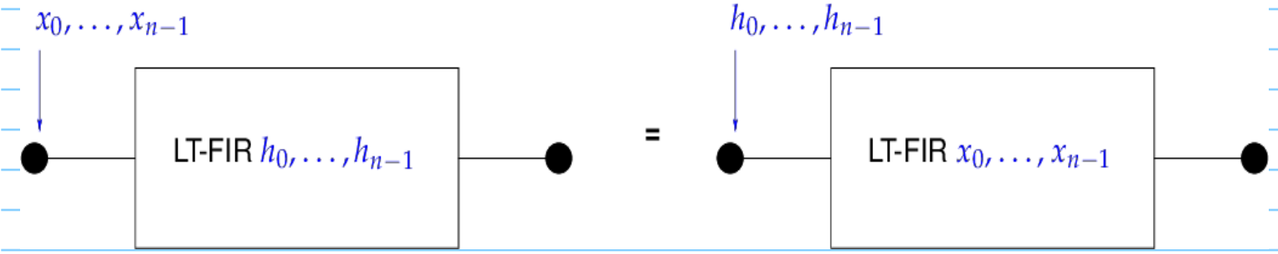
Given $\mathbf{x} = [x_0, \ldots, x_{n-1}]^\top \in \mathbb{K}^n$, $\mathbf{h} = [h_0, \ldots, h_{n-1}]^\top \in \mathbb{K}^n$ their discrete convolution is the vector $\mathbf{y} \in \mathbb{K}^{2n-1}$ with components

$$y_k = \sum_{j=0}^{n-1} h_{k-j} x_j , \quad k = 0, \ldots, 2n-2 \quad (h_j := 0 \text{ for } j < 0) . \tag{4.1.23}$$

Notation: $\underline{y} = \underline{h} * \underline{x} = \underline{x} * \underline{h}$

(4)

$$\sum_{k\in\mathbb{Z}} x_k h_{j-k} \underset{\ell:=j-k}{=} \sum_{\ell\in\mathbb{Z}} x_{j-\ell} h_\ell$$



$\hat{=}$ $\underline{x} \in \mathbb{R}^n : \underline{x} = [x_0, \ldots, x_{n-1}]^T$

Now: n-periodic input $(x_j)_{j\in\mathbb{Z}}$: $x_j = x_{j+n}$ $\forall j$

$\Rightarrow$ n-periodic output of LT-FIR $\hat{=}$ $\underline{y} \in \mathbb{R}^n$

LT-FIR in n-periodic setting : linear mapping $\mathbb{R}^n \to \mathbb{R}^n$

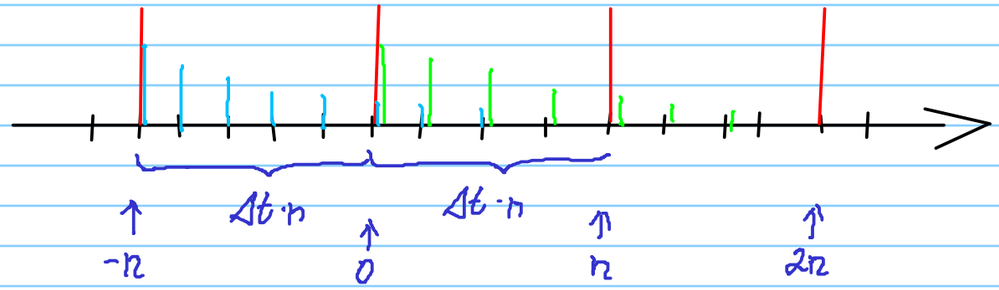
$$y_k = \sum_{\ell=0}^{n-1} p_{k-\ell} x_\ell \qquad [\underline{y} = \underline{p} *_n \underline{x}]$$

p : n-periodic impulse response

If $(h_j)_{j\in\mathbb{Z}} \hat{=}$ I.R. of LT-FIR, then

$$p_j = \sum_{k\in\mathbb{N}_0} h_{j+kn}, \quad j = 0, \ldots, n-1$$

Hint: $(p_j) \hat{=}$ output for



Definition 4.1.29. Discrete periodic convolution

The discrete periodic convolution of two n-periodic sequences $(p_k)_{k\in\mathbb{Z}}$, $(x_k)_{k\in\mathbb{Z}}$ yields the n-periodic sequence

$$(y_k) := (p_k) *_n (x_k) \quad , \quad y_k := \sum_{j=0}^{n-1} p_{k-j} x_j = \sum_{j=0}^{n-1} x_{k-j} p_j \, , \quad k \in \mathbb{Z} \, .$$

↑ defined by n-periodicity

$$\begin{bmatrix} y_0 \\ \vdots \\ y_{n-1} \end{bmatrix} = \underbrace{\begin{bmatrix} p_0 & p_{n-1} & p_{n-2} & \cdots & \cdots & p_1 \\ p_1 & p_0 & p_{n-1} & & & \vdots \\ p_2 & p_1 & p_0 & \ddots & & \\ \vdots & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & p_{n-1} \\ p_{n-1} & \cdots & & & p_1 & p_0 \end{bmatrix}}_{=:\mathbf{P}} \begin{bmatrix} x_0 \\ \vdots \\ x_{n-1} \end{bmatrix} . \qquad (4.1.27)$$

disc. per. conv = multiplication with circulant matrix

(5)

Definition 4.1.34. Circulant matrix → [?, Sect. 54]

A matrix $\mathbf{C} = [c_{ij}]_{i,j=1}^{n} \in \mathbb{K}^{n,n}$ is circulant $\quad [\ C = \mathrm{circul}(p)\ ]$

$:\Leftrightarrow \quad \exists (p_k)_{k\in\mathbb{Z}}$ n-periodic sequence: $c_{ij} = p_{j-i}, 1 \le i,j \le n.$

4.2. Discrete Fourier Transform (DFT)

Experiment : EV of random circulant matrices

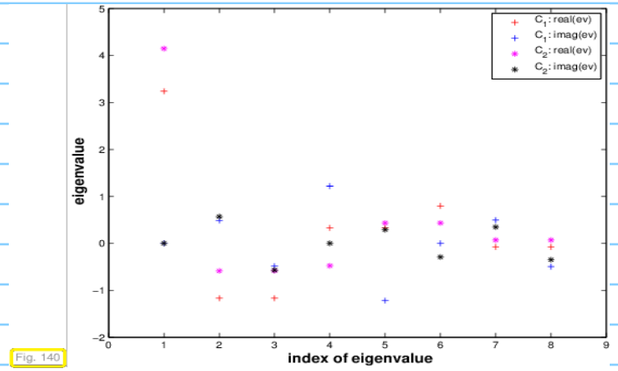


Fig. 140

[8×8 example]

↳ "random spectrum"

Reduction : discrete convolution ⇒ periodic d.c.

→ $\underline{z} = \underline{a} * \underline{b} = \underline{x} *_{2n-1} \underline{y}$

$\underline{a}, \underline{b} \in \mathbb{R}^n$

duration $2n-2$

Zero padding : $\underline{x} := \begin{bmatrix} \underline{a} \\ 0 \end{bmatrix} \in \mathbb{R}^{2n-1}$

$\underline{y} := \begin{bmatrix} \underline{b} \\ 0 \end{bmatrix} \in \mathbb{R}^{2n-1}$

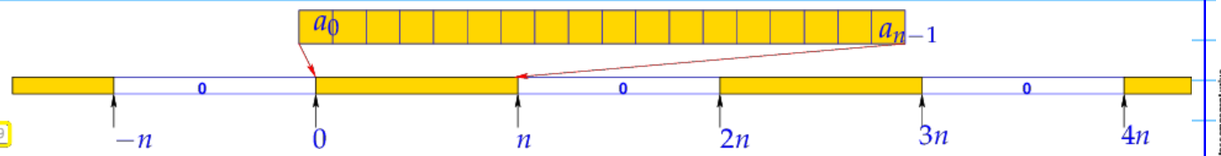
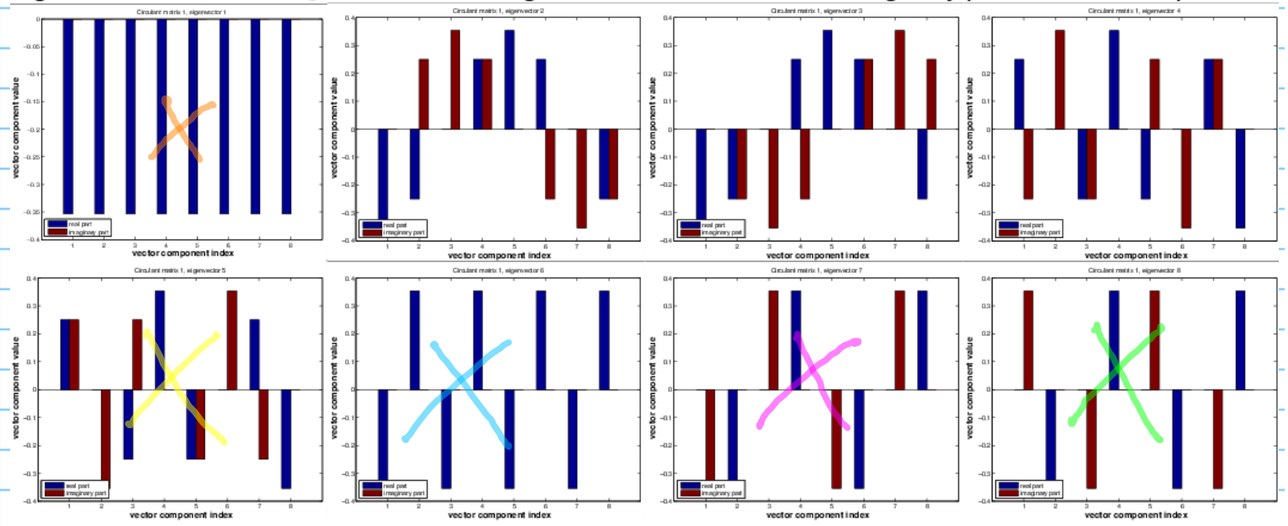
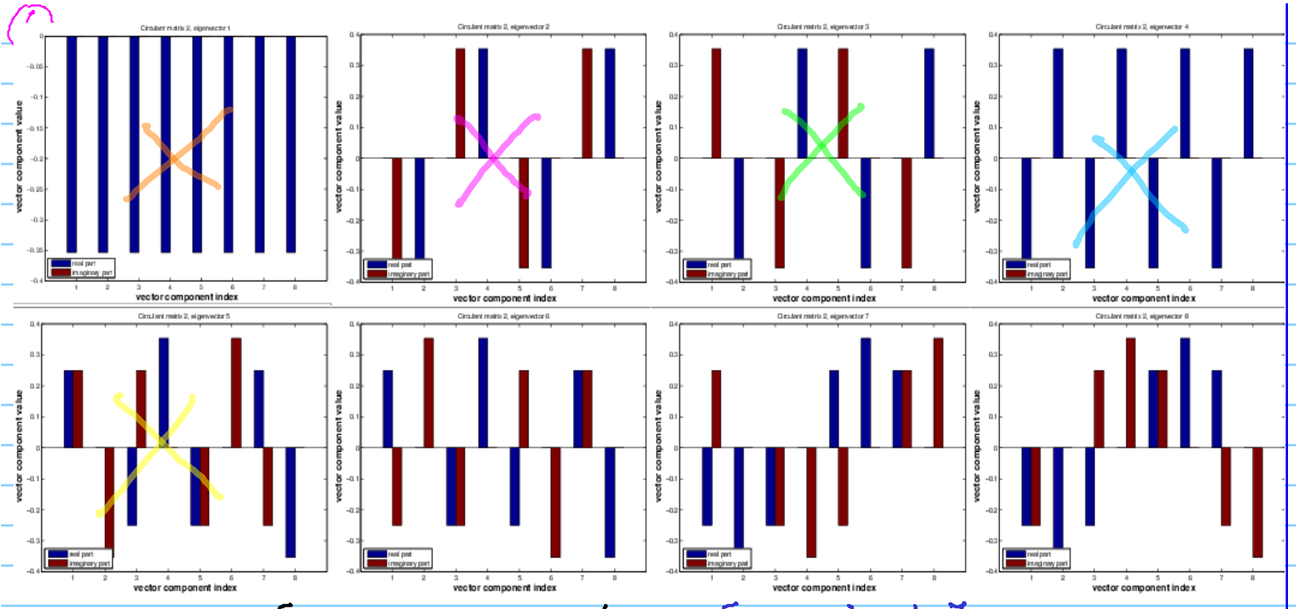


Fig. 139

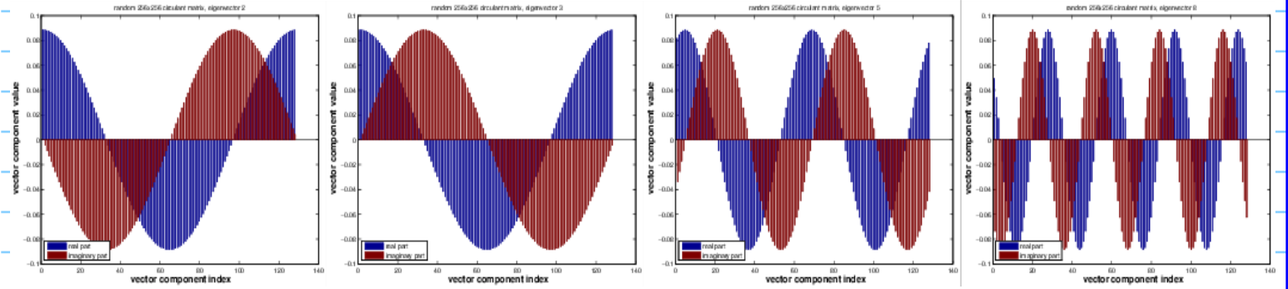
Eigenvectors of matrix $\mathbf{C}_1$, visualized through the size of the real and imaginary parts of their components.





▷ Same eigenvectors [complex!]

Eigenvectors of C = gallery('circul',(1:128)); :



$$\omega_n := e^{-\frac{2\pi i}{n}} = \cos\left(\frac{2\pi}{n}\right) - i\sin\left(\frac{2\pi}{n}\right) \quad \text{[root unity]}$$

$$\omega_n^n = 1 \qquad (*)$$

$$\omega_n^{-\ell} = \overline{\omega}_n^{\ell} \quad \text{[complex conjugation]}$$

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

$$\underline{v}_k = [\omega_n^{jk}]_{j=0}^{n-1} \in \mathbb{C}^n, \quad 0 \le k < n$$

$$C := \mathrm{circul}(\underline{u})$$

$$(C\underline{v}_k)_j = \sum_{\ell=0}^{n-1} u_\ell (\underline{v}_k)_{j-\ell} = (\underline{u} *_n \underline{v}_k)_j$$

$$= \sum_{\ell=0}^{n-1} u_\ell \, \omega_n^{k(j-\ell)}$$

$$= \omega_n^{kj} \underbrace{\sum_{\ell=0}^{n-1} u_\ell \, \omega_n^{-k\ell}}_{\text{eigenvalue } \lambda_k}$$

$\uparrow$ $(\underline{v}_k)_j$

▷ $C\underline{v}_k = \lambda_k \underline{v}_k$

Basis of eigenvectors for any circulant matrix

$$\{\mathbf{v}_0, \ldots, \mathbf{v}_{n-1}\} = \left\{ \begin{bmatrix} \omega_n^0 \\ \vdots \\ \omega_n^0 \end{bmatrix} \begin{bmatrix} \omega_n^0 \\ \omega_n^1 \\ \vdots \\ \omega_n^{n-1} \end{bmatrix} \cdots \begin{bmatrix} \omega_n^0 \\ \omega_n^{n-2} \\ \omega_n^{2(n-2)} \\ \vdots \\ \omega_n^{(n-1)(n-2)} \end{bmatrix} \begin{bmatrix} \omega_n^0 \\ \omega_n^{n-1} \\ \omega_n^{2(n-1)} \\ \vdots \\ \omega_n^{(n-1)^2} \end{bmatrix} \right\}. \qquad (4.2.11)$$

= trigonometric basis of $\mathbb{C}^n$

(7)

$$\mathbf{F}_n = \begin{bmatrix} \omega_n^0 & \omega_n^0 & \cdots & \omega_n^0 \\ \omega_n^0 & \omega_n^1 & \cdots & \omega_n^{n-1} \\ \omega_n^0 & \omega_n^2 & \cdots & \omega_n^{2n-2} \\ \vdots & \vdots & & \vdots \\ \omega_n^0 & \omega_n^{n-1} & \cdots & \omega_n^{(n-1)^2} \end{bmatrix} = \left[\omega_n^{lj}\right]_{l,j=0}^{n-1} \in \mathbb{C}^{n,n} . \qquad (4.2.13)$$

Fourier matrix

$$C F_n = F_n D \quad , D \hat{=} \text{diagonal}$$

$$(D)_{k,k} = \sum_{\ell=0}^{n-1} u_\ell \underbrace{\omega_n^{-k\ell}}_{(\overline{F}_n)_{k\ell}} \Rightarrow [(D)_{k,k}]_k = \overline{F}_n u$$

▷ Orthogonal eigenvector

$$F_n^H F_n = n \cdot I_n$$

Lemma 4.2.14. Properties of Fourier matrix

The scaled Fourier-matrix $\frac{1}{\sqrt{n}}\mathbf{F}_n$ is unitary (→ Def. 6.2.2) : $\mathbf{F}_n^{-1} = \frac{1}{n}\mathbf{F}_n^H = \frac{1}{n}\overline{\mathbf{F}}_n$.

Lemma 4.2.16. Diagonalization of circulant matrices (→ Def. 4.1.34)

For any circulant matrix $\mathbf{C} \in \mathbb{K}^{n,n}$, $c_{ij} = u_{i-j}$, $(u_k)_{k\in\mathbb{Z}}$ n-periodic sequence, holds true

$$\mathbf{C}\overline{\mathbf{F}}_n = \overline{\mathbf{F}}_n \operatorname{diag}(d_1,\ldots,d_n) \quad , \quad \mathbf{d} = \mathbf{F}_n[u_0,\ldots,u_{n-1}]^\top .$$

$$C = \frac{1}{n}\overline{F}_n \operatorname{diag}(d_1,..,d_n) F_n$$

Cx can be realized by multiplication with Fourier matrix

↳ periodic disc. conv.

The Fourier matrix :

$$V_k^H V_m = \sum_{\ell=0}^{n-1} \omega_n^{-k\ell} \omega_n^{m\ell} = \sum_{\ell=0}^{n-1} \omega^{\ell(m-k)} .$$

$V_k = [\omega_n^{jk}]_{j=0}^{n-1}$, $0 \le k, m < n$

$$= \sum_{\ell=0}^{n-1} \left[e^{\frac{2\pi i}{n}(m-k)}\right]^\ell = \begin{cases} n & m = k \\ \frac{1-\left[e^{\frac{2\pi i}{n}(m-k)}\right]^n}{1-e^{\frac{2\pi i}{n}(m-k)}} = 0 & m \neq k \end{cases} \quad (*)$$

$$\begin{bmatrix} \omega_n^0 & \omega_n^k & \cdots & \omega_n^{(n-1)k} \end{bmatrix} \cdot \begin{bmatrix} \omega_n^0 \\ \omega_n^m \\ \vdots \\ \omega_n^{(n-1)m} \end{bmatrix} = V_k^H \cdot V_m$$

Definition 4.2.18. Discrete Fourier transform (DFT)

The linear map $\mathcal{F}_n : \mathbb{C}^n \mapsto \mathbb{C}^n$, $\mathcal{F}_n(\mathbf{y}) := \mathbf{F}_n\mathbf{y}$, $\mathbf{y} \in \mathbb{C}^n$, is called discrete Fourier transform (DFT), i.e. for $\mathbf{c} := \mathcal{F}_n(\mathbf{y}) = \mathbf{F}_n\mathbf{y}$

$$c_k = \sum_{j=0}^{n-1} y_j \omega_n^{kj} \quad , \quad k = 0, \ldots, n-1 \, . \tag{4.2.19}$$

[Lemma 4.2.14]

$$\underline{c} = \mathbf{F}_n \underline{y} \iff \underline{y} = \frac{1}{n}\overline{\mathbf{F}}_n \underline{c}$$

Inverse DFT : $y_k = \frac{1}{n}\sum_{j=0}^{n-1} c_j \omega_n^{-kj}$

C++11 code 4.2.22: Demo: discrete Fourier transform in EIGEN ⇢ GITLAB

```
int main() {
  using Comp = complex<double>;
  const VectorXcd::Index n = 5;
  VectorXcd y(n),c(n),x(n);
  y << Comp(1,0),Comp(2,1),Comp(3,2),Comp(4,3),Comp(5,4);
  FFT<double> fft; // DFT transform object
  c = fft.fwd(y);  // DTF of y, see Def. 4.2.18
  x = fft.inv(c);  // inverse DFT of c, see (4.2.20)

  cout << "y = " << y.transpose() << endl
       << "c = " << c.transpose() << endl
       << "x = " << x.transpose() << endl;
  return 0;
}
```

4.2.1. Discrete Convolutions via DFT

- periodic conv. $\iff$ multiplication w/ circulant matrix

Conclusion (from $\overline{\mathbf{F}}_n = n\mathbf{F}_n^{-1}$): $\mathbf{C} = \mathbf{F}_n^{-1}\operatorname{diag}(d_1, \ldots, d_n)\mathbf{F}_n$. (4.2.17)

$C = \operatorname{circ}(u)$, $d := \mathbf{F}_n u$

C++11 code 4.2.25: Discrete periodic convolution: DFT implementation ⇢ GITLAB

```
VectorXcd pconvfft(const VectorXcd& u, const VectorXcd& x) {
  Eigen::FFT<double> fft;
  VectorXcd tmp = ( fft.fwd(u) ).cwiseProduct( fft.fwd(x) );
  return fft.inv(tmp);
}
```

- general disc. conv. : by zero padding

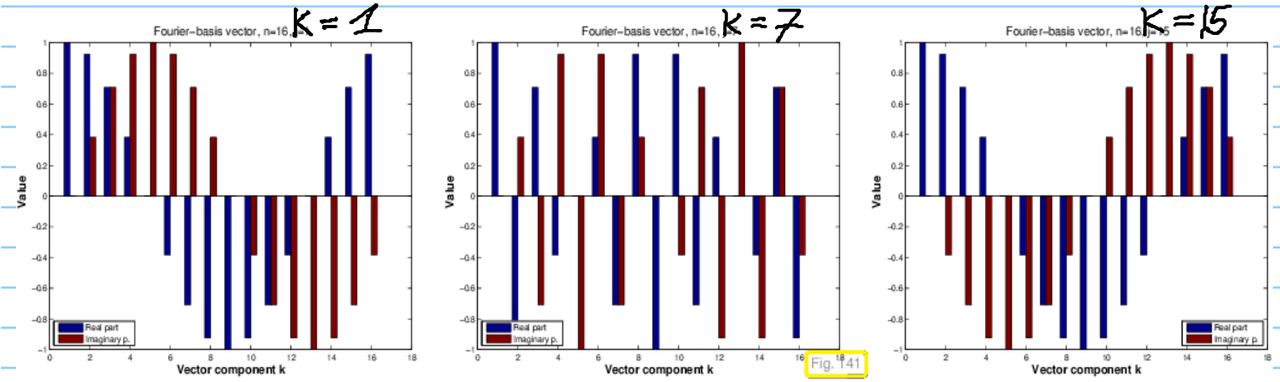
C++11 code 4.2.26: Implementation of discrete convolution (→ Def. 4.1.22) based on periodic discrete convolution ⇢ GITLAB

```
VectorXcd myconv(const VectorXcd& h, const VectorXcd& x) {
  const long n = h.size();
  // Zero padding, cf. (4.1.37)
  VectorXcd hp(2*n - 1), xp(2*n - 1);
  hp << h, VectorXcd::Zero(n - 1);
  xp << x, VectorXcd::Zero(n - 1);
  // Periodic discrete convolution of length 2n-1, ??
  return pconvfft(hp, xp);
}
```

4.2.2. Frequency Filtering via DFT

Trigonometric basis vector $\longleftrightarrow$ oscillation

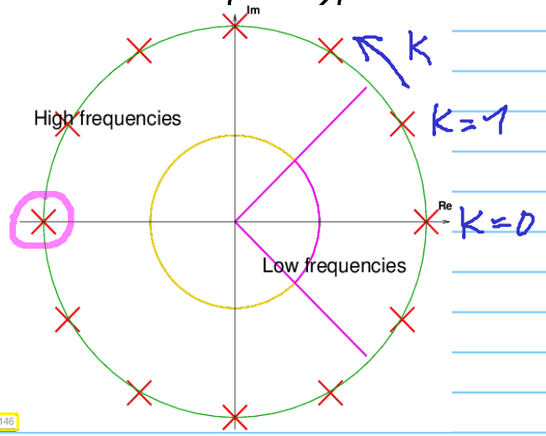
$\hookrightarrow$ $y_k = \left[\omega_n^{kj}\right]_{j=0}^{n-1} = \left[\cos\left(2\pi \frac{kj}{n}\right)\right]_{j=0}^{n-1} + i\left[\sin\left(\frac{2\pi}{n} kj\right)\right]_{j=0}^{n-1}$



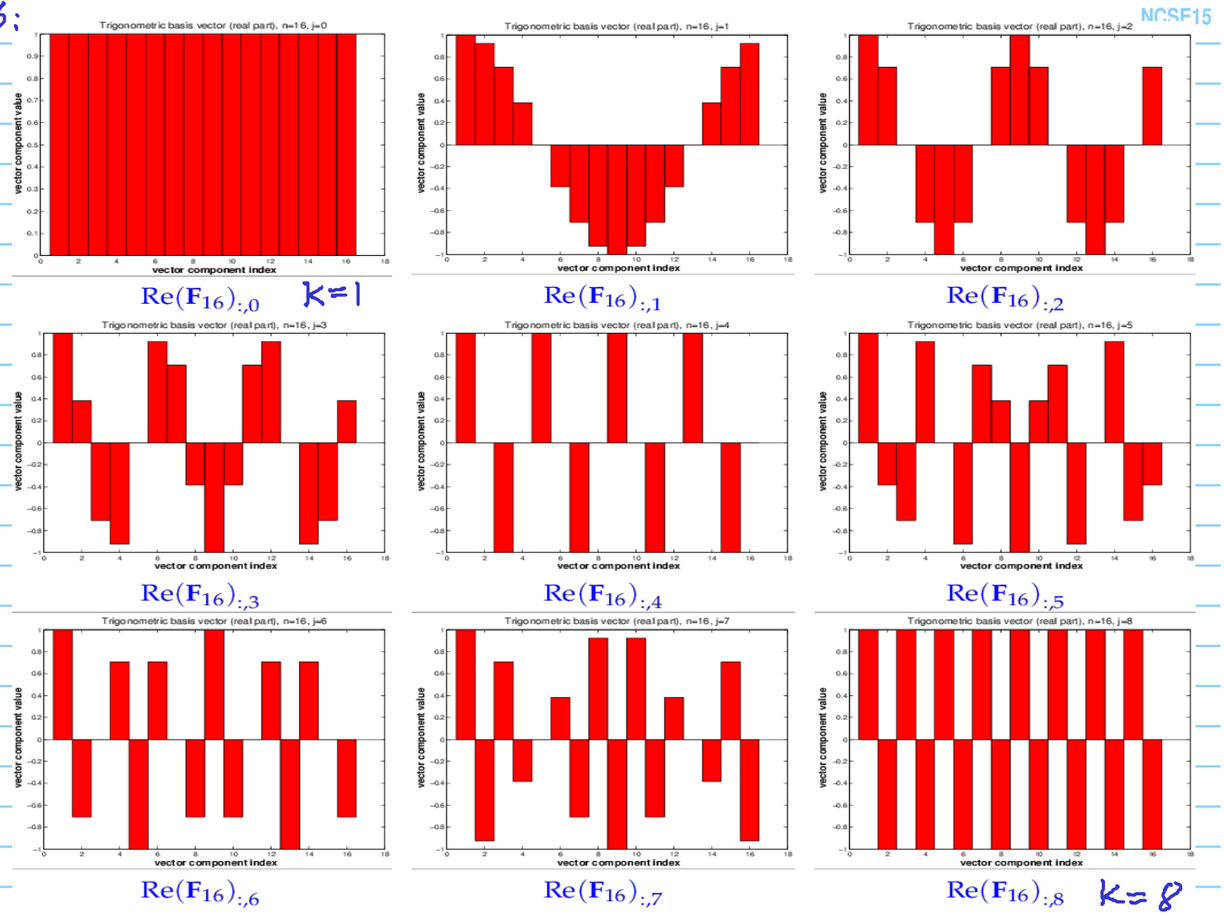
"slow oscillation/low frequency" "fast oscillation/high frequency" "slow oscillation/low frequency"

$n = 16$

DFT $\hat{=}$ Additive decomposition of a signal into frequency contribution



$n = 16$:



$\mathrm{Re}(\mathbf{F}_{16})_{:,0}$ $k=1$ $\mathrm{Re}(\mathbf{F}_{16})_{:,1}$ $\mathrm{Re}(\mathbf{F}_{16})_{:,2}$

$\mathrm{Re}(\mathbf{F}_{16})_{:,3}$ $\mathrm{Re}(\mathbf{F}_{16})_{:,4}$ $\mathrm{Re}(\mathbf{F}_{16})_{:,5}$

$\mathrm{Re}(\mathbf{F}_{16})_{:,6}$ $\mathrm{Re}(\mathbf{F}_{16})_{:,7}$ $\mathrm{Re}(\mathbf{F}_{16})_{:,8}$ $k=8$

C++11-code 4.2.33: DFT-based frequency filtering ➙ GITLAB

```
void freqfilter(const VectorXd& y, int k,
                VectorXd& low, VectorXd& high) {
  const VectorXd::Index n = y.size();
  if (n%2 != 0)
    throw std::runtime_error("Even vector length required!");
  const VectorXd::Index m = y.size()/2;

  Eigen::FFT<double> fft; // DFT helper object
  VectorXcd c = fft.fwd(y); // Perform DFT of input vector

  VectorXcd clow = c;
  // Set high frequency coefficients to zero, Fig. 146
  for (int j = -k; j <= +k; ++j) clow(m+j) = 0;
  // (Complementary) vector of high frequency coefficients
  VectorXcd chigh = c - clow;

  // Recover filtered time-domain signals
  low = fft.inv(clow).real();
  high = fft.inv(chigh).real();
}
```


Frequency identification

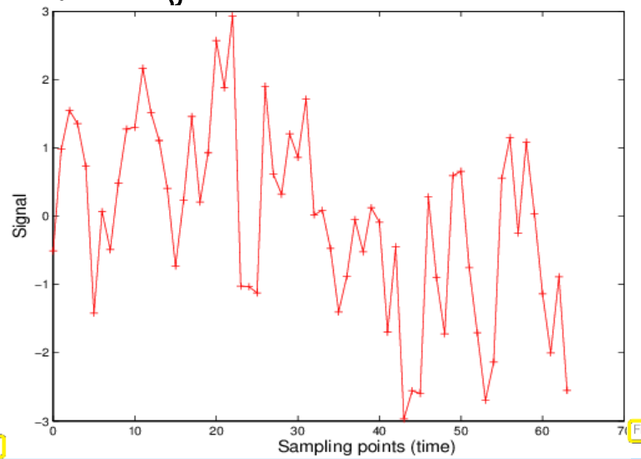


Fig. 142

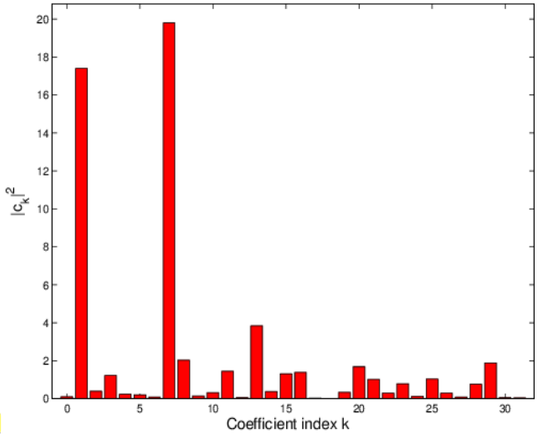


Fig. 143

Google: 'Vorlesungsverzeichnis'

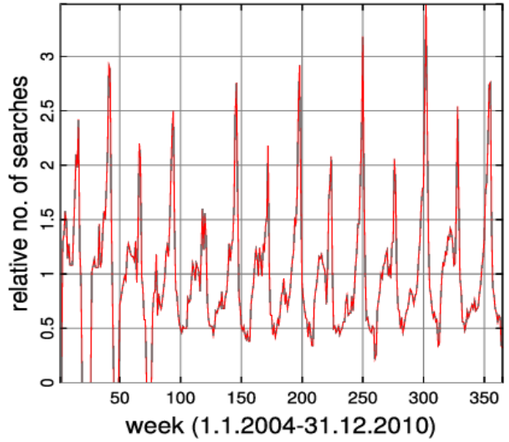


Fig. 144

Fourier spectrum

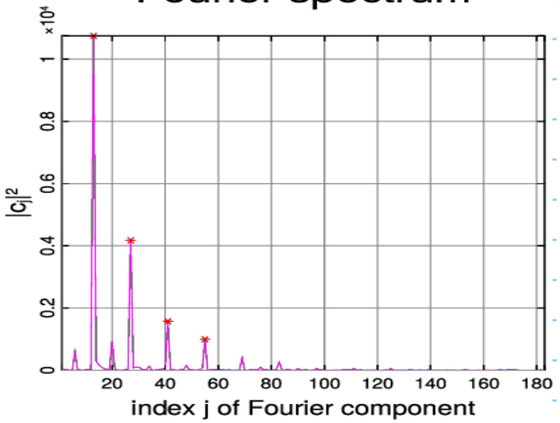


Fig. 145

4.2.4. 2D DFT

2D trigonometric basis of $\mathbb{C}^{m,n}$

$$\{V_{\nu,\mu}\} := \{\underbrace{[\omega_m^{\ell\nu}]_{\ell=0}^{m-1}\,([\omega_n^{k\mu}]_{k=0}^{n-1})^T}_{\in \mathbb{C}^{m,n}}\}_{\substack{\nu=0,\dots,m-1\\ \mu=0,\dots,n-1}}$$

Application: Image processing [Deblurring]

"pixel crosstalk" ↳ $P \in \mathbb{R}^{m,n}$

$$c_{\ell,j} = \sum_{k=-L}^{L}\sum_{q=-L}^{L} s_{k,q}\,(P)_{\ell+k,j+q}\,, \quad L \ll m,n$$

$s_{k,q}$: point spread function (PSF)

[periodic cont. of P assumed]

$$C = \mathcal{B}(P)$$

↳ linear blurring operator $\mathcal{B}: \mathbb{C}^{m,n} \to \mathbb{C}^{m,n}$

$$\Big(\mathcal{B}\big(\underbrace{\omega_m^{\nu k}\omega_n^{\mu q}}_{V_{\mu\nu}}\big)_{k,q\in\mathbb{Z}}\Big)_{l,j} = \sum_{k=-L}^{L}\sum_{q=-L}^{L} s_{k,q}\,\omega_m^{\nu(l+k)}\omega_n^{\mu(j+q)} = \underbrace{\omega_m^{\nu l}\omega_n^{\mu j}}_{=(V_{\mu\nu})_{l,j}}\underbrace{\sum_{k=-L}^{L}\sum_{q=-L}^{L} s_{k,q}\,\omega_m^{\nu k}\omega_n^{\mu q}}_{\text{eigenvalue } \lambda_{\mu\nu}}.$$

$V_{\mu\nu}$ ← eigenvector

▷ $\mathcal{B}(V_{\mu\nu}) = \lambda_{\mu\nu} V_{\mu\nu}$

Expand P into 2D trigonometric basis

$$(P)_{\ell,k} = \underbrace{\sum_{\nu=0}^{m-1}\sum_{\mu=0}^{n-1} (Y)_{\nu,\mu}\,\omega_m^{\ell\nu}\omega_n^{\mu k}}_{\text{2x DFT}}$$

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

$$(C)_{k_1,k_2} = \sum_{j_1=0}^{m-1}\big(\mathbf{F}_n(\mathbf{Y})_{j_1,:}\big)_{k_2}\omega_m^{j_1k_1} \quad\blacktriangleright\quad \boxed{\mathbf{C} = \mathbf{F}_m(\mathbf{F}_n\mathbf{Y}^\top)^\top = \mathbf{F}_m\mathbf{Y}\mathbf{F}_n}\,. \tag{4.2.46}$$

$$\mathbf{C} = \sum_{j_1=0}^{m-1}\sum_{j_2=0}^{n-1} y_{j_1,j_2}(\mathbf{F}_m)_{:,j_1}(\mathbf{F}_n)_{:,j_2}^\top \;\Rightarrow\; \boxed{\mathbf{Y} = \mathbf{F}_m^{-1}\mathbf{C}\mathbf{F}_n^{-1} = \frac{1}{mn}\overline{\mathbf{F}}_m\mathbf{C}\overline{\mathbf{F}}_n}\,. \tag{4.2.47}$$

Fourier basis coefficients ↑ ↑ pixel image

C++11 code 4.2.48: Two-dimensional discrete Fourier transform ↦ GITLAB

```
template <typename Scalar>
void fft2(Eigen::MatrixXcd &C, const Eigen::MatrixBase<Scalar> &Y) {
  using idx_t = Eigen::MatrixXcd::Index;
  const idx_t m = Y.rows(),n=Y.cols();
  C.resize(m,n);
  Eigen::MatrixXcd tmp(m,n);

  Eigen::FFT<double> fft; // Helper class for DFT
  // Transform rows of matrix Y
  for (idx_t k=0;k<m;k++) {
    Eigen::VectorXcd tv(Y.row(k));
    tmp.row(k) = fft.fwd(tv).transpose();
  }

  // Transform columns of temporary matrix
  for (idx_t k=0;k<n;k++) {
    Eigen::VectorXcd tv(tmp.col(k));
    C.col(k) = fft.fwd(tv);
  }
}
```

↳ DFT-based deblurring: $\Leftrightarrow$ $\mathcal{B}^{-1}$

(i) fft2: $\mathcal{P} \to Y$

(ii) Y. cwise Quotient $([\lambda_{pq}])$

(iii) inverse fft2: (4.2.47)

Estimating PSF:

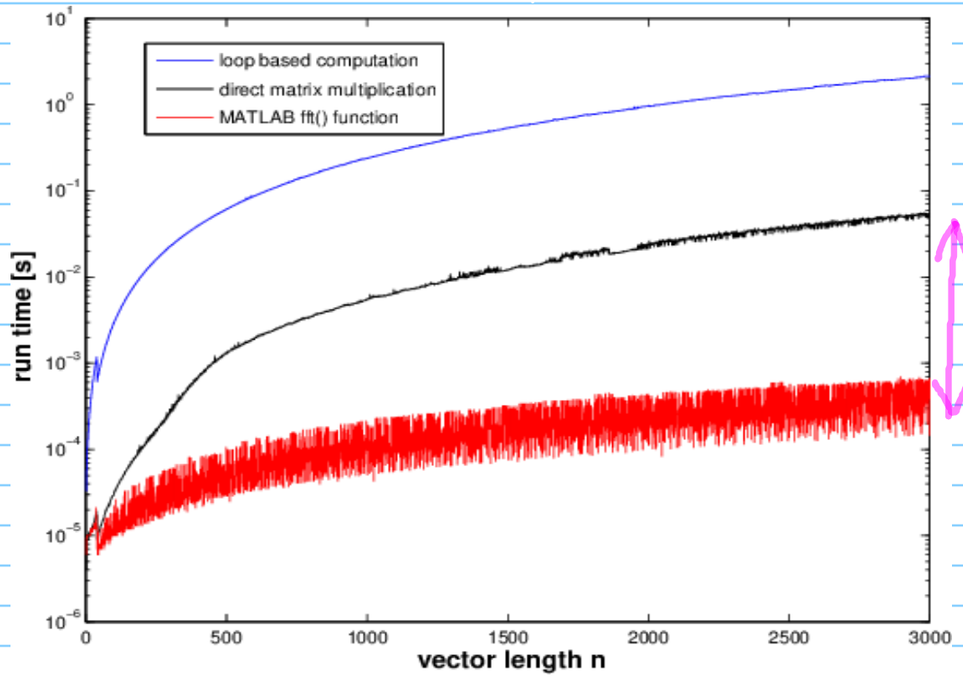
$P_i \hat{=}$ test images

$C_i \hat{=}$ blurred images $\quad i = 1, \ldots, m$

$$[S_{k,q}] := \operatorname*{argmin}_{[r_{k,q}]} \sum_{i=1}^{m} \| \mathcal{B}([r_{k,q}]; P_i) - C_i \|_F^2$$

4.3. Fast Fourier Transform (FFT)

Naive DFT: $\underline{c} = F_n \underline{y}$: cost $= O(n^2)$



DFT for $n = 2m$: [Gauss 1805]

$$c_k = \sum_{j=0}^{n-1} y_j \omega_n^{kj} \qquad [\omega^{kj} = e^{\frac{2\pi i}{n} kj}]$$

$$= \sum_{j=0}^{m-1} y_{2j} \omega_m^{kj} + \sum_{j=0}^{m-1} y_{2j+1} \omega_{2m}^{k(2j+1)}$$

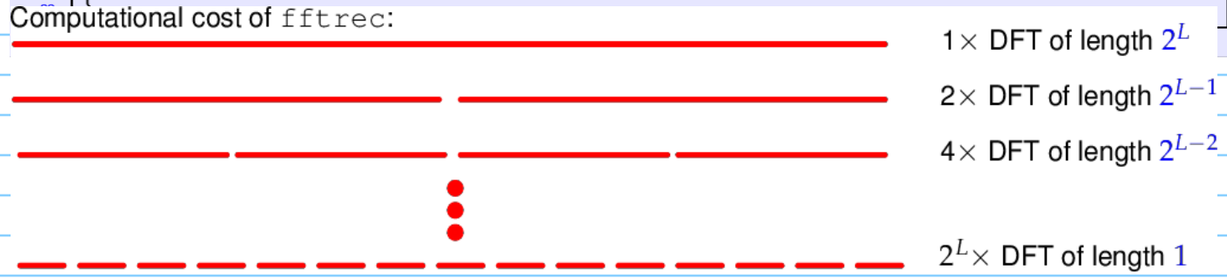
$$= \underbrace{\sum_{j=0}^{m-1} y_{2j} \omega_m^{kj}}_{\text{DFT of length } m} + \omega_n^k \underbrace{\sum_{j=0}^{m-1} y_{2j+1} \omega_m^{kj}}_{\text{DFT of length } m}$$

$n = 2^L \Rightarrow$ recursion

C++11 code 4.3.5: Recursive FFT → GITLAB

```
// Recursive DFT for vectors of length n = 2^L
VectorXcd fftrec(const VectorXcd& y) {
  using idx_t = VectorXcd::Index;
  using comp = std::complex<double>;
  // Nothing to do for DFT of length 1
  const idx_t n = y.size();
  if (n == 1) return y;
  if (n % 2 != 0) throw std::runtime_error("size(y) must be even!");
  const Eigen::Map<const
      Eigen::Matrix<comp,Eigen::Dynamic,Eigen::Dynamic,Eigen::RowMajor>>
    Y(y.data(),n/2,2); // for selecting even and off compoents
  const VectorXcd c1 = fftrec(Y.col(0)),c2 = fftrec(Y.col(1));
  const comp omega = std::exp(-2*M_PI/n*comp(0,1)); // root of unity
                                                    ω_n
  comp s(1.0,0.0);
  VectorXcd c(n);
  // Scaling of DFT of odd components plus periodic copying
  for (long k = 0; k < n; ++k) {
    c(k) = c1(k%(n/2)) + c2(k%(n/2))*s;
    s *= omega;
  }
  return c;
}
```

Computational cost of `fftrec`:



▷ Cost $= O(Ln) = O(\log_2 n \cdot n)$

$\ll O(n^2)$ for $n \gg 1$

Rationale for reduction to DFT !

Asymptotic complexity of discrete periodic convolution, see Code 4.2.25:

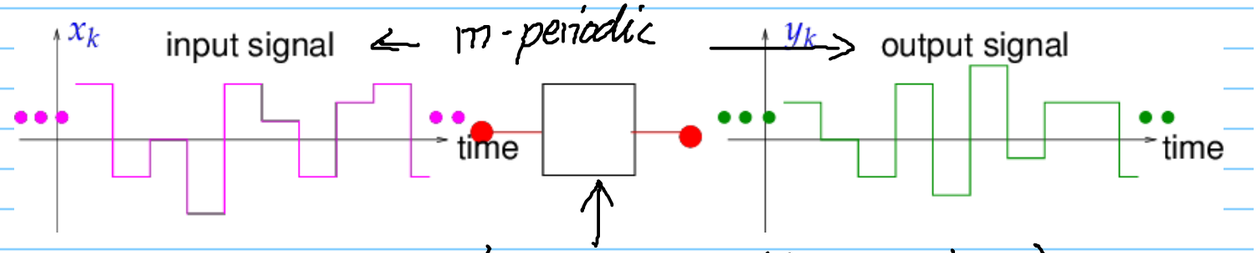
Cost(`pconvfft(u,x)`, $\mathbf{u},\mathbf{x} \in \mathbb{C}^n$) = $O(n \log n)$.

Asymptotic complexity of discrete convolution, see Code 4.2.26:

Cost(`myconv(h,x)`, $\mathbf{h},\mathbf{x} \in \mathbb{C}^n$) = $O(n \log n)$.

4.5. Toeplitz Matrix Technique

Ex: Parameter estimation for LT-FIR channel



Impulse response $(h_0, \ldots, h_{n-1})$, $m \geq n$

$\underline{y} = \underline{h} *_m \underline{x} \hat{=}$ m equ.

↳ overdetermined LSE for $\underline{h}$

$\Leftrightarrow$ least squares problem

$$\|\mathbf{Ah} - \mathbf{y}\|_2 = \left\| \begin{bmatrix} x_0 & x_{-1} & \cdots & & \cdots & x_{1-n} \\ x_1 & x_0 & x_{-1} & & & \vdots \\ \vdots & x_1 & x_0 & \ddots & & \\ & & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & \ddots & x_{-1} \\ x_{n-1} & & & & x_1 & x_0 \\ x_n & x_{n-1} & & & & x_1 \\ \vdots & & & & & \vdots \\ x_{m-1} & \cdots & & & \cdots & x_{m-n} \end{bmatrix} \begin{bmatrix} h_0 \\ \vdots \\ \underline{x} \\ \vdots \\ h_{n-1} \end{bmatrix} - \begin{bmatrix} y_0 \\ \vdots \\ \underline{b} \\ \vdots \\ y_{m-1} \end{bmatrix} \right\|_2 \to \min .$$

A

$$(A)_{ij} = x_{i-j}$$

Normal equ.: $A^T A x = A^T b$

$$(A^T A)_{ij} = \sum_{k=0}^{m-1} x_{i-k}\, x_{j-k} =: z_{i-j} : (z_j)$$

for m-periodic

sum over one period → index shifting

Toeplitz matrix ("data space")

Definition 4.5.8. Toeplitz matrix

$\mathbf{T} = (t_{ij})_{i,j=1}^{n} \in \mathbb{K}^{m,n}$ is a Toeplitz matrix, if there is a vector $\mathbf{u} = [u_{-m+1}, \ldots, u_{n-1}] \in \mathbb{K}^{m+n-1}$ such that $t_{ij} = u_{j-i}$, $1 \le i \le m$, $1 \le j \le n$.

$$\mathbf{T} = \begin{bmatrix} u_0 & u_1 & \cdots & & \cdots & u_{n-1} \\ u_{-1} & u_0 & u_1 & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & u_1 \\ u_{1-m} & \cdots & & \cdots & u_{-1} & u_0 \end{bmatrix}$$

Efficient alg. for $y = Tx$

Idea [m = n]:

$$\mathbf{C} = \left[\begin{array}{cccccc|ccccc} u_0 & u_1 & \cdots & & \cdots & u_{n-1} & 0 & u_{1-n} & \cdots & & u_{-1} \\ u_{-1} & u_0 & u_1 & & & \vdots & u_{n-1} & 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \vdots & \vdots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \ddots & & & & & \ddots & \\ \vdots & & & \ddots & \ddots & u_1 & \vdots & & \ddots & \ddots & u_{1-n} \\ u_{1-n} & \cdots & & \cdots & u_{-1} & u_0 & u_1 & & & u_{n-1} & 0 \\ \hline 0 & u_{1-n} & \cdots & & \cdots & u_{-1} & u_0 & u_1 & \cdots & & u_{n-1} \\ u_{n-1} & 0 & \ddots & & & \vdots & u_{-1} & u_0 & u_1 & & \vdots \\ \vdots & \ddots & \ddots & & & & \vdots & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & & & \vdots & & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & u_{1-n} & & \vdots & & & \ddots & u_1 \\ u_1 & & & u_{n-1} & 0 & & u_{1-n} & \cdots & & u_{-1} & u_0 \end{array}\right]$$

A circulant matrix: $C = \begin{bmatrix} T & * \\ * & * \end{bmatrix}$

$$C \begin{bmatrix} x \\ 0 \end{bmatrix} = \begin{bmatrix} Tx \\ * \end{bmatrix}$$

$\triangleright$ Cost(Tx) = Cost(Cx) = $O(n \log n)$