Technische Universität Braunschweig

Institute for Numerical Analysis

Ramp Up Mathematics — Numerical Analysis Ramp Up for Data Science

Matthias Bollhöfer, SS 2024

Contents I

Contents II

Discrete Fourier Transformation

- signal processing, noise reduction often require approximation of functions which allow suppressing high frequency modes
- frequencies are modeled by periodic functions $\sin, \cos$
- this leads to the Fourier transformation where functions are approximated by Fourier series

Definition 6.1

$$P(x) = c_{-r}e^{-r\jmath x} + c_{-r+1}e^{(-r+1)\jmath x} + \cdots + c_{-1}e^{-\jmath x} + c_0 + c_1 e^{\jmath x} + \cdots + c_s e^{s\jmath x},$$

where $\jmath = \sqrt{-1}$, $r, s \in \mathbb{N}$, *is called trigonometric polynomial*

Example 6.1 (trigonometric polynomial)

$$P(x) = e^{3\jmath x} + e^{-3\jmath x} =$$

Fourier Series

Definition 6.2

Let $f : [0, 2\pi) \to \mathbb{C}$ such that $\int_0^{2\pi} |f(x)|^2 \, dx < \infty$. Then we call

$$\hat{f}(x) = \sum_{k=-\infty}^{\infty} c_k e^{\jmath kx},$$

where

$$c_k = \langle f, e^{\jmath kx} \rangle = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-\jmath kx} \, dx$$

the Fourier series associated with f. The coefficients c_k are called Fourier coefficients.

Theorem 6.1 (Representation of a function by its Fourier series)

Let $f : [0, 2\pi) \to \mathbb{C}$ such that $\int_0^{2\pi} |f(x)|^2 \, dx < \infty$. Then we have

$$\int_0^{2\pi} |f(x) - \hat{f}(x)|^2 \, dx = 0$$

Example 6.2 (Signal)

$$f(x) = \begin{cases} -1 & x \in [0, \pi] \\ 1 & x \in (\pi, 2\pi] \end{cases}$$

$$k \neq 0 : \; 2\pi c_k = -\int_0^{\pi} e^{-\jmath kx}\, dx + \int_{\pi}^{2\pi} e^{-\jmath kx}\, dx =$$

$$=$$

$$\Rightarrow c_0 = \quad , c_{2l} = \quad , \; c_{2l+1} = \qquad \Rightarrow \hat{f}(x) =$$

real-valued representation: we have $c_{-k} = \overline{c_k}$*:*

$$\hat{f}(x) = \qquad = \boxed{}$$

Fourier Polynomials

In order to approximate the Fourier series itself we have to solve two tasks:

1. we have to truncate the series to a *finite Fourier polynomial* $\hat{f}_{-r,s}(x) = \sum_{k=-r}^{s} c_k e^{\mathrm{j}kx}$
2. integrals c_k have to be computed numerically

Task 1.

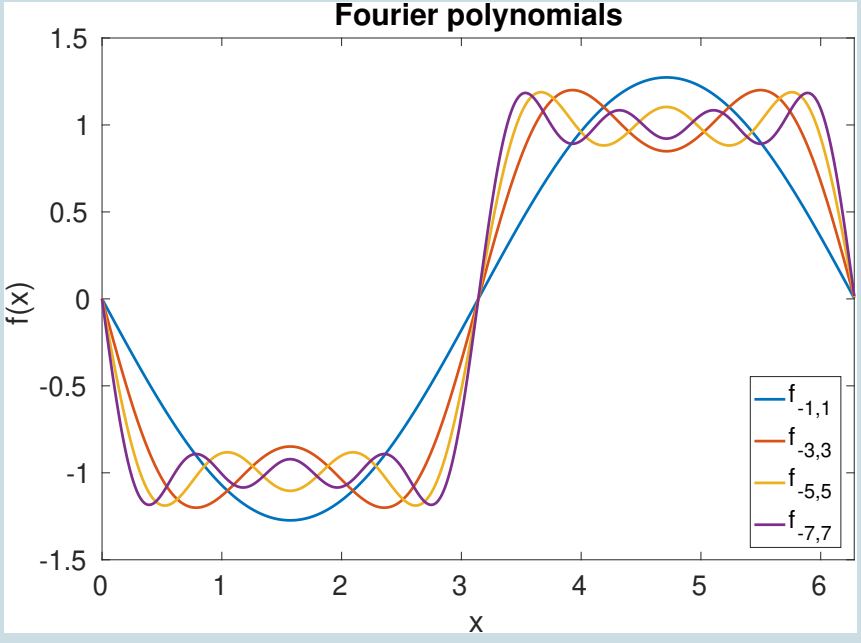
Example 6.3 (Signal approximated via Fourier transformation)

$$f(x) = \begin{cases} -1 & x \in [0, \pi] \\ 1 & x \in (\pi, 2\pi) \end{cases}$$

$$\hat{f}_{-2r-1,2r+1}(x) = -\frac{4}{\pi} \sum_{l=0}^{r} \frac{\sin((2l+1)x)}{2l+1}$$

$$\hat{f}_{-1,1}(x) =$$

$$\hat{f}_{-3,3}(x) =$$



Fourier Polynomials

Task 2. replace integral by quadrature formula, here: piecewise trapezoidal rule

- set $h = \frac{2\pi}{n}$ and use n grid points

$$x_l = lh, \ l = 0, \ldots, n$$

- replace integral

$$\int_0^{2\pi} f(x)\, dx$$

by piecewise trapezoidal rule for periodic functions

$$T(h) = h \sum_{l=0}^{n-1} f(x_l)$$

Note that for periodic functions we extend f to the real axis via $f(x + 2\pi) \equiv f(x)$, in particular this way we extend $f(2\pi)$ via $f(2\pi) := f(0)$

Fourier Polynomials

- Set $g(x) = e^{\imath kx}$. We thus replace the complex-valued integral $\langle f, g\rangle$:

$$\langle f, g\rangle = \frac{1}{2\pi}\int_0^{2\pi} f(x)\overline{g(x)}\,dx \approx \frac{1}{n}\sum_{l=0}^{n-1} f(x_l)\overline{g(x_l)} =: (f, g)$$

The expressions $\langle f, g\rangle$ and (f, g) are *scalar products* and induce norms

$$\|f\| := \sqrt{\frac{1}{2\pi}\int_0^{2\pi} |f(x)|^2\,dx} \text{ and } |f| := \sqrt{\frac{1}{n}\sum_{l=0}^{n-1} |f(x_l)|^2}$$

Definition 6.3

Let $f : [0, 2\pi) \to \mathbb{C}$ piecewise continuous and let $n, r, s \in \mathbb{N}$. The we call

$$\tilde{f}_{-r,s}(x) = \sum_{k=-r}^{s} \tilde{c}_k e^{\imath kx}, \text{ where } \tilde{c}_k = (f, e^{\imath kx}) = \frac{1}{n}\sum_{l=0}^{n-1} f(x_l)e^{-\imath kx_l}$$

discrete Fourier transform of f.

Discrete Fourier Transformation

Example 6.4 (Signal approximated by discrete Fourier transform)

$$f(x) = \begin{cases} -1 & x \in (0, \pi) \\ 0 & x = 0, \pi, 2\pi \\ 1 & x \in (\pi, 2\pi) \end{cases}$$

Let n be even (for simplicity) $\Rightarrow \tilde{c}_k =$

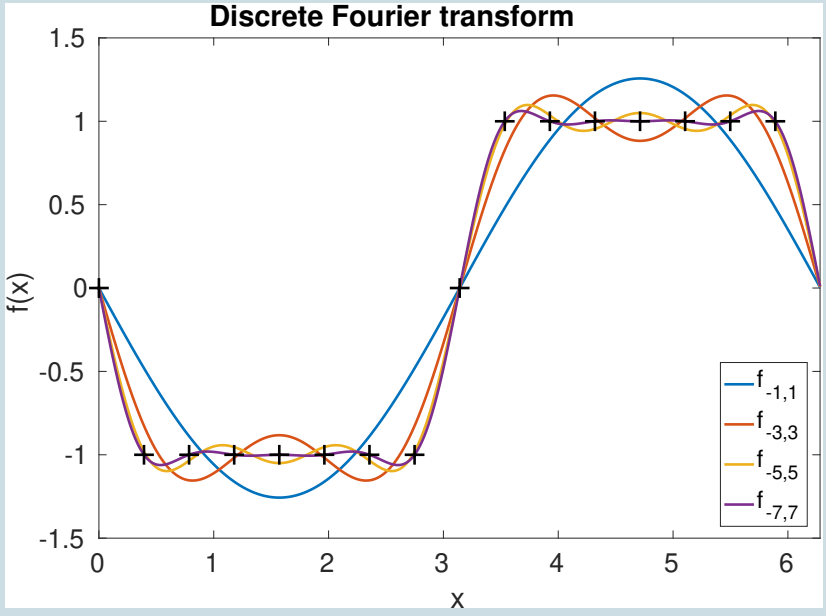
For $n = 16$ *we obtain* $\tilde{c}_{2l} = 0$,

$$\tilde{c}_1 \approx -1.26, \quad \tilde{c}_3 \approx -0.374,$$
$$\tilde{c}_5 \approx -0.167, \quad \tilde{c}_7 \approx -0.0497.$$

$$\tilde{f}_{-1,1}(x) =$$

$$\tilde{f}_{-3,3}(x) =$$

$$\tilde{f}_{-5,5}(x) =$$



Discrete Fourier Transformation

- regarding the norms $\|\bullet\|, |\bullet|$, Fourier transforms are the optimal approximations to f w.r.t. the space spanned by $\left\{e^{\jmath kx}\right\}_{k=-r,\ldots,s}$
- $\hat{f}_{-r,s}$, respectively $\tilde{f}_{-r,s}$ are the associated orthogonal projections of f
- the functions $\left\{e^{\jmath kx}\right\}_{k=-r,\ldots,s}$ form an orthonormal system

Let us summarize the principle computation of a discrete Fourier transform

1. Choose $n \in \mathbb{N}$ and the grid points $(x_0, f_0), \ldots, (x_{n-1}, f_{n-1})$, where $x_l = l\frac{2\pi}{n}$.
2. Select r, s, such that $r + s \leqslant n - 1$ and compute the coefficients of the discrete Fourier transform
$$\tilde{c}_k = \frac{1}{n} \sum_{l=0}^{n-1} f_l e^{-\jmath k x_l}, \quad k = -r, \ldots, s.$$
3. Then our discrete Fourier transform is given by
$$\tilde{f}_{-r,s}(x) = \sum_{k=-r}^{s} \tilde{c}_k e^{\jmath kx}.$$

Fast Fourier Transformation (FFT)

- efficient computation of the coefficients $\tilde{c}_{-r}, \ldots, \tilde{c}_s$. is performed via the *fast Fourier transformation (FFT)*
- note that for $r + s = n - 1$ we have $\tilde{c}_{-k} = \tilde{c}_{n-k}$, i.e., the coefficients are cyclic
- For simplicity we set $r = 0$ and $s = n - 1 \rightarrow \tilde{c}_0, \ldots, \tilde{c}_{n-1}$.
- important for the efficient computation: for n only use powers of two

$$n = 2^p,$$

in this case there is a simple recursion formula

Fast Fourier Transformation (FFT)

- our goal is to compute the coefficients

$$\tilde{c}_k = \frac{1}{n} \sum_{l=0}^{n-1} f_l e^{-\jmath \frac{2\pi kl}{n}} = \frac{1}{n} \sum_{l=0}^{n-1} f_l \omega_n^{kl}, \text{ where } \omega_n = e^{-\jmath \frac{2\pi}{n}}$$

  efficiently.
- formally this can be read as a matrix-vector product of type

$$\begin{bmatrix} \tilde{c}_0 \\ \vdots \\ \tilde{c}_{n-1} \end{bmatrix} = \frac{1}{n} W \begin{bmatrix} f_0 \\ \vdots \\ f_{n-1} \end{bmatrix}, \text{ where } W = \left[\omega_n^{kl}\right]_{k,l=0,\ldots,n-1}.$$

- clue: use the fact that ω_n is n–th unit root, i.e., all powers of ω_n are also located on the unit circle, exponents can be computed easily

Fast Fourier Transformation (FFT)

the basic sketch of the FFT works as follows. Set $m := \frac{n}{2}$.

FFT for $f_0, \ldots, f_{n-1}$.

split the $\tilde{c}_0, \ldots, \tilde{c}_{n-1}$ to be computed into even and odd indices.

$\downarrow$ $\qquad\qquad$ $\downarrow$

$\tilde{c}_0, \tilde{c}_2, \ldots, \tilde{c}_{n-2}$ $\qquad\qquad$ $\tilde{c}_1, \tilde{c}_3, \ldots, \tilde{c}_{n-1}$

reduction step: $\qquad\qquad$ reduction step:

compute auxiliary quantities $z_0, \ldots, z_{m-1}$ $\qquad\qquad$ compute auxiliary quantities $z_m, \ldots, z_{n-1}$

call FFT for $z_0, \ldots, z_{m-1}$ $\qquad\qquad$ call FFT for $z_m, \ldots, z_{n-1}$

$\downarrow$ $\quad$ $\downarrow$ $\qquad\qquad$ $\downarrow$ $\quad$ $\downarrow$

$\tilde{c}_0, \tilde{c}_4, \tilde{c}_8, \ldots$ $\qquad$ $\tilde{c}_2, \tilde{c}_6, \tilde{c}_{10}, \ldots$ $\qquad\qquad$ $\tilde{c}_1, \tilde{c}_5, \tilde{c}_9, \ldots$ $\qquad$ $\tilde{c}_3, \tilde{c}_7, \tilde{c}_{11}, \ldots$

Fast Fourier Transformation (FFT)

- FFT of size $n = 2^p$ can be traced back to two FFTs of size $m = \frac{n}{2} = 2^{p-1}$
- additional computational effort of complexity $\mathcal{O}(n)$ is required for the reduction step
- in total, the same principle is recursively applied p times, where $p = \log_2(n)$
- instead of a matrix vector multiplication that costs $\mathcal{O}(n^2)$ we obtain a complexity of $\mathcal{O}(n \log_2 n)$
- two FFTs of half size can be computed independently, implementation on a parallel computer is feasible

Fast Fourier Transformation (FFT)

1. discuss even indices $k = 2q$:
   - for the reduction step compute the quantities $z_l := f_l + f_{m+l}$ for $l = 0, \ldots, m-1$.
   - Note that $\omega_n^{2q(m+l)} =$ $=$ $=$
   - for the reduction step we sum up the terms with indices l and $m+l$:

$$\Rightarrow n \cdot \tilde{c}_{2q} = \underbrace{\sum_{l=0}^{n-1} f_l \omega_n^{2ql}}_{FFT_n(f_0,\ldots,f_{n-1})} =$$

2. analogously proceed for the odd indices $k = 2q+1$:
   - compute quantities $z_{m+l} = (f_l - f_{m+l})\omega_n^l$ for $l = 0, \ldots, m-1$
   - We have $\omega_n^{(2q+1)(m+l)} =$ $=$ $=$ $=$
   - Again, sum up the terms with indices l and $m+l$:

$$\Rightarrow n \cdot \tilde{c}_{2q+1} = \underbrace{\sum_{l=0}^{n-1} f_l \omega_n^{(2q+1)l}}_{FFT_n(f_0,\ldots,f_{n-1})} = \quad =$$

Fast Fourier Transformation (FFT)

Algorithm 6.1 (FFT Blue Print)

Input: *function values* $f_0, \ldots, f_{n-1} \in \mathbb{C}$

Output: *coefficients* $n \cdot \tilde{c}_0, \ldots, n \cdot \tilde{c}_{n-1}$ *of the discrete Fourier transform*

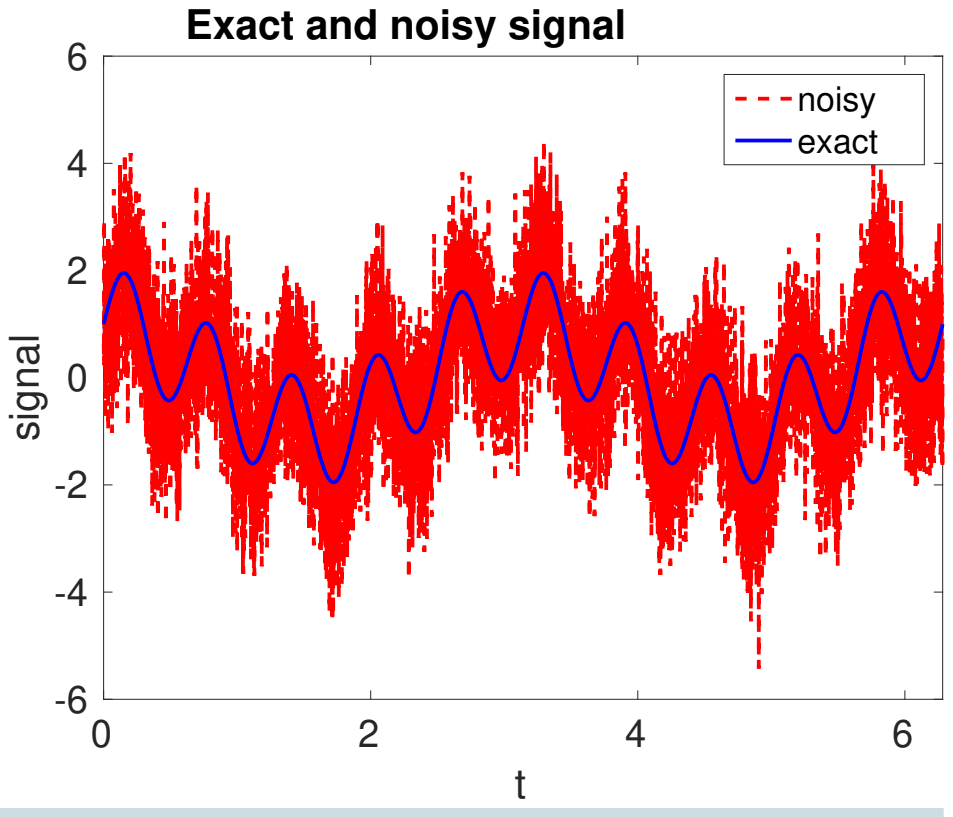
1: If $n = 1$
2: $\quad n\tilde{c}_0 \leftarrow f_0$
3: Else
4: $\quad$ *{reduction step for even indices}*
5: $\quad z_l \leftarrow f_l + f_{m+l},\ l = 0, \ldots, m-1$
6: $\quad [n\tilde{c}_0, n\tilde{c}_2, \ldots, n\tilde{c}_{n-2}] = FFT(z_0, \ldots, z_{m-1})$
7: $\quad$ *{reduction step for odd indices}*
8: $\quad z_{m+l} \leftarrow (f_l - f_{m+l})\omega_n^l,\ l = 0, \ldots, m-1$
9: $\quad [n\tilde{c}_1, n\tilde{c}_3, \ldots, n\tilde{c}_{n-1}] = FFT(z_m, \ldots, z_{n-1})$
10: [End] If

Remark: For the proper Fourier coefficients $\tilde{c}_k$ we have to eventually divide by n.

Fast Fourier Transformation (FFT)

Example 6.5 (Signal Denoising)

- *We are seeking to reconstruct the exact signal* $g(t) = sin(10t) + cos(2t)$ *in* $[0, 2\pi]$
- *we let the signal be superposed by a normally distributed noise* $n(t)$, $\tilde{g}(t) = g(t) + n(t)$
- *we sample the signal with a step size of* $\Delta t = 0.001$.
- *we compute the Fourier coefficients* $c_0, \ldots, c_{n-1}$ *via FFT*
- *we only consider those coefficients* $c_0, \ldots, c_{n-1}$ *which are greatest in magnitude and skip the the others*
- *watch the associated Fourier transform* $\hat{g}(t)$



Fast Fourier Transformation (FFT)

Example 6.6 (Signal Denoising (ctd.))