FFT: Die schnelle Fourier-Transformation

Sebastian Gesemann

sgeseman@upb.de

Fahrplan

- Fourier-Transformation
  - kontinuierlich
  - diskret (DFT)
- DFT und FFT
- Anwendungsbeispiele der FFT
- Wie funktioniert die FFT ?
- DEMO!

Fourier-Transformation

- Jean Babtiste Joseph Fourier (französischer Mathematiker, 1768-1830)
- „Jede (reell- oder komplexwertige) periodische Funktion lässt sich als Summe von Sinus- und Cosinus-Funktionen darstellen.“

Fourier-Transformation

(komplex, kontinuierlich)

Gegeben: Eine Funktion

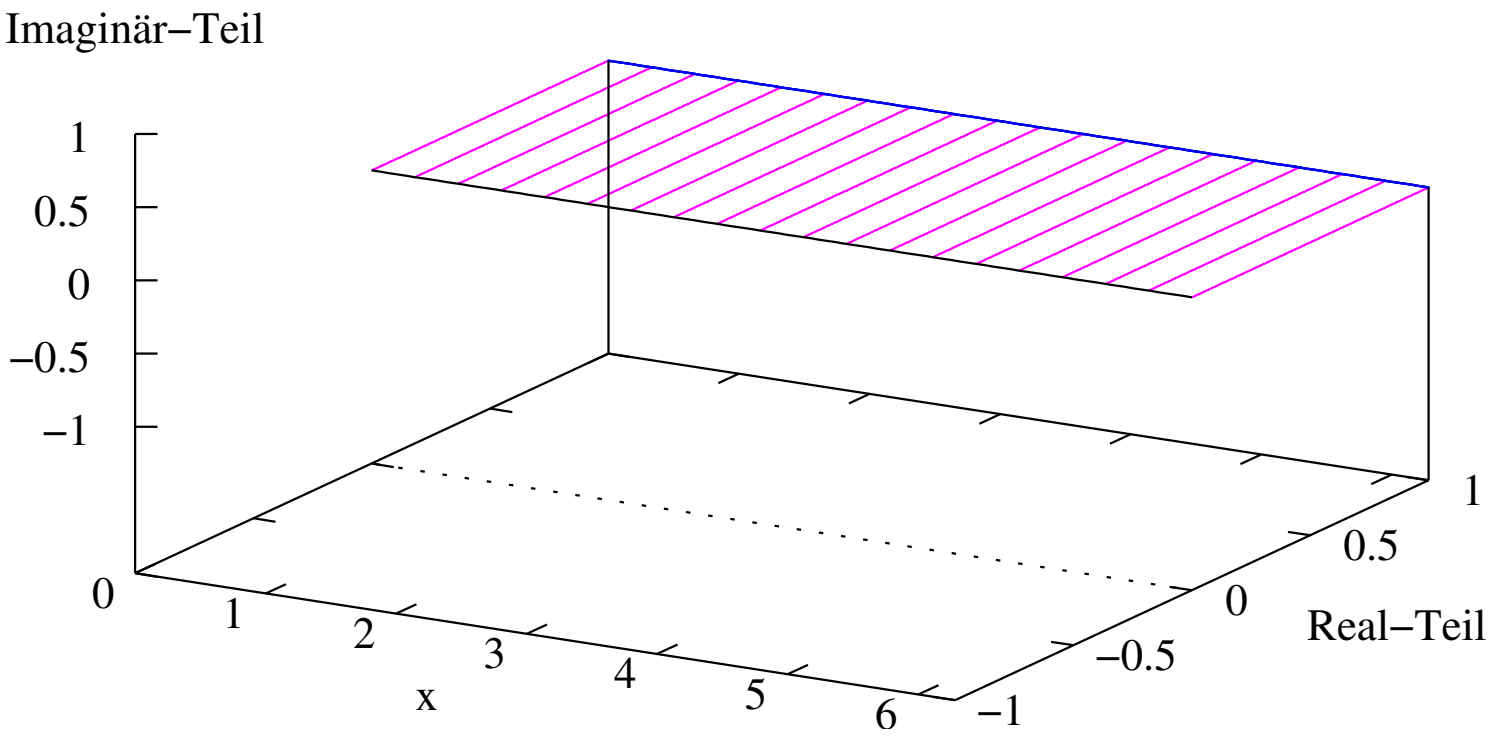
$$g : [0, 2\pi) \longrightarrow \mathbb{C}$$

Gesucht: Die Koeffizienten $a_f \in \mathbb{C}$ der komplexen „Fourier-Reihe“ mit

$$g(x) = \lim_{k \to \infty} \sum_{f=-k}^{k} a_f \, e^{\sqrt{-1} f x}$$

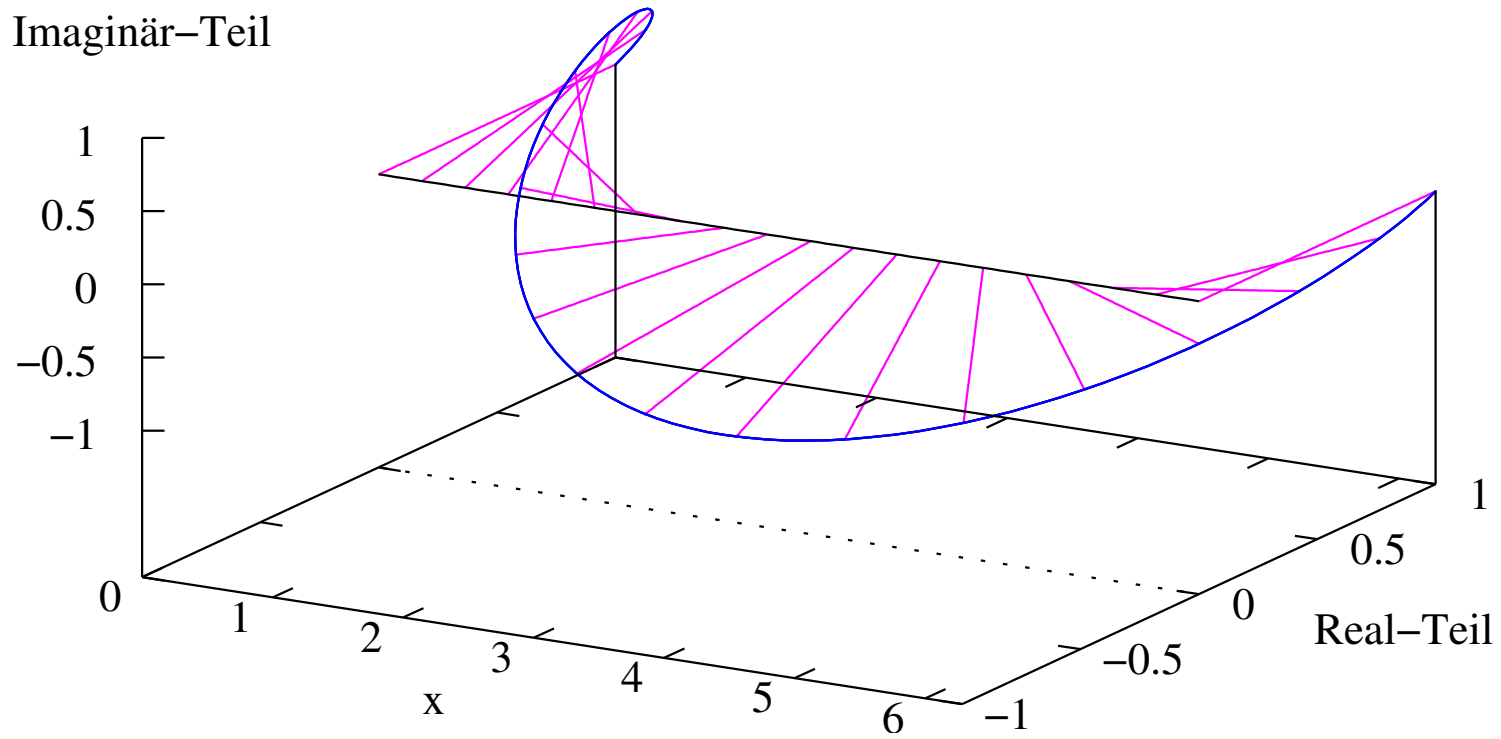
Fourier-Transformation

$$e^{\sqrt{-1}fx} \quad \text{für } f = 0$$



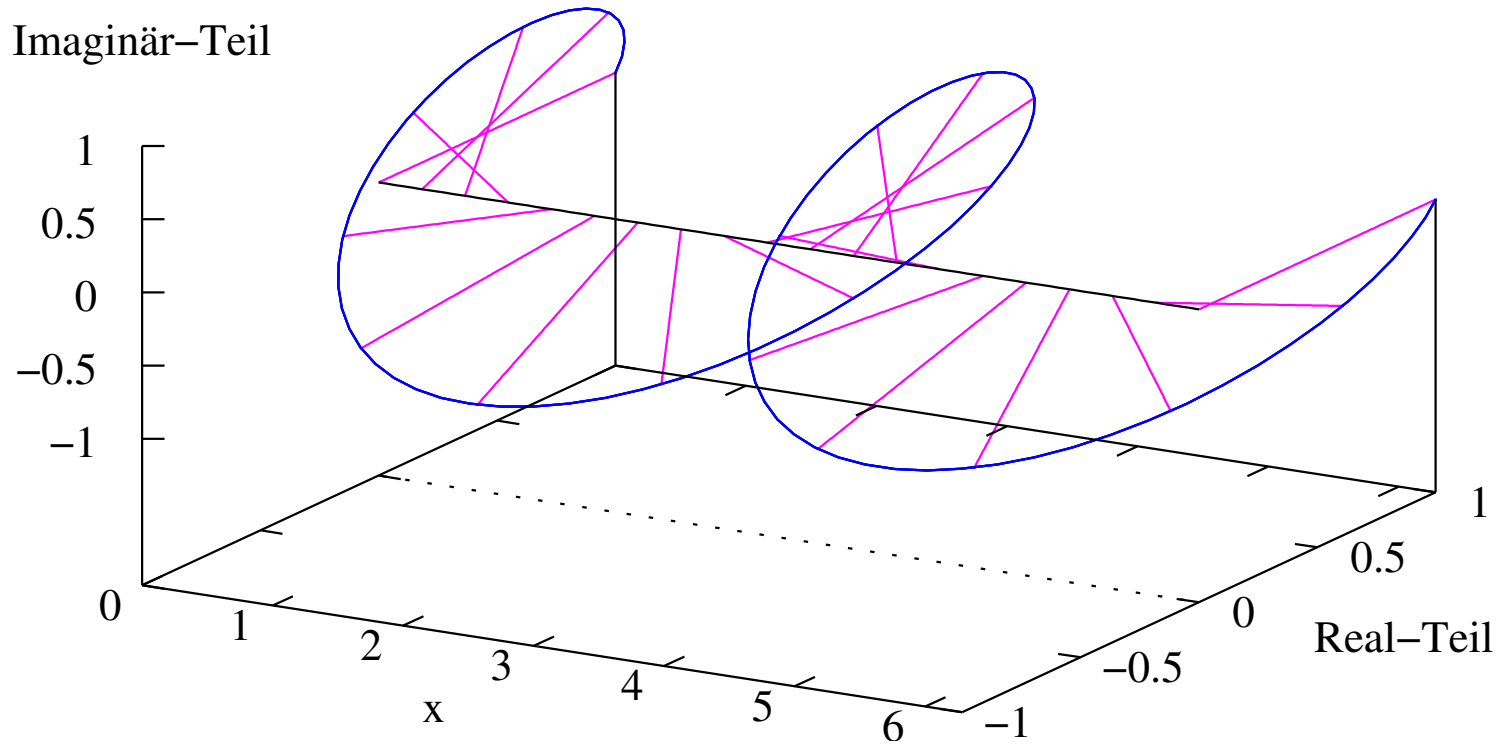
Fourier-Transformation

$$e^{\sqrt{-1}fx} \quad \text{für } f = 1$$



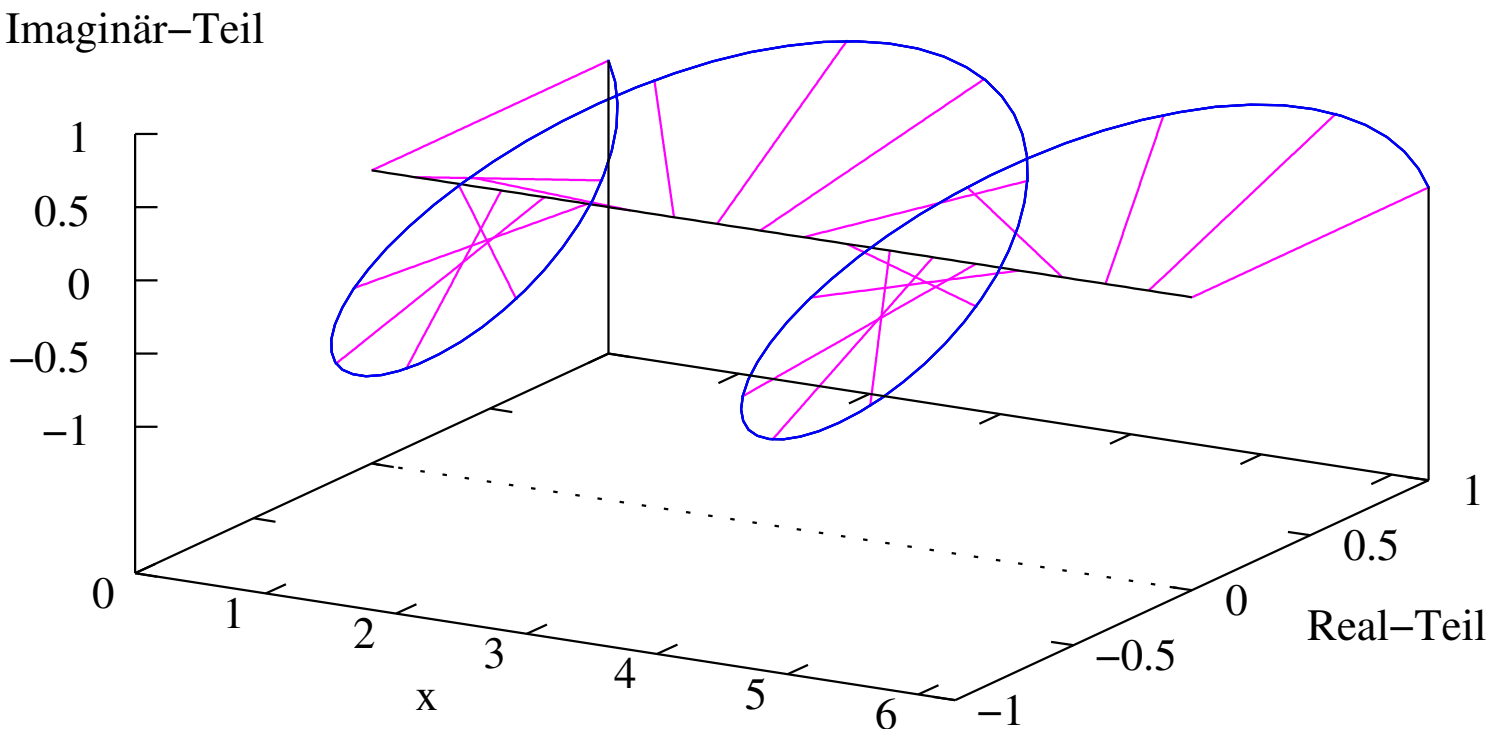
Fourier-Transformation

$e^{\sqrt{-1}fx}$ für $f = 2$



Fourier-Transformation

$$e^{\sqrt{-1}fx} \quad \text{für } f = -2$$



Fourier-Transformation

Definieren wir ein Skalarprodukt durch

$$< g, h >:= \int_0^{2\pi} g(x)\overline{h(x)}\,dx$$

kann gezeigt werden, dass die „Vektoren“

$$e^{\sqrt{-1}fx} \quad \text{für } f \in \mathbb{Z}$$

eine *orthogonale Basis* für den Vektorraum der Funktionen von $[0, 2\pi)$ nach $\mathbb{C}$ bilden (Körper: $\mathbb{C}$).

Fourier-Transformation

Damit lassen sich die Koeffizienten $a_f \in \mathbb{C}$ der Fourier-Reihe für die Funktion g durch

$$\begin{aligned} a_f &= \frac{<g, e^{\sqrt{-1}fx}>}{<e^{\sqrt{-1}fx}, e^{\sqrt{-1}fx}>} \\ &= \frac{1}{2\pi}\int_0^{2\pi} g(x)\overline{e^{\sqrt{-1}fx}}\,dx \\ &= \frac{1}{2\pi}\int_0^{2\pi} g(x)e^{-\sqrt{-1}fx}\,dx \end{aligned}$$

bestimmen.

Fourier-Transformation

(komplex, diskret)

- diskreter Definitionsbereich:
  $\mathcal{D} = \left\{ x \,|\, x = \frac{2\pi k}{n},\ k \in \{0, 1, \ldots, (n-1)\} \right\}$
- Endlich-dimensionaler Vektorraum (Dimension n)
- Transformation und inverse Transformation als Matrix-Vektor-Produkt über dem Körper $\mathbb{C}$ darstellbar

Fourier-Transformation

(komplex, diskret, als Matrix-Vektor-Produkt)

$$\begin{pmatrix} g(0) \\ g(1\frac{2\pi}{n}) \\ \vdots \\ g((n-1)\frac{2\pi}{n}) \end{pmatrix} = \begin{bmatrix} c_{0,0}^{(n)} & c_{0,1}^{(n)} & \cdots & c_{0,n-1}^{(n)} \\ c_{1,0}^{(n)} & c_{1,1}^{(n)} & \cdots & c_{1,n-1}^{(n)} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n-1,0}^{(n)} & c_{n-1,1}^{(n)} & \cdots & c_{n-1,n-1}^{(n)} \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

mit

$$c_{i,j}^{(n)} = e^{2\pi\sqrt{-1}\frac{ij}{n}} \ \forall i,j \in \{0,1,\ldots,(n-1)\}$$

Fourier-Transformation

Die Spalten dieser Matrix $C^{(n)}$ sind bezüglich des Skalarproduktes

$$< a, b >:= \sum_{i=0}^{n-1} a_i \overline{b_i}$$

für $a = (a_0, \ldots, a_{n-1})^T \in \mathbb{C}^n$, $b = (b_0, \ldots, b_{n-1})^T \in \mathbb{C}^n$

wieder orthogonal und besitzten jeweils die Länge $\sqrt{n}$ (Euklidische Norm). Außerdem gilt $c_{i,j}^{(n)} = c_{j,i}^{(n)}$.

$$\Rightarrow C^{(n)^{-1}} = \frac{1}{n}\overline{C^{(n)^T}} = \frac{1}{n}\overline{C^{(n)}}$$

Laufzeit von DFT & FFT

- Die DFT kann per Matrix-Vektor-Produkt berechnet werden und benötigt damit $O(n^2)$ Zeit.
- Die FFT ist ein *Algorithmus*, der die DFT in $O(n\log(n))$ Zeit berechnen kann. Der Algorithmus nutzt die spezielle Struktur der Matrizen C und C^{-1} aus.

Anwendungsbeispiele der FFT

- Andere wichtige Transformationen lassen sich in linearer Zeit auf die FFT reduzieren und damit auch in $O(n\log(n))$ berechnen.
  - Diskrete Cosinus-Transformation, DCT
  - Modifizierte diskrete Cosinus-Transformation, MDCT
- Analyse von digitalen Signalen („*Spectral-Analyse*“)
- Die FFT als Werkzeug für eine *„schnelle Faltung“* (Eine Operation aus der Signalverarbeitung, die zwei Signale zu einem neuen Signal verknüpft)

Wie funktioniert die FFT ?

Behauptung: Wir können $C^{(n)}x$ in Zeit $O(n\log(n))$ berechnen und damit die (inverse) DFT auch in Zeit $O(n\log(n))$ berechnen.

Untersuchen wir dazu die Matrix $C^{(n)}$ ...

Wie funktioniert die FFT ?

$$C^{(n)} = \begin{bmatrix} c_{0,0}^{(n)} & c_{0,1}^{(n)} & \cdots & c_{0,n-1}^{(n)} \\ c_{1,0}^{(n)} & c_{1,1}^{(n)} & \cdots & c_{1,n-1}^{(n)} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n-1,0}^{(n)} & c_{n-1,1}^{(n)} & \cdots & c_{n-1,n-1}^{(n)} \end{bmatrix}$$

mit

$$c_{i,j}^{(n)} = e^{2\pi\sqrt{-1}\frac{ij}{n}}$$

Sei n gerade und $j < \frac{n}{2}$, dann gilt:

$$\begin{aligned} c_{i,j+\frac{n}{2}}^{(n)} &= c_{i,j}^{(n)}\, c_{i,\frac{n}{2}}^{(n)} \\ &= c_{i,j}^{(n)}\, e^{2\pi\sqrt{-1}\frac{i}{2}} \end{aligned}$$

Wie funktioniert die FFT ?

$$e^{2\pi\sqrt{-1}\frac{i}{2}} = \begin{cases} 1 & \text{falls } i \text{ gerade ist} \\ -1 & \text{falls } i \text{ ungerade ist} \end{cases}$$

$$\Rightarrow c^{(n)}_{i,j+\frac{n}{2}} = \begin{cases} c^{(n)}_{i,j} & \text{falls } i \text{ gerade ist} \\ -c^{(n)}_{i,j} & \text{falls } i \text{ ungerade ist} \end{cases}$$

Wie funktioniert die FFT ?

Es folgt . . .

$$
\begin{aligned}
y &= C^{(n)} x \\
\Leftrightarrow \quad y_i &= \sum_{j=0}^{n-1} c_{i,j}^{(n)} x_j \\
&= \sum_{j=0}^{\frac{n}{2}-1} c_{i,j}^{(n)} x_j + c_{i,j+\frac{n}{2}}^{(n)} x_{j+\frac{n}{2}} \\
&= \begin{cases} \sum_{j=0}^{\frac{n}{2}-1} c_{i,j}^{(n)} \left(x_j + x_{j+\frac{n}{2}} \right) & \text{falls } i \text{ gerade ist} \\ \sum_{j=0}^{\frac{n}{2}-1} c_{i,j}^{(n)} \left(x_j - x_{j+\frac{n}{2}} \right) & \text{falls } i \text{ ungerade ist} \end{cases}
\end{aligned}
$$

. . .

Wie funktioniert die FFT ?

Die Matrix $C^{(n)}$ kann damit faktorisiert werden:

$$C^{(n)} = P^{(n)} \begin{bmatrix} ? & 0 \\ 0 & ? \end{bmatrix} \begin{bmatrix} I^{(\frac{n}{2})} & I^{(\frac{n}{2})} \\ I^{(\frac{n}{2})} & -I^{(\frac{n}{2})} \end{bmatrix}$$

mit

$$P^{(n)} = \left[p_{i,j}^{(n)} \right]_{i,j \in \{0,1,\ldots,(n-1)\}} \quad \text{(Permutation)}$$

$$p_{i,j}^{(n)} = \begin{cases} 1 & \text{falls } i = ((2j) \bmod n) + \left\lfloor \frac{2j}{n} \right\rfloor \\ 0 & \text{sonst} \end{cases}$$

Wie funktioniert die FFT ?

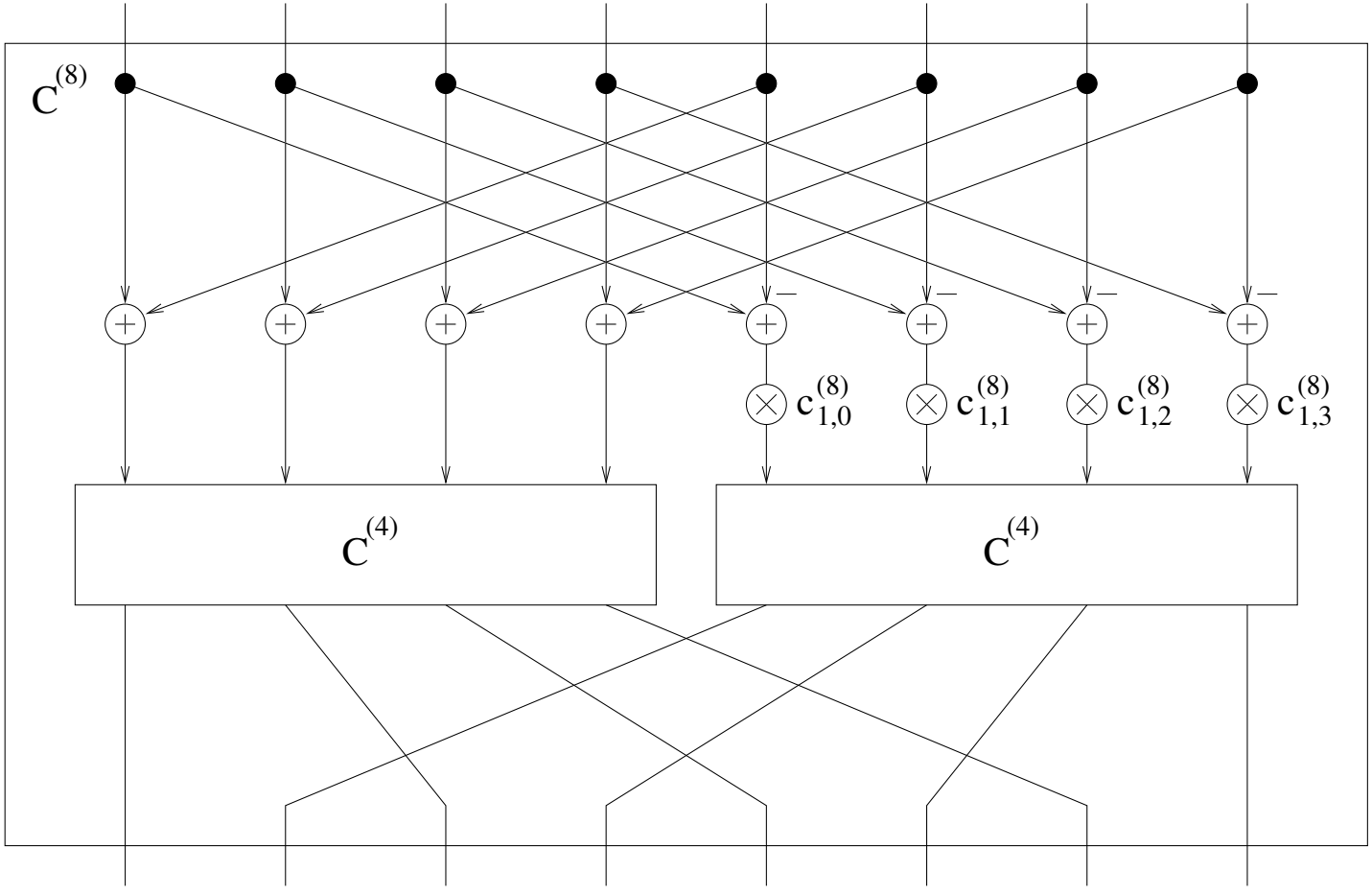
Die Matrix $C^{(n)}$ kann folgendermaßen faktorisiert werden:

$$C^{(n)} = P^{(n)} \begin{bmatrix} C^{(\frac{n}{2})} & 0 \\ 0 & C^{(\frac{n}{2})} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & R^{(\frac{n}{2})} \end{bmatrix} \begin{bmatrix} I^{(\frac{n}{2})} & I^{(\frac{n}{2})} \\ I^{(\frac{n}{2})} & -I^{(\frac{n}{2})} \end{bmatrix}$$

mit

$$R^{(\frac{n}{2})} = \begin{bmatrix} c_{1,0}^{(n)} & 0 & \cdots & 0 \\ 0 & c_{1,1}^{(n)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_{1,\frac{n}{2}-1}^{(n)} \end{bmatrix}$$

Beispiel $C^{(8)}$ als Blockdiagramm



FFT, ein Divide & Conquer Algo

- $2n$-Punkt DFT in $O(n)$ auf zwei n-Punkt DFTs reduzierbar
- Es gibt $\log_2(n)$ Rekursions-Stufen, falls n eine Zweierpotenz ist
- In jeder Stufe werden insgesamt n Operationen benötigt

$\Rightarrow$ Der FFT-Algorithmus benötigt $O(n\log(n))$ Zeit