Beispiel 5.11. Gegeben sei die Dreiecksfunktion

$$f(x) = \begin{cases} x, & 0 \le x < \frac{1}{2}, \\ 1-x, & \frac{1}{2} \le x < 1, \\ 0, & \text{sonst.} \end{cases}$$

Mit Hilfe einer Interpolationsapproximation mit $N = 4$ erhält man für p die Situation aus Abbildung 14.

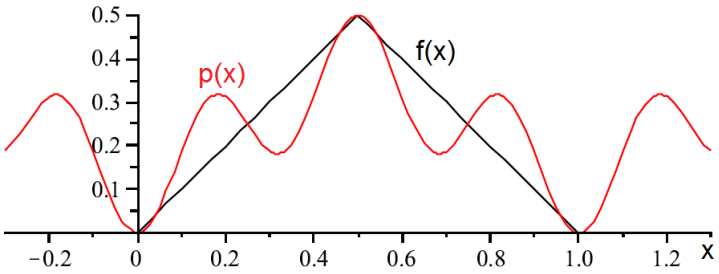

Abbildung 14: Interpolationsapproximation der Dreiecksfunktion durch das Polynom $p(x)$ aus Gl. (29) mit $L = 1$ und $N = 4$.

An diesem Beispiel ist sofort zu erkennen, dass p zwar an den Stützstellen exakt stimmt, die restliche Funktion aber nur sehr schlecht approximiert wird. Wählt man weitere Stützstellen um die Qualität der Approximation zu verbessern, oszilliert die Funktion p allerdings noch stärker. Abbildung 15 zeigt die Interpolationsapproximation der Dreiecksfunktion mit $N = 16$, die klarerweise noch weniger brauchbar ist als die mit $N = 4$. Wir sehen also, dass der erste Versuch mit Gl. (29) nicht wirklich geeignet ist.

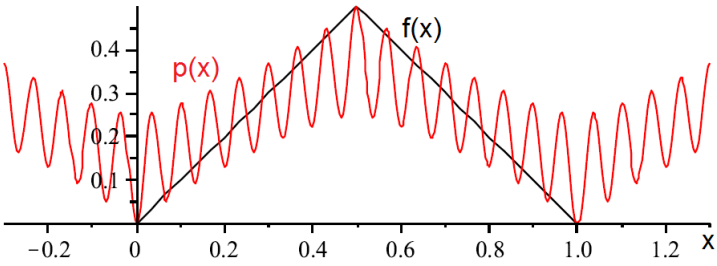

Abbildung 15: Interpolationsapproximation der Dreiecksfunktion mit $N = 16$. Die Approximation oszilliert dabei noch stärker als für $N = 4$.

Beispiel 5.11 zeigt, dass man andere Polynome benötigt. Dazu betrachten wir den

folgenden (in gewisser Weise symmetrisierten) Ansatz für N gerade:

$$\begin{aligned} r(x) &:= \frac{1}{\sqrt{N}} \sum_{k=-N/2}^{N/2-1} d_{k+N/2}\, \mathrm{e}^{2\pi \mathrm{i} kx/L} \\ &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} d_k\, \mathrm{e}^{2\pi \mathrm{i}(k-N/2)x/L} \\ &= \underbrace{\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} d_k\, \mathrm{e}^{2\pi \mathrm{i} kx/L}}_{\text{entspricht } p(x)}\, \mathrm{e}^{-\pi \mathrm{i} Nx/L}. \end{aligned} \tag{30}$$

Zunächst ergeben sich die Parameter wie folgt.

Satz 5.12. *Sei $N \in \mathbb{N}$ beliebig, aber fest. Zu äquidistanten Stützstellen $x_j \in [0, L]$ und beliebigen Stützwerten $f_j \in \mathbb{C}$ für $0 \le j \le N-1$ erfüllt das trigonometrische Polynom r aus Gl. (30) die Interpolationsgleichungen $r(x_j) = f_j$ genau dann, wenn gilt:*

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{N-1} \end{pmatrix} = M_N \begin{pmatrix} f_0 \\ -f_1 \\ f_2 \\ \vdots \\ (-1)^{N-1} f_{N-1} \end{pmatrix}.$$

Beweis. Das Resultat folgt wieder aus dem Eindeutigkeitssatz der DFT (Satz 5.6), wenn man noch beachtet, dass $\mathrm{e}^{k\pi \mathrm{i}} = (-1)^k$ für $k \in \mathbb{Z}$ gilt. □

Für das obige Beispiel ergeben sich nun deutlich bessere Approximationen an die Dreiecksfunktion, wie Abbildung 16 illustriert.

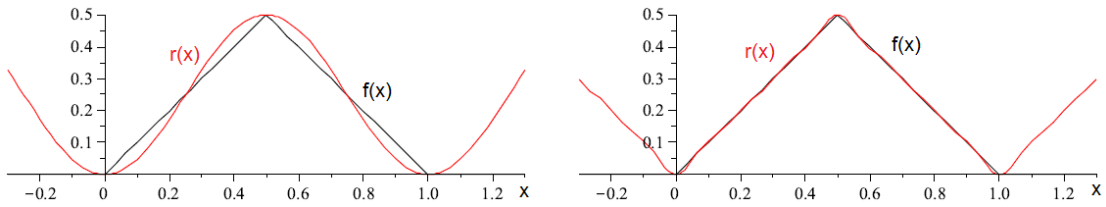

Abbildung 16: Beispiel der Interpolationsapproximation der Dreiecksfunktion durch das verbesserte Polynom r für $N = 4$ (links) und $N = 16$ (rechts). Die Approximation ist deutlich besser als durch das vorherige Polynom p.

Die obige Diskussion soll auch klarmachen, dass man bei Anwendungen der DFT Vorsicht walten lassen muss.

5.2 Schnelle Fourier-Transformation

Dieses Kapitel beschäftigt sich mit der effizienten Berechnung der DFT für einen gegebenen Datensatz $(f_0, f_1, \ldots, f_{N-1})$. Der Algorithmus der schnellen Fourier-Transformation (kurz: FFT, für 'fast Fourier transform') wird auch in vielen Computer-Programmen zur Fourier-Transformation verwendet. Das Problem der DFT, wie sie bisher dargestellt wurde, ist die Matrix-Multiplikation, welche sehr rechenintensiv ist. Der Aufwand liegt dabei in $\mathcal{O}(N^2)$. Mit einem einfachen Trick lässt sich dies besser gestalten.

Für $N = 2M$ und eine Umsortierung der Summanden der k-ten Komponente $\mathcal{F}_k$ erhält man zunächst

$$\begin{aligned}
&\mathcal{F}_k[g_0, g_M, g_1, g_{M+1}, \ldots, g_{M-1}, g_{2M-1}] \\
&\quad = \frac{1}{\sqrt{2M}}\Bigg(\sum_{j=0}^{M-1} g_j \, \mathrm{e}^{-2\pi \mathrm{i}\, k(2j)/2M} + \sum_{j=0}^{M-1} g_{M+j} \, \mathrm{e}^{-2\pi \mathrm{i}\, k(2j+1)/2M}\Bigg) \\
&\quad = \frac{1}{\sqrt{2M}}\Bigg(\sum_{j=0}^{M-1} g_j \, \mathrm{e}^{-2\pi \mathrm{i}\, kj/M} + \mathrm{e}^{-\pi \mathrm{i}\, k/M} \sum_{j=0}^{M-1} g_{M+j} \, \mathrm{e}^{-2\pi \mathrm{i}\, kj/M}\Bigg),
\end{aligned}$$

und man erkennt, dass auf der rechten Seite Ausdrücke stehen, die wir aus der DFT eines Datensatzes der Länge M kennen. Allgemeiner gilt nun folgender Zusammenhang.

Satz 5.13. *Sei $M \in \mathbb{N}$ beliebeig, aber fest und $N = 2M$. Aus der DFT der beiden Datensätze $(g_0, g_1, ..., g_{M-1})$ und $(g_M, g_{M+1}, ..., g_{2M-1})$ der Länge M bekommt man die DFT des Datensatzes $(g_0, g_M, g_1, g_{M+1}, ..., g_{M-1}, g_{2M-1})$ der Länge $2M$ wie folgt:*

$$\begin{aligned}
\mathcal{F}_k[g_0, g_M, \ldots, g_{2M-1}] &= \tfrac{1}{\sqrt{2}}\big(\mathcal{F}_k[g_0, g_1, \ldots, g_{M-1}] + \mathrm{e}^{-\pi \mathrm{i}\, k/M} \mathcal{F}_k[g_M, g_{M+1}, \ldots, g_{2M-1}]\big), \\
\mathcal{F}_{k+M}[g_0, g_M, \ldots, g_{2M-1}] &= \tfrac{1}{\sqrt{2}}\big(\mathcal{F}_k[g_0, g_1, \ldots, g_{M-1}] - \mathrm{e}^{-\pi \mathrm{i}\, k/M} \mathcal{F}_k[g_M, g_{M+1}, \ldots, g_{2M-1}]\big),
\end{aligned}$$

jeweils mit $0 \le k < M$.

Beweis. Die erste Relation folgt sofort aus obiger Rechnung. Die zweite Relation rechnet man analog nach, wobei man noch folgende Identität beachtet:

$$\begin{aligned}
\mathrm{e}^{-2\pi \mathrm{i}\,(k+M)\ell/2M} &= \mathrm{e}^{-\pi \mathrm{i}\, \ell} \, \mathrm{e}^{-\pi \mathrm{i}\,(k\ell)/M} = (-1)^{\ell} \mathrm{e}^{-\pi \mathrm{i}\,(k\ell)/M} \\
&= \begin{cases} \mathrm{e}^{-2\pi \mathrm{i}\,(kj)/M}, & \ell = 2j, \\ -\mathrm{e}^{-\pi \mathrm{i}\, k/M} \, \mathrm{e}^{-2\pi \mathrm{i}\,(k\ell)/M}, & \ell = 2j+1. \end{cases}
\end{aligned}$$

Damit ist die behauptete Reduktion gezeigt. □

Mit Hilfe von Satz 5.13 lässt sich der Aufwand zur Berechnung der DFT nun deutlich reduzieren. Allerdings ist dies mittels Satz 5.13 zunächst nur für Datensätze mit einer bestimmten Länge von 2^n hilfreich. Die Verbesserung ist dafür aber bemerkenswert, denn die Berechnung kann nun in $\mathcal{O}(n \log(n))$ ablaufen. Für andere Primzahlen gibt es analoge Reduktionen, so dass man am Ende für jeden endlichen Datensatz einen entsprechenden FFT-Algorithmus bekommt, indem man die Primfaktorzerlegung der Länge heranzieht.

Bemerkenswert ist hier, dass dieses Verfahren – in voller Allgemeinheit – bereits Gauß bekannt war. Es findet sich in seinen Notizbüchern, und zwar zu einem Datum, das *vor* der Veröffentlichung der Fourier-Reihen (durch Fourier) liegt.

Das Problem des Umfangs des Datensatzes ist auch in der Praxis oft gar nicht hinderlich, da die Datensätze in der Regel manuell gewählt werden und so die Anzahl im Vorfeld sinnvoll bestimmt werden kann. Abbildung 17 zeigt ein Beispiel für die komplette Behandlung eines Datensatzes der Länge 8. Man sieht, dass man lediglich den anfänglichen Datensatz einmalig umsortieren muss, und dann in drei Schritten am Ziel ist (da $f_j = \mathcal{F}[f_j]$ für Datensätze der Länge 1) — entsprechend der Relation $8 = 2^3$. Die Umordnung nimmt man über eine binäre Darstellung des Index mit nachfolgender Bitumkehr vor (Details: Übung). Weiterführende Literatur ist [6].

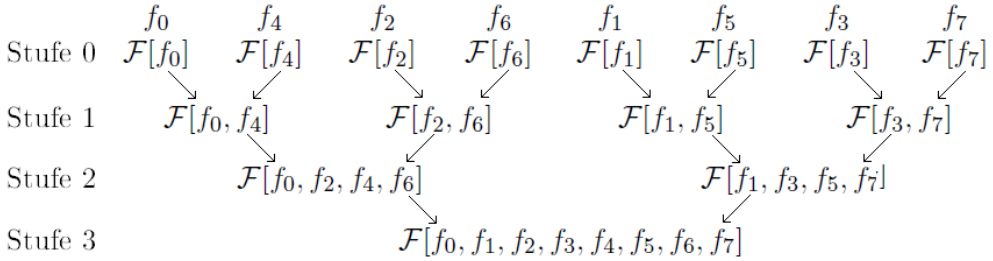

Abbildung 17: Beispiel für die FFT eines Datensatzes der Länge 8 mit einmaliger Umsortierung des ursprüglichen Datensatzes.

5.3 Beispiel: Bildkompression

Mit Hilfe der DFT bzw. der FFT ist die effiziente Kompression von Bildern möglich. Um das Prinzip näher zu erläutern, soll ein Bild zunächst nur aus Grauwerten bestehen. Die Idee ist nun, jede Bildzeile getrennt zu betrachten, und den Grauwert eines einzelnen Pixels k einer Zeile als Funktionswert f_k zu interpretieren. Daraus entsteht eine diskrete Funktion f, auf welche die DFT angewendet werden kann.

Um nun eine tatsächliche Kompression zu erhalten, muss abgeschätzt werden, auf welchem Anteil der so erhaltenen Transformation die 'wichtige' Informationen für das ursprüngliche Bild liegen. Z.B. kann man nur die ersten 20% speichern unter der Annahme, dass die letzten 80% vernachlässigbar sind. Dadurch benötigt man dann nur etwa 20% an Speicherbedarf der ursprünglichen Pixel-Werte f_k und kann durch die Inverse der DFT annähernd das vollständige Bild wiederherstellen. Dies ist eine sehr rudimentäre Herangehensweise, welche noch stark verbessert werden kann. Eine ausführliche Erklärung optimierter Verfahren findet sich in [9, Kap. 13].