4.2 Fourier-Transformation diskreter Signale

Wir nähern uns langsam der Fourier-Analyse von diskreten Signalen. Schließlich wollen wir etwaige spektrale Analysen und dergleichen im Digitalen durchführen, um die Vorzüge von digitalen Rechenwerken dabei nutzen zu können. Die vorher eingeführten Transformationen sind zwar hilfreich für theoretische Argumentation, wie beispielsweise beim Sampling-Theorem 2.1. Deshalb wenden wir uns nun der spektralen Analyse von diskreten Signalen zu. Wir werden aber im Verlauf auch wieder ähnliche Zusammenhänge wie in (4.1.6) finden.

4.2.1 Fourier-Transformation diskreter periodischer Signale

Wir beginnen mit diskreten Signalen, die gleichzeitig periodisch sind, also ein N existiert, sodass

$$x[n] = x[n+N] \quad \text{für alle} \quad n \in \mathbb{Z}$$

gilt. Aus vorherigen Diskussionen in Abschnitt 2 wissen wir einerseits, dass diskrete Signale ein periodisches Spektrum auf $(0,1)$ besitzen. Andererseits wissen wir aus Abschnitt 4.1.1, dass periodische Signale ein *diskretes* Spektrum besitzen. Wir finden also nun intuitiv, dass das Spektrum von diskreten periodischen Signalen *ebenfalls* diskret und periodisch ist. Denken wir zurück an Abschnitt 2.3 so haben wir bereits alles Notwendige betrachtet. Ein N-periodisches diskretes Signal ergibt sich aus der Linearkombination der diskreten Signale $x_k[\cdot] : \mathbb{Z} \to \mathbb{C}$ definiert durch

$$x_k[n] = \exp\left(\jmath 2\pi \frac{k}{N} n\right) \quad \text{mit} \quad k = 0, \dots, N-1.$$

Das heißt, für $x[\cdot]$ setzen wir mit

$$x[\cdot] = \sum_{k=0}^{N-1} c[k] x_k[\cdot]$$

an. Damit sind bereits beide Eigenschaften des Spektrums „eingepreist“. Denn man erkennt in den $x_k[\cdot]$ die vorher erwähnte *zweifache* Periodizität wieder, weil sowohl $x_{k+N}[\cdot] = x_k[\cdot]$ für alle k als auch $x_k[n+N] = x_k[n]$ für alle n gilt.

Es ist nun unser Ziel für gegebene Werte $x[n]$ von $x[\cdot]$ die Werte des *ebenfalls periodischen und diskreten* Signals $c[\cdot]$ zu bestimmen. Dies verläuft ganz analog zu Abschnitt 4.1.1. Wir definieren das Skalarprodukt $\langle \cdot, \cdot \rangle$ für N-periodische und diskrete Signale via

$$\langle x_1[\cdot], x_2[\cdot] \rangle = \sum_{n=0}^{N-1} x_1[n] x_2[n]^*.$$

Das heißt, dass wir periodisches Signal $x[\cdot]$ mit dem *endlich-dimensionalen* Vektor $\boldsymbol{x} \in \mathbb{C}^N$ identifizieren, wir setzen also die Einträge des Vektors als $\boldsymbol{x}_i = x[i-1]$. Dann können wir auch für das entsprechende Skalarprodukt der Vektoren $\boldsymbol{x}_{1,2} \in \mathbb{C}^N$

$$\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle = (\boldsymbol{x}_2^*)^{\mathrm{T}} \cdot \boldsymbol{x}_1 = \boldsymbol{x}_2^{\mathrm{H}} \cdot \boldsymbol{x}_1$$

schreiben. Analog identifizieren wir die periodische und diskrete Sequenz $c[\cdot]$ mit dem Vektor $\boldsymbol{c} \in \mathbb{C}^N$.

Wie in Abschnitt 4.1.1 müssen wir nur

$$\langle \boldsymbol{x}_k, \boldsymbol{x}_\ell \rangle = \sum_{i=0}^{N-1} x_k[i] x_\ell[i]^* = \sum_{i=0}^{N-1} \exp\left(\jmath 2\pi \frac{i(k-l)}{N}\right) = \begin{cases} N & \text{falls} \quad k = \ell \\ \frac{1-\exp\left(\jmath 2\pi \frac{(k-l)}{N}\right)^N}{1-\exp\left(\jmath 2\pi \frac{(k-l)}{N}\right)} & \text{sonst} \end{cases} = \begin{cases} N & \text{falls} \quad k = \ell \\ 0 & \text{sonst} \end{cases}$$

berechnen. Hierbei nutzen wir, dass $\exp(\jmath 2\pi k) = 1$ für alle $k \in \mathbb{Z}$. Wie in Abschnitt 4.1.1 finden wir, dass also gilt $\langle \boldsymbol{x}_k, \boldsymbol{x}_\ell \rangle = 0$, falls $k \neq \ell$ und $\langle \boldsymbol{x}_k, \boldsymbol{x}_k \rangle = N$. Damit ergibt sich bei Anwendung auf das eigentliche Signal $\boldsymbol{x}$ für den Vektor $\boldsymbol{c}$, dass

$$\langle \boldsymbol{x}, \boldsymbol{x}_\ell \rangle = \left\langle \sum_{k=1}^{N} \boldsymbol{c}_k \boldsymbol{x}_k, \boldsymbol{x}_\ell \right\rangle = \sum_{k=1}^{N} \boldsymbol{c}_k \langle \boldsymbol{x}_k, \boldsymbol{x}_\ell \rangle = N\boldsymbol{c}_\ell \Rightarrow \boldsymbol{c}_\ell = \frac{1}{N} \langle \boldsymbol{x}, \boldsymbol{x}_\ell \rangle.$$

Zusammenfassend, kann man also sagen, dass sich folgende Analyse- und Synthesegleichungen ergeben:

$$x[n] = \sum_{k=0}^{N-1} c[k] \exp(\jmath 2\pi kn/N), \quad \text{und} \quad c[k] = \frac{1}{N} \sum_{n=0}^{N-1} x[n] \exp(-\jmath 2\pi kn/N). \tag{4.2.1}$$

In Vektorschreibweise lässt sich der erste Teil durch

$$\boldsymbol{x} = \sum_{k=0}^{N-1} \langle \boldsymbol{x}, \boldsymbol{x}_k \rangle \boldsymbol{x}_k \tag{4.2.2}$$

ausdrücken.

Weiterhin, können wir eine Matrix $\boldsymbol{F}_N \in \mathbb{C}^{N \times N}$ definieren, deren k-te Spalte den Vektor $\boldsymbol{x}_k$ beinhaltet. Wir definieren also

$$\boldsymbol{F}_N = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \ldots, \boldsymbol{x}_N]$$

Dann können wir noch einen Schritt weitergehen und sehen, dass

$$\boldsymbol{c} = \frac{1}{N} \boldsymbol{F}_N^{\mathrm{H}} \boldsymbol{x}, \quad \text{und} \quad \boldsymbol{x} = \boldsymbol{F}_N \boldsymbol{c}$$

gilt. Das heißt, dass sich Fourier-Analyse und Fourier-Synthere von diskreten und periodischen Signalen durch eine Matrix-Vektor-Multiplikation durchführen lässt! Das sind erst einmal gute Nachrichten, denn damit wissen wir, dass digitale Rechenwerke sehr gut darin sind, diese Transformation durchzuführen. Die hier vorgestellte Transformation nennen wir Discrete Fourier Transform (DFT).

In Codeschnipsel 18 zeigen wir ein einfaches Beispiel für $x[\cdot] = u[\cdot] - u[\cdot - k]$. Außerdem zeigen wir auch die beiden Berechungsmethoden der $c[\cdot]$ – einerseits mit der Definition und andererseits über ein Matrix-Vektor-Produkt.

4.2.2 Leichtungsdichte-Spektrum diskreter periodischer Signale

Da periodische Signale keine endliche Energie besitzen, betrachten wir die durchschnittliche Leistung über eine Periode. Wir betrachten also für ein N-periodisches Signal $x[\cdot]$ das Funktional

$$P(x[\cdot]) = \frac{1}{N} \sum_{i=0}^{N-1} |x[n]|^2. \tag{4.2.3}$$

Dann können wir ganz analog zu vorher wieder herleiten, dass für

$$x[\cdot] = \sum_{i=0}^{N-1} c[k] x_k[\cdot]$$

und dessen Leistung gilt, dass

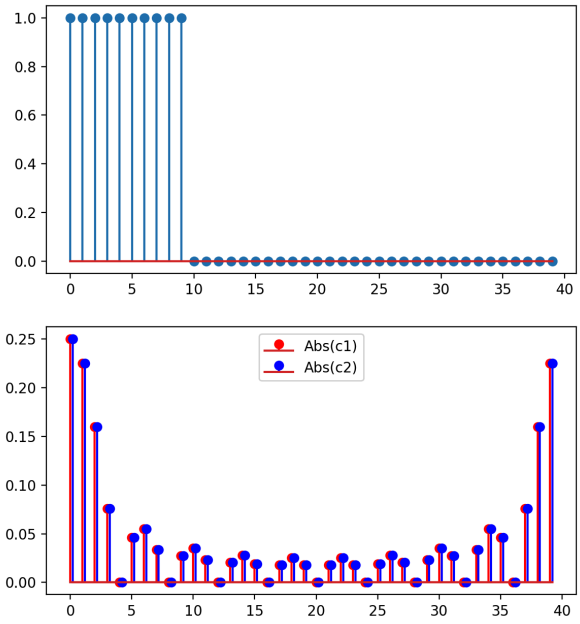
$$P(x[\cdot]) = \sum_{i=0}^{N-1} |c[k]|^2.$$

Also auch hier finden wir wieder, dass sich die durchschnittliche Leistung einer Periode im Zeitbereich auf die Leistung einer Periode im Frequenzbereich überträgt. Auch hier können wir die Folge $|c[\cdot]|^2$ als Leistungsdichte-Spektrum interpretieren.

```python
def x_k(k: int, N: int) -> np.ndarray[complex]:
    return np.exp(
        1j * 2 * np.pi * k * np.arange(N) / N
    )


N = 40
x = np.zeros(N)
x[: N // 4] += np.ones(N // 4)

c1 = np.array(
    [
        (x * x_k(kk, N).conj()).sum() / N
        for kk in range(N)
    ]
)
F = np.array([x_k(kk, N) for kk in range(N)]).T
c2 = F.conj().T.dot(x) / N
```



Codeschnipsel 18: Berechnung und Darstellung von (4.2.1), siehe `dsv/code/dft_1.py`

4.2.3 Fourier-Transformation diskreter aperiodischer Signale

Wiederum analog zu Abschnitt 4.1 benötigen wir noch eine Transformation für diskrete Signale, die keine Periodizität aufweisen. Hierzu definieren wir die zugehörige Fourier-Transformation, die wir DTFT nennen, durch

$$X(f) = \sum_{n\in\mathbb{Z}} x[n] \exp(-\jmath 2\pi f n), \tag{4.2.4}$$

genau wie in Abschnitt 2 bei der Herleitung von (2.1). Im Unterschied zur „analogen“ Fourier-Transformation (4.1.4) sehen wir, dass X nur für Frequenzen $f \in (0, 1]$ definiert ist, da das diskrete Signal $x_f[n] = \exp(\jmath 2\pi f n)$ periodisch in f ist. Es gilt also $x_{f+k}[\cdot] = x_f[\cdot]$ für alle $k \in \mathbb{Z}$. Dies „passt“ auch zur Natur von diskreten Signalen, da deren Frequenzbereich *immer* periodisch sein muss.

Wir können nun für die inverse Transformation einmal (4.2.4) mit (4.1.1) vergleichen. Bis auf das Vorzeichen in der Funktion $\exp()$ gleicht (4.2.4) einer Fourier-Reihe der periodischen Funktion X. In der Tat, können wir die Folge $x[\cdot]$ als Fourier-Koeffizienten der Funktion X durch

$$x[n] = \int_{-1/2}^{+1/2} X(f) \exp(\jmath 2\pi f n)\, \mathrm{d}f \tag{4.2.5}$$

wiederfinden. Diese Synthese-Operation nennen wir dann Inverse Discrete Time Fourier Transform (iDTFT). Es ist wiederum anzumerken, dass die DTFT „nur“ ein theoretisches Werkzeug ist, da sich im Allgemeinen die unendliche Summe in (4.2.4) praktisch nicht realisieren lässt, genauso wenig wie deren Resultat, eine kontinuierliche Funktion.

In Codeschnipsel 19 zeigen wir dennoch, wie man die DTFT einer aperiodischen Folge approximieren

kann. Wir studieren hierzu das Signal

$$x[n] = \begin{cases} \frac{\omega}{\pi} & \text{für} \quad n = 0, \\ \frac{\omega}{\pi} \frac{\sin(\omega n)}{\omega n} & \text{sonst} \end{cases} .$$

Dessen analytisch bestimmte DTFT X ist

$$X(f) = \text{rect}(f/(2\pi\omega)),$$

welche wir durch eine endliche Summe über die von uns verfügbaren Werte von $x[\cdot]$ bestimmen.

Nun treten hierbei mehrere Effekte zutage.

- Die Approximation der DTFT mittels des „abgeschnittenen“ Signals x stimmt noch nicht sehr gut mit der analytischen Lösung überein. Verändert man den Wert der Variable N im Skript, tritt dieser Effekt stärker oder schwächer zutage.
- Auch bei Erhöhung von N stellt sich immernoch keine Konvergenz von X_{approx} gegen X_{true} ein. Dies liegt daran, dass die DTFT von $x[\cdot]$ nicht konvergiert, wie wir in Codeschnipsel 15 bereits gesehen hatten. Diesen Effekt bezeichnet man allgemein als Gibbssches Phänomen[11].
- Augenscheinlich ist es besser die iDTFT aus X_{approx} zu berechnen, als aus X_{true}. Doch dies liegt legidlich daran, dass die numerische Integration der rect-Funktion nicht genau genug ist. Erhöht man die Anzahl der Punkte im Array `F`, dann stimmen im dritten Plot die drei Graphen besser überein.

[11] https://en.wikipedia.org/wiki/Gibbs_phenomenon

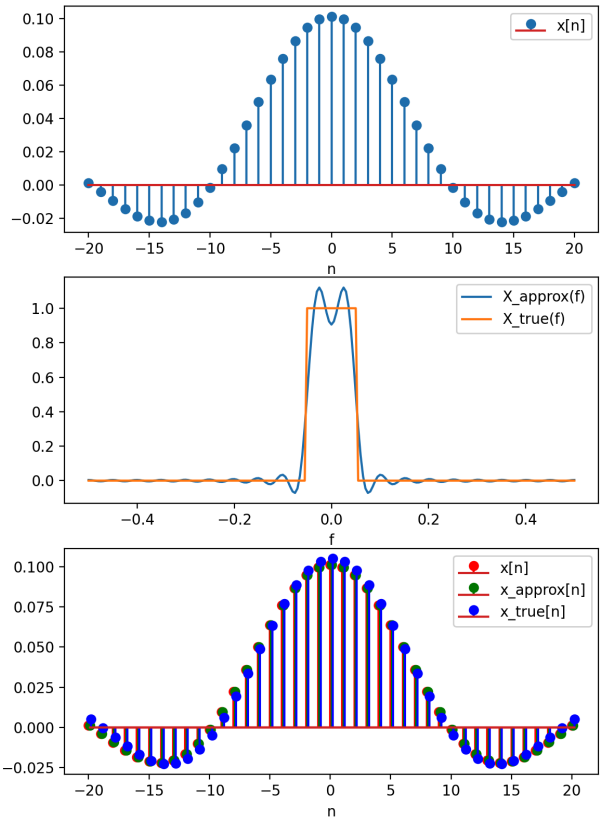
```python
def dtft(
    x: np.ndarray[complex],
    n: np.ndarray[int],
    F: np.ndarray[float],
) -> np.ndarray[complex]:
    return np.array(
        [
            (
                x * np.exp(-2j * np.pi * n * f)
            ).sum()
            for f in F
        ]
    )


def idtft(
    X: np.ndarray[complex],
    F: np.ndarray[float],
    n: np.ndarray[int],
):
    return np.array(
        [
            (F[1] - F[0])
            * (
                X * np.exp(2j * np.pi * nn * F)
            ).sum()
            for nn in n
        ]
    )


N = 20
n = np.linspace(-N, +N, 2 * N + 1)
omega = 1 / np.pi

x = np.sin(omega * n) / (np.pi * n)
x[N] = omega / np.pi
F = np.linspace(-0.5, +0.5, 201)
X_approx = dtft(x, n, F)
X_true = np.zeros_like(F)
X_true[np.abs(F) <= omega / (2 * np.pi)] += 1

x_approx = idtft(X_approx, F, n)
x_true = idtft(X_true, F, n)
```



Codeschnipsel 19: Berechnung und Darstellung von (4.2.4), siehe dsv/code/dtft.py

4.2.4 Leichtungsdichte-Spektrum diskreter aperiodischer Signale

Die Energie eines diskreten Signals haben wir in (3.1.1) durch

$$\mathcal{E}(x[\cdot]) = \sum_{n\in\mathbb{Z}} |x[n]|^2$$

definiert. Wiederum können wir uns überlegen, dass dann für die DTFT X gilt, dass

$$\mathcal{E}(x[\cdot]) = \int_{-1/2}^{+1/2} |X(f)|^2 \,\mathrm{d}f.$$

Schlussendlich bezeichnet man dann

$$S_{xx}(f) = |X(f)|^2$$

als das Leichtungsdichte-Spektrum des Signals $x[\cdot]$. In manchen Anwendungen ist es auch wieder sinnvoll Betrag und Phase von $X(f)$ zu analysieren.