2.6 Diskrete Fourier-Transformation

Neben Polynomen und rationalen Funktionen können auch trigonometrische Funktionen zur Approximation von Funktionen genutzt werden. Dies bietet sich besonders bei periodischen Funktionen an. → Fourier-Reihe.
Die Fourier-Transformation spielt an sich schon eine große Rolle in der Physik:

- Quantenmechanik
- Quantenfeldtheorie
- Bildbearbeitung

Deswegen werden wir uns mit der diskreten Fourier-Transformation (DFT) beschäftigen.
Unter sehr allgemeinen Bedingungen lässt sich im Intervall $[0, 2\pi[$ die periodische Funktion $f(z)$, $z \in [0, 2\pi[$ durch eine Fourier-Reihe darstellen:

$$f(z) = \frac{1}{\sqrt{2\pi}} \sum_{k=-\infty}^{\infty} g_k e^{ikz} \qquad g_k = \frac{1}{2\pi} \int_0^{2\pi} e^{-ikz} f(z)\, dz$$

Wir diskretisieren diese, indem wir Stützstellen $z_j = \frac{2\pi}{n} j$ und Stützwerte f_j, $j = 0, 1, \dots, n-1$ einführen.
Die Periodizität ist dann

$$f(z+2\pi) = f(z) \longrightarrow f_{j+n} = f_j$$

und die ebenen Wellen werden zu

$$f_k = \frac{1}{\sqrt{2\pi}} e^{ikz} \longrightarrow f_{k,j} = \frac{1}{\sqrt{n}} e^{ikz_j} = \frac{1}{\sqrt{n}} e^{i\frac{2\pi}{n}kj}$$

Die DFT hat dann folgende Form:

$$f_j = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} g_k e^{i \frac{2\pi}{n} k \cdot j} \qquad g_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} e^{-i \frac{2\pi}{n} kj} f_j$$

Fasst man die f_j und g_k als Vektoren auf, so kann man

$$W_{kj} = \frac{1}{\sqrt{n}} \omega_n^{kj} \qquad , \omega_n := e^{-\frac{2\pi}{n} i}$$

benutzen und die DFT umschreiben

$$g = W \cdot f = W_{kj} f^j \qquad , f = W^+ g$$

Die Matrix W hat n^2 Einträge, aber nur n verschiedene Einträge $\omega^0, \dots, \omega^{n-1}$. Im Prinzip braucht man dennoch n^2 komplexe Additionen und Multiplikationen um z.B. $g = Wf$ zu berechnen. Der (Rechen-) Aufwand kann erheblich reduziert werden, wenn die schnelle Fourier-Transformation (FFT) benutzt wird. Im folgenden für $n = 2^k$, aber im Prinzip für beliebige n möglich.

<u>Beispiel</u>: $n = 4$

$$\begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 \\ 1 & \omega^2 & 1 & \omega^2 \\ 1 & \omega^3 & \omega^2 & \omega^1 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{pmatrix} \Rightarrow \begin{pmatrix} g_0 \\ g_2 \\ g_1 \\ g_3 \end{pmatrix} = \left(\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & \omega^2 & 1 & \omega^2 \\ \hline 1 & \omega^1 & \omega^2 & \omega^3 \\ 1 & \omega^3 & \omega^2 & \omega \end{array}\right) \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{pmatrix}$$

vertausche Zeile 2 und 3

$$= \left(\begin{array}{cc|cc} 1 & 1 & \multicolumn{2}{c}{0} \\ 1 & \omega^2 & & \\ \hline \multicolumn{2}{c|}{0} & 1 & 1 \\ & & 1 & \omega^2 \end{array}\right) \left(\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 1 & 0 & \omega^2 & 0 \\ 0 & \omega^1 & 0 & \omega^3 \end{array}\right) \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \end{pmatrix}$$

mit
$$z_0 = f_0 + f_2$$
$$z_1 = f_1 + f_3$$
$$z_2 = \omega^0 (f_0 - f_2)$$
$$z_3 = \omega^1 (f_1 - f_3)$$

$$= \begin{pmatrix} 1 & 1 & \multicolumn{2}{c}{0} \\ 1 & \omega^2 & & \\ \multicolumn{2}{c}{0} & 1 & 1 \\ & & 1 & \omega^2 \end{pmatrix} \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix}$$

da $\omega^2 = -\omega^0 = -1$ und $\omega^3 = -\omega^1$

$$= \begin{pmatrix} z_0 + z_1 \\ z_2 + z_3 \\ z_0 + \omega^2 z_1 \\ z_2 + \omega^2 z_3 \end{pmatrix}$$

Wir haben also eine DFT 4. Ordnung in zwei 2fer Ordnung überführt.

<u>Allgemein:</u> Sei $n = 2m,\ m \in \mathbb{N}$

$$g_{2l} = \sum_{j=0}^{2m-1} f_j \omega_n^{2lj} = \sum_{j=0}^{m-1} (f_j + f_{j+m}) \omega_n^{2lj}$$

da $\omega_n^{2l(m+j)} = \omega_n^{2lj} \omega_n^{2lm} = \omega_n^{2jl}$

$$\omega_n^{2lm} = 1 = \exp\left(-\frac{2\pi}{n} i\, 2lm\right) = \exp\left(-\frac{2\pi}{n} i l n\right) = \exp(-2\pi i l)$$

$$= \sum_{j=0}^{m-1} (f_j + f_{j+m}) (\omega_n^2)^{lj}$$

Sei $z_j = f_j + f_{j+m} \Rightarrow g_{2l} = \sum_{j=0}^{m-1} z_j \omega_m^{jl}$ $\qquad \omega_n^2 = \omega_m$

$$g_{2l+1} = \sum_{j=0}^{2m-1} f_j \omega_n^{(2l+1)j} = \sum_{j=0}^{m-1} \left[f_j \omega^{(2l+1)j} + f_{j+m} \omega^{(2l+1)(j+m)} \right)$$

$$\omega_n^{(2l+1)(j+n)} = \omega_n^{(2l+1)m} = \omega_n^{(2l+1)j} \omega_n^{nl} \omega_n^{m} = -\omega_n^{(2l+1)j}$$

$$\Rightarrow g_{2l+1} = \sum_{j=0}^{m-1} \underbrace{(f_j - f_{j+m})}_{z_{j+m}} \omega_n^{(2l+1)j} = \sum_{j=0}^{m-1} z_{j+m} \omega_n^{(2l+1)j}$$

$$= \sum_{j=0}^{m-1} z_{j+m} \omega_n^{j} \omega_m^{jl}$$

außerdem: $z_{j+m} = z_j - 2f_{j+m}$

Beschleunigung der DFT durch Teilen des Systems der Größe n in zwei gleichgroße Systeme der Größe $n/2$. Diese Art der Algorithmen werden „divide and conquer" (D & C) Algorithmen genannt. Neben der FFT gibt es z.B. das Problem der „Türme von Hanoi" oder den Sortieralgorithmus „merge sort".

<u>Definition</u> Komplexität

Die DFT benötigt n^2 komplexe Additionen und Multiplikationen. Man sagt, dass sie in der Komplexitätsklasse $\mathcal{O}(n^2)$ liegt. Der FFT Algorithmus liegt in der Komplexitätsklasse $\mathcal{O}(n \log[n])$, wobei der Faktor $\log(n)$ von dem Vorgehen kommt, das Problem in zwei kleinere Unterprobleme zu zerlegen. Für große n verhält sich $n \log(n)$ <u>viel</u> besser als n^2
$\Rightarrow$ große Geschwindigkeitsvorteile in der Praxis

Der FFT-Algorithmus kann für beliebige n formuliert werden, die nicht Vielfache von 2 sind. Bibliothek (C/C++): FFTW

<u>Vorgehen:</u> (Beschränkung auf $n = 2^r$)

- Behandle z_j und z_{j+m} immer zusammen, da sie die gleichen Werte f_j und f_{j+m} benötigen.

- f_j und f_{j+m} werden nur für z_j und z_{j+m} benötigt
  $\Rightarrow$ können also im Speicher überschrieben werde

- nach dem Ende der FFT müssen wir die Werte $\{g_k\}$ umsortieren.

- Es ist sinnvoll alle Werte von ω^j vorher zu berechnen und im Speicher zu halten, um wiederholtes Berechnen zu vermeiden.

<u>Algorithmus:</u> FFT

Input: $n = 2$, $\{f_i\}$

$m = n/2$, $K = 1$

for $i = 0, \dots, r-1$

  for $k = 0, \dots, K$

    for $j = 0, \dots, m-1$

      $a = 2km + j$ $\qquad (j)$

      $b = a + m$ $\qquad (j+m)$

      $f_a = f_a + f_b$ $\qquad (z_j)$

      $f_b = \omega_n^{Kj} (f_a - 2 f_b)$ $\qquad (z_{j+m} \omega_n^j)$

  $m = m/2$

  $K = 2K$

output $g_i = f_i / \sqrt{n}$ (bis auf Reihenfolge)

3 Differentation und Integratoon

3.1 Numerische Differentation

Gegeben: $y = f(x)$ und o.B.d.A $n+1$ äquidistante Stützstellen $x_0, \ldots, x_n$. Wir suchen den Wert $f(\bar{x})$ an der Stelle $\bar{x}$.

<u>Vorgehen:</u> Ersetze $f(x)$ durch ein Interpolationspolinom und leite dieses ab.