Technische Universität München
Institut für Informatik
Prof. Dr. Hans-Joachim Bungartz
Dipl.-Inf. Christoph Riesinger
Dipl.-Math. Jürgen Bräckle

WS 2013/2014

Numerisches Programmieren, Übungen

5. Übungsblatt: Diskrete Fourier-Transformation, FFT

1) Frequenzanalyse

Eine von vielen Anwendungen der DFT ist die Transformation eines diskreten Signals $s \in \mathbb{C}^n$ aus dem Wertebereich (z.B. räumliche oder zeitliche Domäne) in den Spektralbereich (z.B. Frequenzraum).

Dargestellt wird der Spektralbereich für gewöhnlich durch das Frequenzspektrum. Es gibt an, wie hoch die jeweiligen Anteile der Grundfrequenzen im Ausgangssignal sind. Die k-te Grundfrequenz ist durch $e^{i\frac{2\pi}{n}k}$ definiert.

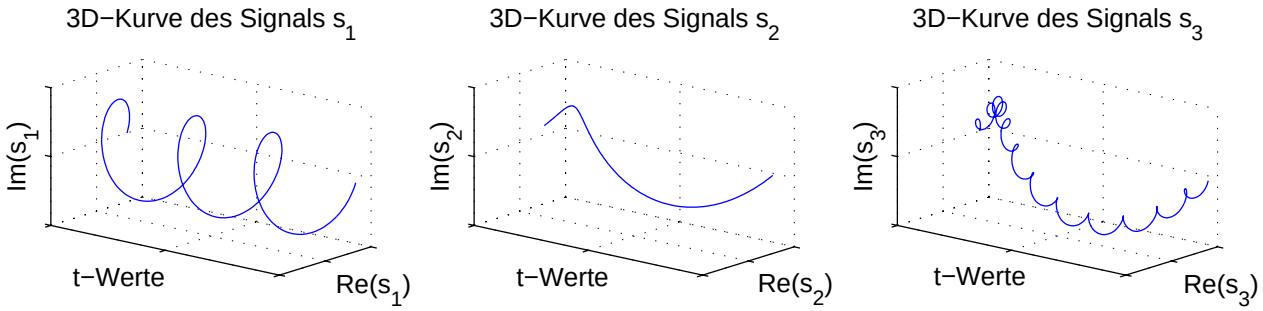

Abbildung 1: Komplexe Signale

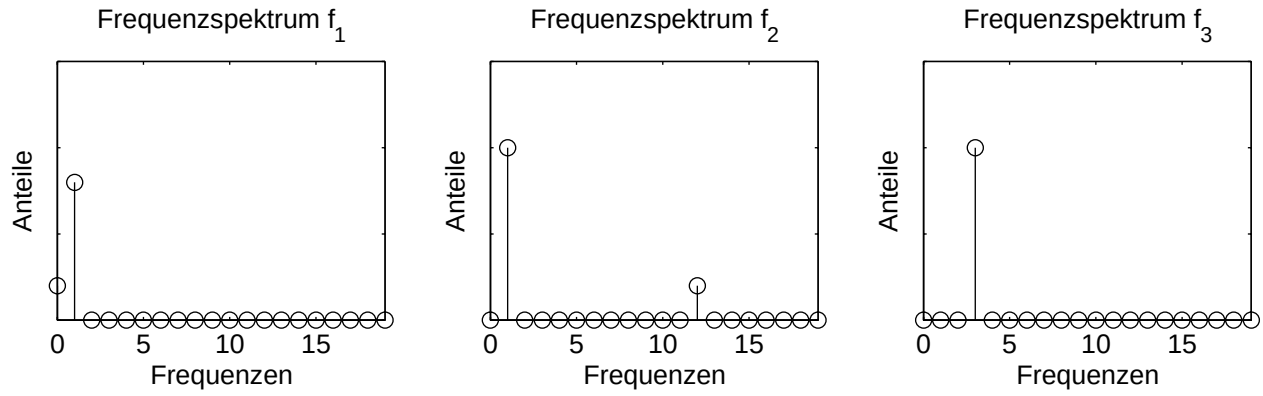

Abbildung 2: Frequenzspektren der Signale

Ordnen Sie den Signalen s_1 bis s_3 (Abbildung 1) die Frequenzspektren f_1 bis f_3 (Abbildung 2) zu! Begründen Sie darüber hinaus Ihre Entscheidung!

Dabei sind die Signale zum besseren Verständnis kontinuierlich dargestellt (z.B. $s_1 = e^{3it},\ t \in [0; 2\pi]$), wohingegen die Frequenzspektren den Betrag der (komplexen) Koeffizienten der DFT mit $n = 21$ gesampelten Werten des jeweiligen Signals beschreiben.

2) Eigenschaften der diskreten Fourier-Transformation (DFT)

Wie in der Vorlesung ist die diskrete Fourier-Transformation von komplexen Eingabedaten $v = (v_0, v_1, \ldots, v_{n-1})^T$ definiert als

$$c_k \;=\; \big(DFT(v)\big)_k \;:=\; \tfrac{1}{n} \sum_{j=0}^{n-1} v_j \cdot \overline{\omega}^{jk} \qquad\qquad k = 0, 1, \ldots, n-1 \tag{1}$$

mit $\omega = \exp(i\,\frac{2\pi}{n})$.

Analog ist die inverse diskrete Fourier-Transformation als Auswertung des trigonometrischen Interpolationspolynoms an den Interpolationspunkten definiert:

$$v_l \;=\; \big(IDFT(c)\big)_l \;:=\; \sum_{k=0}^{n-1} c_k \cdot \omega^{kl} \qquad\qquad l = 0, 1, \ldots, n-1. \tag{2}$$

a) Zeigen Sie, dass gilt: $DFT(v) = \frac{1}{n}\,\overline{IDFT(\bar{v})}$.

b) Zeigen Sie, dass gilt: $DFT(v + u) = DFT(v) + DFT(u)$.

c) Zeigen Sie, dass folgende Beziehungen in Bezug auf ω gelten:

$$\sum_{k=0}^{n-1} \overline{\omega}^{kl} \;=\; \begin{cases} n, & \text{für } l = 0 \\ 0, & \text{für } l = 1, \ldots, n-1 \end{cases}$$

$$\sum_{k=0}^{n-1} \omega^{kl}\ \overline{\omega}^{kj} \;=\; \begin{cases} n, & \text{für } l = j \\ 0, & \text{für } l \neq j \end{cases} \quad .$$

d) Zeigen Sie, dass DFT und $IDFT$ tatsächlich Umkehroperationen sind, indem Sie mit Hilfe der Definitionen (1) und (2) nachweisen:

$$IDFT(DFT(v))_l = v_l$$

e) Berechnen Sie die diskrete Fouriertransformation DFT für folgende drei Vektoren $a, b, c \in \mathbb{C}^n$!
$a = (a_0, a_1, \ldots, a_{n-1})^T = (1, 0, 0, \ldots, 0, 0)^T$,
$b = (b_0, b_1, \ldots, b_{n-1})^T = (i, i, i, \ldots, i, i)^T$,
$c = (c_0, c_1, \ldots, c_{n-1})^T = (1, -1, 1, -1, \ldots)^T$ für gerades n!

3) Schnelle Fourier-Transformation (FFT)

In dieser Aufgabe wollen wir die schnelle Fouriertransformation (FFT) im Detail nachvollziehen und mit der direkten Formel der inversen diskreten Fourier-Transformation ($IDFT$) vergleichen. Der rekursive FFT-Algorithmus lautet:

```
FUNCTION [v[0],...,v[n-1]]=IDFT(c[0],...,c[n-1],n)
if n==1
   v[0] = c[0];
else
   m = n/2;
   z1 =IDFT(c[0],c[2],...,c[n-2],n/2);
   z2 = IDFT(c[1],c[3],...,c[n-1],n/2);
   omega = exp(2*pi*i/n);
   for j=0,...,m-1
      v[j]    = z1[j] + omega^j * z2[j];
      v[m+j]  = z1[j] - omega^j * z2[j];
   end
end
return;
```

Der Algorithmus kann wie in der Vorlesung durch eine Sortierphase und eine Kombinationsphase (sukzessive Anwendung des Butterfly-Operators) visualisiert werden (vgl. Abb. 3 und Abb. 4).

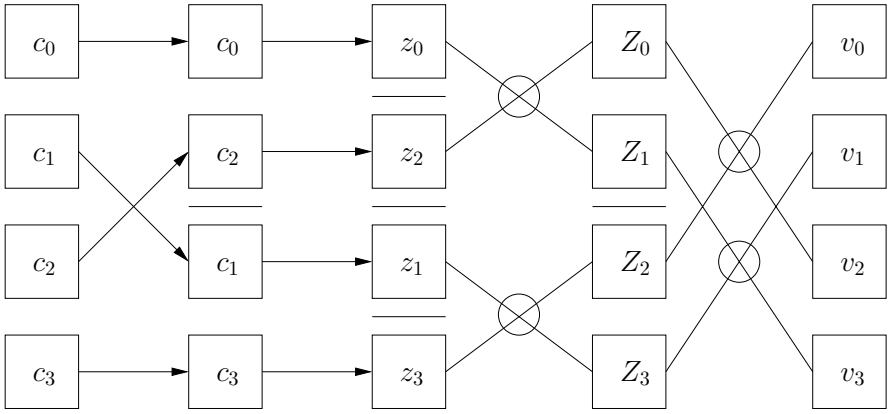

Abbildung 3: Schematischer Ablauf des FFT-Algorithmus.

Der Einfachheit halber soll nun $n = 4$ sein, d.h. wir wollen mit der $IDFT$ aus den vier Koeffizienten c_0, c_1, c_2, c_3 die zugehörigen vier Funktionswerte v_0, v_1, v_2, v_3 berechnen.

Die direkte Formel dazu lautet ja

$$v_j \;=\; \sum_{k=0}^{n-1} c_k \omega^{jk}, \quad j = 0, \ldots, n-1. \tag{3}$$

a) Berechnen Sie die v_j zunächst nach der direkten Formel (3)!

b) Verwenden Sie nun den FFT-Algorithmus, um zu zeigen, dass man damit tatsächlich dasselbe Ergebnis wie in a) erhält!

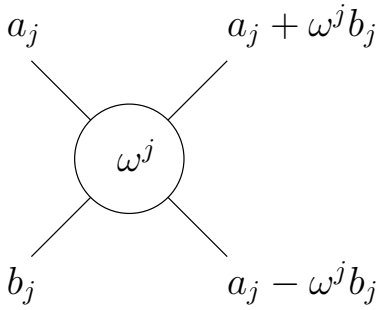

Abbildung 4: Der Butterfly-Operator.

4) Zusatzaufgabe: Sound-Effekte oder: Wozu brauchen wir das überhaupt?

Neben zahlreichen anderen Anwendungen spielt (schnelle) Fourier-Transformation eine wichtige Rolle bei Sound-Effekten, die wir hier am Beispiel eines einfachen Spieleszenarios behandeln wollen.

Betritt man z. B. in einem Spiel eine Kathedrale, so sollten die Schritte oder auch der Sound von Schüssen von den Wänden zurückhallen um ein realistischeres Erlebnis für den Spieler zu schaffen. Zuerst sollten wir uns deshalb überlegen, wie so ein Zurückhallen überhaupt gemessen werden kann, bevor wir unsere Soundsamples (Schritte, Schüsse, etc.) verarbeiten.

Beim Zurückhallen wird zu jedem Geräusch ein bestimmter Anteil nach einer bestimmten Zeit "zurückgeschickt". Um diese Anteile festzustellen, könnte man in einer Halle einen einzelnen Impuls aussenden - z. B. durch das Klatschen in die Hände, durch einen Impuls der im Computer erzeugt wurde, durch einen lauten Knall o. ä. Methoden. Direkt nach dem Impuls wird der Rückhall über ein Mikrofon gemessen.

Dieses Ergebnis speichern wir uns in dem Vektor $\kappa = (\kappa_0, \ldots, \kappa_{T-1})$ und nutzen dies um ein allgemeines Signal wie z. B. das Geräusch unserer Fußschritte mit einem ähnlichem Hall zu versehen. Dies lässt sich mit der sogenannten Faltung (Symbol $*$) berechnen:

Für ein Eingangssignal $E(t)$ ergibt sich das Ausgangssignal $A(t)$ wie folgt:

$$A(t) = (E * K)(t) = \sum_{i_t=0}^{T-1} E(i_t) \cdot K(t - i_t)$$

K wird Kernelfunktion genannt. Sie beschreibt, welche Signale des Eingangssignal E zum Zeitpunkt t ausgegeben werden ($A(t)$). Sie gibt zu diskreten Abtaststellen $x \in [-k; k-1]$ die dazugehörige Komponente des Vektors κ zurück (Teilaufgabe c)). Welche das sein wird, werden wir im Laufe des Aufgabenblattes feststellen. Desweiteren legen wir fest, dass $K(x) = 0$ für alle $x \notin [-k; k-1]$ und $E(t) = 0$ für alle $t \notin [0; T-1]$.

Wie man sehr schnell sieht, ist der Rechenaufwand linear für jeden Samplingpunkt zum Zeitpunkt t und sogar quardatisch für alle Samplingpunkte unseres Echointervalls $[0; T-1]$. Eine Möglichkeit wäre, das Echo nur für bestimmte Kernelelemente zu aktivieren, in der Kernelfunktion also > 0 zu setzen (die anderen Elemente des Vektors κ würde man dann auf 0 setzen), um damit die teure quadratische Auswertung zu "mildern". Allerdings wollen wir einen realistischen Sound erzeugen und gehen davon aus, dass der Kernelvektor voll besetzt ist.

Um aus dem Dilemma der quadratischen Kosten zu entkommen ermöglicht uns die FFT einen Ausweg: *Die Faltung im Frequenzraum.* Diese erlaubt es uns nach der Transformation in den Frequenzraum (im Weiteren mit F bezeichnet bzw. mit F^{-1} für die Rücktransformation) die Faltung komponentenweise ($\cdot$) zu vollziehen:

$$(E * K) = F^{-1}(F(E * K)) = F^{-1}(F(E) \cdot F(K))$$

a) Überlegen Sie sich, wie ein Algorithmus für eine solche Verarbeitung eines Soundsignals aussehen müsste. Erstellen Sie eine Skizze mit Eingabesignalen, Kernel, Faltung und

dem Ausgabesignal.

b) Zeigen Sie, dass die Faltung mit $\kappa_a = \delta_{a,0}$ das Eingangssignal E unverändert zurück gibt.

c) Zeigen Sie, dass die Faltung mit $\kappa_a = \delta_{a,c}$ das Eingangssignal $E(-c)$ für eine Konstante c zurück gibt. Was bedeutet das für den Faltungskern in ihrem skizzierten Algorithmus?