

FFT und Anwendungen

Projekt zur Vorlesung „Einführung in die Numerik“

David Cohen, Christian Stohrer

Frühlingssemester 2011

Intro

Trigonometrische Interpolation

Definitionen und Bezeichnungen
Komplexe Formulierung

Diskrete Fourier Transformation (DFT)

Definitionen
Eigenschaften
Beispiele

Schnelle Fourier Transformation (FFT)

Aufwand der DFT
Herleitung der FFT
Implementierung
Zweidimensional Fourier Transformation

Hinweise zum Projekt

Übersicht über die Aufgaben
Literaturverzeichnis

Intro

- ▶ FFT - Was ist das?
 - ▶ Forum Freies Theater
 - ▶ Final Fantasy Tactics
 - ▶ Football Federation Tasmania
 - ▶ Fast Fourier Transformation
- ▶ Wozu braucht man das?
 - ▶ Analyse periodischer Vorgänge.
- ▶ Was gibt es für Anwendungen?
 - ▶ Signalverarbeitung
 - ▶ Filterung von Messdaten
 - ▶ Digitale Sound- und Bildbearbeitung
 - ▶ ...

Trigonometrische Interpolation

Definitionen und Bezeichnungen

Gegeben seien N Punkte

$$(x_k, y_k)_{k=0}^{N-1}$$

mit $x_k = 2\pi k/N$ und $y_k := f(x_k) \in \mathbb{R}$ für $k = 0, \dots, N-1$ mit f periodisch.

Für das ganze Projekt schränken wir uns der Einfachheit halber auf den Fall $N = 2^n$, mit $n \in \mathbb{N}$ ein. Zusätzlich setzen wir $M = N/2$.

Trigonometrische Interpolation

Trigonometrisches Polynom

Wir suchen ein **trigonometrisches Polynom**, also ein Ausdruck der Form,

$$p_N(x) := \frac{A_0}{2} + \sum_{\ell=1}^{M-1} (A_\ell \cos(\ell x) + B_\ell \sin(\ell x)) + \frac{A_M}{2} \cos(Mx),$$

das durch diese Punkte geht. Das heisst

$$p_N(x_k) = y_k \quad \text{für } k = 0, \dots, N-1.$$

Dies sind N Gleichungen für N Unbekannte.

AUFGABE: Finde die Koeffizienten $A_\ell, B_\ell \in \mathbb{R}$!

Trigonometrische Interpolation

Komplex ist's einfacher!

Nun lassen wir auch $y_k \in \mathbb{C}$ zu. Die Eulersche Formel

$$e^{ix} = \cos(x) + i \sin(x)$$

motiviert den Ansatz

$$\tilde{p}_N(x) = \sum_{\ell=0}^{N-1} z_\ell e^{i\ell x}.$$

NEUE AUFGABE: Finde die Koeffizienten $z_\ell \in \mathbb{C}$ so, dass

$$\tilde{p}_N(x_k) = y_k.$$

für alle $k = 0, \dots, N-1$.

Diese zwei Aufgaben sind **äquivalent**!

Die Berechnung der z_k zu gegebenen y_k nennt man **diskrete Fourier Transformation**.

Trigonometrische Interpolation

Äquivalenz der Aufgaben – Formel für die Koeffizienten

Um die Äquivalenz zu zeigen benötigen wir die zwei Formeln

$$\cos(x) = \frac{e^{ix} + e^{-ix}}{2} \quad \text{und} \quad \sin(x) = \frac{e^{ix} - e^{-ix}}{2i}.$$

Zusätzlich erinnern wir uns an die Definition der x_k ($x_k = 2\pi k/N$) und schliessen damit

$$e^{-ix_k} = e^{i(2\pi - x_k)} = e^{i\frac{2\pi(N-k)}{N}} = e^{ix_{N-k}}.$$

Mit einem Koeffizientenvergleich erhalten wir

$$A_0 = 2z_0,$$

$$A_\ell = z_\ell + z_{N-\ell}, \quad B_\ell = i(z_\ell - z_{N-\ell}), \quad \text{für } \ell = 1, \dots, M-1,$$

$$A_M = 2z_M.$$

Diskrete Fourier Transformation (DFT)

Definitionen

Der Raum P_N der komplexen N -periodischen Funktion ist gegeben durch

$$P_N := \{(y_k)_{k \in \mathbb{Z}} : y_k \in \mathbb{C} \text{ und } y_{k+N} = y_k\}.$$

BEMERKUNG: Ein $y \in P_N$ ist durch y_0, \dots, y_{N-1} eindeutig bestimmt.

Die **diskrete Fourier Transformation** von $y \in P_N$ ist die Folge $(z_k)_{k \in \mathbb{Z}}$ wobei

$$z_k := \frac{1}{N} \sum_{\ell=0}^{N-1} y_\ell e^{-ikx_\ell} = \frac{1}{N} \sum_{\ell=0}^{N-1} y_\ell \omega^{-k\ell}$$

mit $x_\ell = 2\pi\ell/N$ für $\ell = 0, \dots, N-1$ und $\omega = e^{2\pi i/N}$.

BEMERKUNG: Aus Zeitgründen lassen wir den Beweis hier aus.

NOTATION: $z = \mathcal{F}_N(y)$.

Diskrete Fourier Transformation (DFT)

Eigenschaften und IDFT

- Für $y \in P_N$ hat man $z = \mathcal{F}_N(y) \in P_N$.
- $\mathcal{F}_N: P_N \rightarrow P_N$ ist linear und bijektiv.

Die Umkehrabbildung (**IDFT – inverse diskrete Fourier Transformation**) von \mathcal{F}_N ist gegeben durch

$$y_k = (\mathcal{F}_N^{-1}(z))_k = \sum_{\ell=0}^{N-1} z_\ell \omega^{k\ell},$$

und $\omega = e^{2\pi i/N}$.

Die IDFT und die DFT sind sehr ähnlich. Die zwei Unterschiede sind

- Kein Vorfaktor bei IDFT.
- Vorzeichen im Exponenten von ω . (DFT: $\omega^{-k\ell}$, IDFT: $\omega^{k\ell}$)

Diskrete Fourier Transformation (DFT)

Beispiele – $N = 1$ und $N = 2$

$$N = 1: \quad x = 0, y = y_0, \omega = e^{2\pi i/1} = 1.$$

$$z = z_0 = \frac{1}{N} \sum_{\ell=0}^{N-1} y_\ell \omega^{-0\ell} = \frac{1}{1} y_0 1^0 = y_0 = y$$

$$N = 2: \quad x = (0, \pi), y = (y_0, y_1), \omega = e^{2\pi i/2} = -1.$$

$$z_0 = \frac{1}{N} \sum_{\ell=0}^{N-1} y_\ell \omega^{-0\ell} = \frac{1}{2} (y_0 (-1)^0 + y_1 (-1)^0) = \frac{y_0 + y_1}{2}$$

$$z_1 = \frac{1}{N} \sum_{\ell=0}^{N-1} y_\ell \omega^{-1\ell} = \frac{1}{2} (y_0 (-1)^0 + y_1 (-1)^{-1}) = \frac{y_0 - y_1}{2}$$

Diskrete Fourier Transformation (DFT)

Beispiele – $\sin(3x)$

Wir betrachten nun die Funktion $f(x) = \sin(3x)$ und $N = 64$ Punkten $(x_k, y_k)_{k=0}^{N-1}$ mit $x_k = 2\pi k/N$ und $y_k = f(x_k)$. Wir betrachten das Frequenzspektrum.

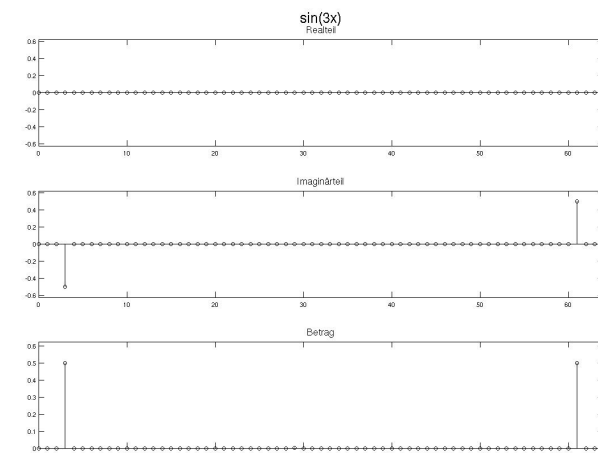
Wir stellen unter anderem folgendes fest:

- \sin ergibt imaginäre Anteile.
- Niedrige Frequenzen bei 0 und N .
- Höchste Frequenz bei $M = N/2$.
- Symmetrie um M .
- Man „sieht“ die Formeln der Trigonometrischen Interpolation.

Man kann das Frequenzspektrum auch um die 0-Frequenz zentriert darstellen. (Vergleiche `plotfreq.m`.)

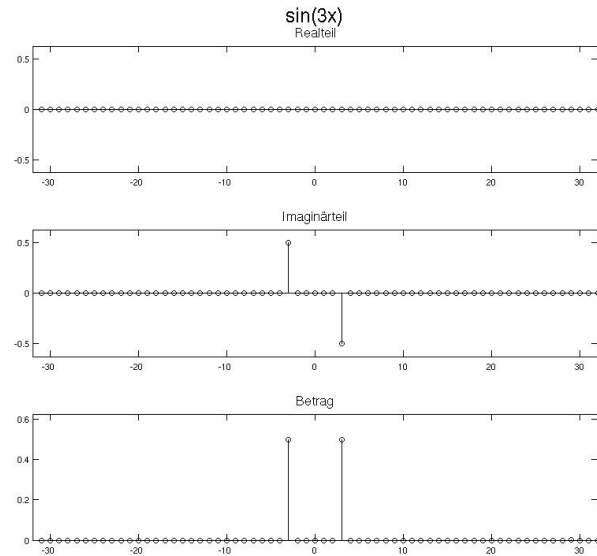
Diskrete Fourier Transformation (DFT)

Beispiele – $\sin(3x)$



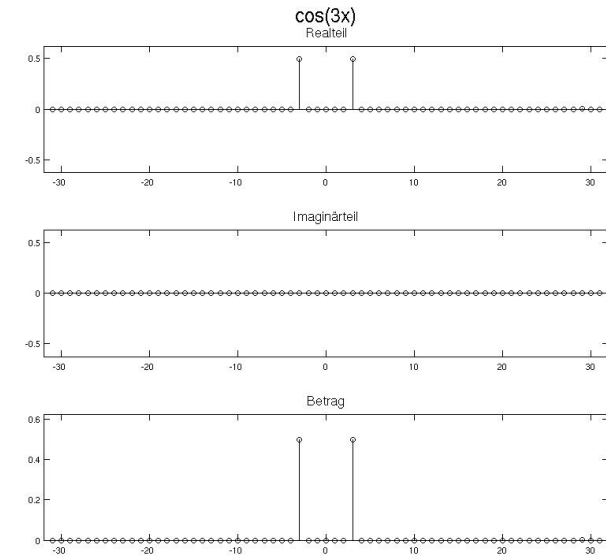
Diskrete Fourier Transformation (DFT)

Beispiele – $\sin(3x)$



Diskrete Fourier Transformation (DFT)

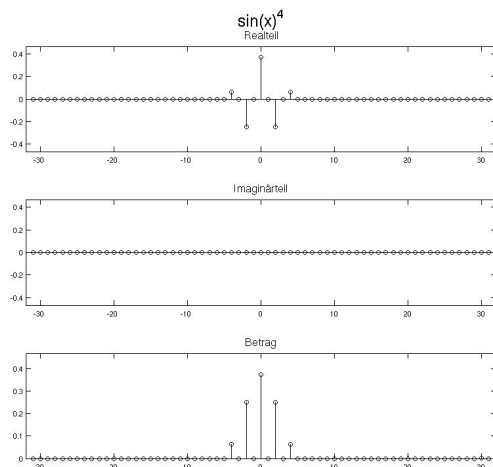
Beispiele – Das ganze nun mit $\cos(3x)$, direkt geshiftet



Diskrete Fourier Transformation (DFT)

Beispiele – Additionstheorem mittels Fouriertransformation

Wir wollen $\sin(x)^4$ mittels $\sin(kx)$ und $\cos(kx)$ darstellen.



Wir erkennen die Formel $\sin(x)^4 = 3/8 - 1/2 \cos(2x) + 1/8 \cos(4x)$

Schnelle Fourier Transformation (FFT)

Aufwand der DFT

Wir betrachten nun einen einfachen Algorithmus zur Berechnung der DFT. (Hier nur als Pseudo-Code angegeben. Die zugehörige MATLAB-Funktion `dft.m` kann heruntergeladen werden.)

```
function z = dft(y)
    N := length(y)
     $\omega := e^{2\pi i/N}$ 
    for k = 0, ..., N - 1 do
         $z_k := \frac{1}{N} \sum_{\ell=0}^{N-1} y_{\ell} \omega^{-k\ell}$ 
    end for
end
```

Dieser Algorithmus durchläuft die for-Schleife N mal. Bei jedem Durchlauf ist eine Summe von N -Einträgen zu berechnen. Dies führt zu einem Aufwand von $\mathcal{O}(N^2)$.

Dies bedeutet vierfacher Aufwand bei doppelter Länge!

Schnelle Fourier Transformation (FFT)

Idee und Erfinder

ZIEL: Berechne die DFT mit weniger Aufwand.

Der FFT-Algorithmus berechnet die DFT mit einem Aufwand von $\mathcal{O}(N \log(N))$. Er wurde von Cooley und Tukey im Jahr 1965 vorgestellt. Sie stützten sich dabei auf Ideen von Runge (1925).

IDEE:

„vierfacher Aufwand bei doppelter Länge“

=

„ein Viertel des Aufwands bei halber Länge“

Versuche also die DFT der Länge N aus zwei DFTs der Länge $N/2$ zusammenzusetzen. (Divide and conquer!)

Schnelle Fourier Transformation (FFT)

Herleitung - Notationen

Wir verwenden die Notationen

$$N = 2^n, \quad n \in \mathbb{N}, \quad M = \frac{N}{2}, \quad \omega_N = e^{2\pi i/N}.$$

Damit erhalten wir

$$\frac{1}{N} = \frac{1}{2M}, \quad \omega_N^2 = e^{2 \cdot 2\pi i/N} = e^{2\pi i/M} = \omega_M.$$

Wir teilen unseren Vektor y folgendermassen in zwei Subvektoren y^0 und y^1 auf

$$y_k^j = y_{2k+j}, \quad \text{für } k = 0, \dots, N/2 - 1 \text{ und } j = 0, 1.$$

Ist $y \in P_N$, so gilt $y^0, y^1 \in P_M$.

Schnelle Fourier Transformation (FFT)

Herleitung - Rechnung

$$\begin{aligned} z_k &= (\mathcal{F}_N(y))_k = \frac{1}{N} \sum_{\ell=0}^{N-1} y_\ell \omega_N^{-k\ell} \\ &= \frac{1}{N} \sum_{\ell=0}^{M-1} y_{2\ell} \omega_N^{-k(2\ell)} + \frac{1}{N} \sum_{\ell=0}^{M-1} y_{2\ell+1} \omega_N^{-k(2\ell+1)} \\ &= \frac{1}{N} \sum_{\ell=0}^{M-1} y_\ell^0 \omega_N^{-k(2\ell)} + \frac{1}{N} \sum_{\ell=0}^{M-1} y_\ell^1 \omega_N^{-k(2\ell+1)} \\ &= \frac{1}{N} \sum_{\ell=0}^{M-1} y_\ell^0 \omega_M^{-k\ell} + \frac{1}{N} \sum_{\ell=0}^{M-1} y_\ell^1 \omega_N^{-k} \omega_M^{-k\ell} \\ &= \frac{1}{2} \left(\frac{1}{M} \sum_{\ell=0}^{M-1} y_\ell^0 \omega_M^{-k\ell} + \frac{1}{M} \sum_{\ell=0}^{M-1} y_\ell^1 \omega_N^{-k} \omega_M^{-k\ell} \right) \end{aligned}$$

Schnelle Fourier Transformation (FFT)

Herleitung - Resultat

$$z_k = (\mathcal{F}_N(y))_k = \frac{1}{2} \left(\mathcal{F}_M(y^0)_k + \omega_N^{-k} \mathcal{F}_M(y^1)_k \right) \quad \text{für } k = 0, \dots, N-1.$$

Oder mit $\omega_N^{-(M+k)} = \omega_N^{-M} \omega_N^{-k} = \omega_N^{-N/2} \omega_N^{-k} = e^{-i\pi} \omega_N^{-k} = -\omega_N^{-k}$
noch besser

$$\begin{aligned} z_k &= (\mathcal{F}_N(y))_k = \frac{1}{2} \left(\mathcal{F}_M(y^0)_k + \omega_N^{-k} \mathcal{F}_M(y^1)_k \right) \\ z_{M+k} &= (\mathcal{F}_N(y))_{M+k} = \frac{1}{2} \left(\mathcal{F}_M(y^0)_k - \omega_N^{-k} \mathcal{F}_M(y^1)_k \right) \end{aligned}$$

für $k = 0, \dots, M-1$.

Man kann auch die IDFT so behandeln! Die Ergebnisse und Algorithmen sind mutandis mutatis dieselben.

Schnelle Fourier Transformation (FFT)

Implementierung -Pseudocode

```
function z = fft(y)
    N := length(y)
    if N = 1 then
        z := y
    else
        ▷ Rekursive Berechnung
        ▷  $(y^0)_k = y_{2k}$ 
        ▷  $(y^1)_k = y_{2k+1}$ 
         $z^0 := \text{fft}(y^0)$ 
         $z^1 := \text{fft}(y^1)$ 
         $\omega := e^{2\pi i/N}$ 
         $M := N/2$ 
        for k = 0, ..., M - 1 do
             $z_k := (z_k^0 + \omega^{-k} z_k^1)/2$ 
             $z_{M+k} := (z_k^0 - \omega^{-k} z_k^1)/2$ 
        end for
    end if
end
```

Schnelle Fourier Transformation (FFT)

Das ganze nun in 2D

In der Bildverarbeitung benötigt man oft Fouriertransformationen von Matrizen. Dies entspricht einer zweidimensionalen Fouriertransformation. Diese kann man auf den eindimensionalen Fall zurückführen.

Man wendet einfach zuerst eindimensionale Fouriertransformation auf die Zeilen und danach auf die Spalten an.

Ohne FFT wäre der Aufwand dafür schnell viel zu gross. Mit der FFT lässt sich dies jedoch problemlos durchführen.

Schnelle Fourier Transformation (FFT)

„2D-Kochrezept“

Sei Y eine $N \times N$ -Matrix. Wir wollen ihre zweidimensionale Fouriertransformation Z – ebenfalls eine $N \times N$ -Matrix – berechnen und gehen dazu folgendermassen vor.

- ▶ Wende eine eindimensionale FFT auf jede einzelne Zeile von Y an und berechne damit die $N \times N$ -Hilfsmatrix H . Die Zeilen von H sind die eindimensionalen Fouriertransformationen der Zeilen von Y .
- ▶ Wende eine eindimensionale FFT auf jede einzelne Spalte von H an und berechne damit Z . Die Spalten von Z sind die (eindimensionalen) Fouriertransformationen der Spalten von H .

Für die zweidimensionale inverse Fouriertransformation muss man zuerst die Spalten und dann die Zeilen zurück transformieren.

Projektaufgaben

Übersicht

Die Aufgaben (siehe Projekt-Beschrieb) sind in vier Teile unterteilt.

1. Fast Fourier Transform
 - ▶ Programmieren der FFT und der IFFT
 - ▶ Vergleich der Laufzeiten von FFT
 - ▶ Variation der FFT
2. Trigonometrische Interpolation
3. Tonerkennung und Soundbearbeitung
 - ▶ Akkorderkennung
 - ▶ Verbesserung einer Tonaufnahme (Was kann man mit unseren einfachen Methoden erreichen? Resultat muss nicht perfekt sein!)
4. Bildbearbeitung
 - ▶ Zweidimensional Fourier Transformation
 - ▶ Kantendetektion mittels Hochpassfilter
 - ▶ Weichzeichner mittels Tiefpassfilter

Projektaufgaben

Beispiel zur Bildbearbeitung – Hochpassfilter

Ein **Hochpassfilter** ist ein Filter der hohe Frequenzen durchlässt und niedrige Frequenzen dämpft oder löscht.

Original



Hochpassfilter



Projektaufgaben

Beispiel zur Bildbearbeitung – Tiefpassfilter

Ein **Tiefpassfilter** ist ein Filter der niedrige Frequenzen durchlässt und hohe Frequenzen dämpft oder löscht.

Original



Tiefpassfilter



Literaturverzeichnis

- ▶ James W. Cooley and John W. Tukey, *An Algorithm for the Machine Calculation of Complex Fourier Series*, Mathematics of Computation (1965), Vol. 19, No. 90, pp. 297-301
- ▶ Tilman Butz, *Fouriertransformation für Fussgänger* (6. Auflage), Vieweg+Teubner (2009), ISBN 78-3-8348-0538-6
- ▶ Ulrich Stein, *Einstieg in das Programmieren mit MATLAB*, Hanser (2007), ISBN 978-3-446-41009-1
- ▶ Josef Stoer, *Numerische Mathematik 1* (8. Auflage), Springer (1999), ISBN 3-540-66154-9
- ▶ Wikipedia (http://de.wikipedia.org/wiki/Schnelle_Fourier-Transformation)
- ▶ Wolfram Mathworld (<http://mathworld.wolfram.com/FastFourierTransform.html>)