(Fast) Fourier Transformation und ihre Anwendungen

Johannes Lülff

Universität Münster

14.01.2009

Inhaltsverzeichnis

Definition

Fouriertransformation

$$F(\omega) = \mathcal{F}\left[f(t)\right](\omega) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}t\, f(t) e^{-i\omega t}$$

Fouriersynthese

$$f(t) = \mathcal{F}^{-1}\left[F(\omega)\right](t) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{d}\omega\, F(\omega) e^{i\omega t}$$

Beschreibung

- Fouriertransformation 'entwickelt' eine Funktion nach $\sin$ und $\cos$
- Die Fouriertransformierte beschreibt die Intensität der sog. 'Moden'
- FT überführt eine Funktion von der *zeitlichen Domäne* in die *Frequenzdomäne*
- oder analog: Vom Ortsraum in den (Orts-)Frequenzraum

Rechenregeln

Linearität

$$\mathcal{F}[\alpha f(t) + \beta g(t)] = \alpha \mathcal{F}[f(t)] + \beta \mathcal{F}[g(t)]$$

Faltung

$$\mathcal{F}[f(t) \cdot g(t)] = \mathcal{F}[f(t)] * \mathcal{F}[g(t)]$$

Ableitung

$$\mathcal{F}\left[\frac{\mathrm{d}^{\mathrm{n}}}{\mathrm{d}t^n} f(t)\right] = (i\omega)^n \mathcal{F}[f(t)]$$

Periodische FT

Falls $f(t)$ periodisch mit $f(t) = f(t+T)$: Fouriersynthese wird zur diskreten Fourierreihe!

$$f(t) = \sum_{n=-\infty}^{\infty} \hat{a}_n e^{\frac{2\pi i}{T} nt}$$

mit den Fourierkomponenten

$$\hat{a}_n = \frac{1}{T} \int_0^T \mathrm{d}t \, f(t) e^{-\frac{2\pi i}{T} nt}$$

Diskrete FT

Falls $f(t)$ nur an N diskreten Datenpunkten $a_{i=0\ldots N-1}$ gegeben:
Endlich viele Fourierkomponenten $\hat{a}_n$!

$$\hat{a}_n = \sum_{k=0}^{N-1} a_k e^{-\frac{2\pi i}{N} kn}$$

Rücktrafo gegeben durch

$$a_n = \frac{1}{N} \sum_{k=0}^{N-1} \hat{a}_k e^{\frac{2\pi i}{N} kn}$$

Fast Fourier Transformation

Naives Ausführen der DFT benötigt $\mathcal{O}(N^2)$ komplexe Multiplikationen.

Berechnung der DFT möglich in $\mathcal{O}(N \log N)$ mittels **Fast** Fourier Transformation:

FFT

- Grundlagen bereits 1805 durch Gauss, formelle Beschreibung 1965 durch Cooley / Tukey
- Klassischer *Divide-and-Conquer*–Algorithmus: Unterteile DFT der Größe N in 2 DFTs der Größe $N/2$.

Fast Fourier Transformation

Zerlege die Summe der DFT in gerade (**e**ven) und ungerade (**o**dd) Anteile:

$$
\begin{aligned}
F_k &= \sum_{j=0}^{N-1} f_j e^{-\frac{2\pi i}{N} jk} \\
&= \sum_{j=0}^{N/2-1} f_{2j} e^{-\frac{2\pi i}{N}(2j)k} + \sum_{j=0}^{N/2-1} f_{2j+1} e^{-\frac{2\pi i}{N}(2j+1)k} \\
&= \sum_{j=0}^{N/2-1} f_{2j} e^{-\frac{2\pi i}{N/2} jk} + e^{-\frac{2\pi i}{N} k} \sum_{j=0}^{N/2-1} f_{2j+1} e^{-\frac{2\pi i}{N/2} jk} \\
&= F_k^{(e)} + W^k F_k^{(o)}
\end{aligned}
$$

Fast Fourier Transformation

DFT der Größe N als 2 DFTs der Größe $N/2$ dargestellt, damit Rechenaufwand von N^2 auf $2 \cdot (N/2)^2$ reduziert.

$\rightarrow$ rekursiv weitermachen! ($N = 2^p$ wird vorrausgesetzt)

$$
\begin{aligned}
F_k &= F_k^{(e)} + W^k F_k^{(o)} \\
&= F_k^{(ee)} + W^k F_k^{(eo)} + W^k F_k^{(oe)} + W^{2k} F_k^{(oo)} \\
&\vdots \quad p \text{ Schritte} \\
&= \underbrace{F_k^{(e\ldots e)} + \ldots + W^{\cdots} F_k^{(eooe\ldots o)} + \ldots + W^{pk} F_k^{(o\ldots o)}}_{N \text{ Summanden}}
\end{aligned}
$$

Fast Fourier Transformation

Im letzten Schritt sind N DFTs der Länge 1 zu berechnen
$\rightarrow$ DFT der Länge 1 ist der Funktionswert selber!
$\rightarrow$ Jeder Kombination $(eooe \ldots o)$ entspricht ein Element des Eingabevektors f

Bit Reversal

Aufgabe: Finde für beliebige Kombination $(eooe \ldots o)$ das j, für das $F_k^{(eooe \ldots o)} = f_j$ gilt.

- Drehe Reihenfolge der e's und o's um
- Setze $e \hat{=} 0$ und $o \hat{=} 1$
- Dies ist die binäre Darstellung von j!

Bspl.: $(eeo) \rightarrow (oee) \rightarrow (100) \rightarrow j = 4$

Fast Fourier Transformation

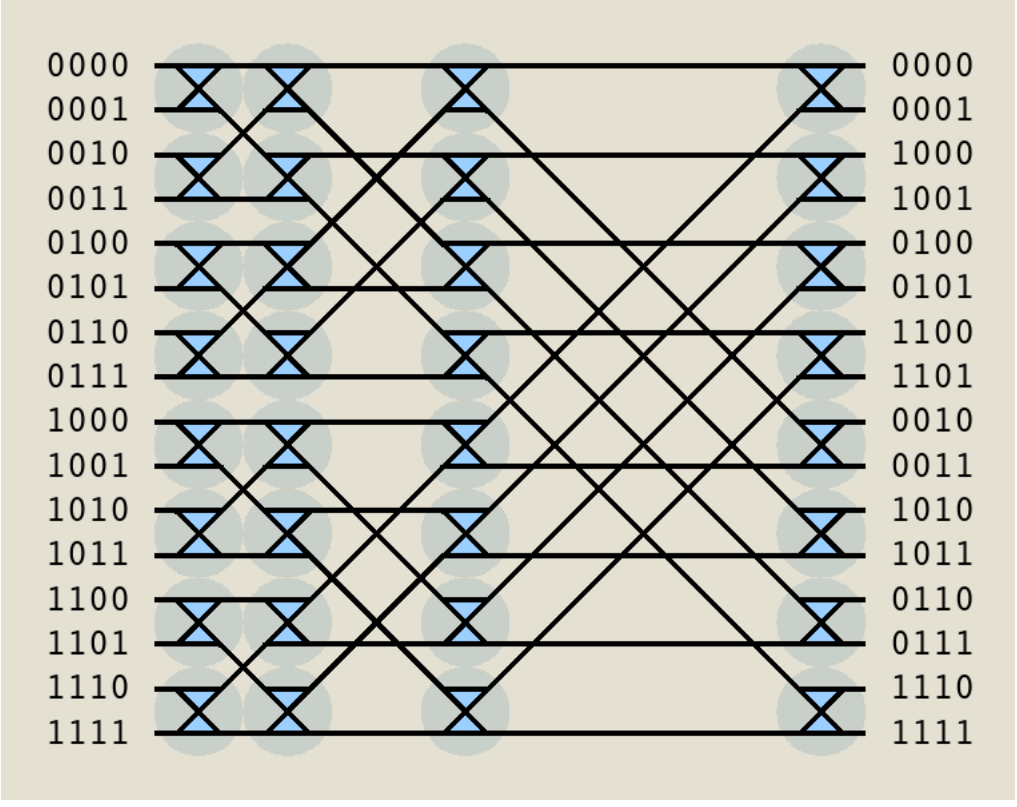
Vorgehen erfolgt 'rückwärts':

- Bringe die Elemente des Eingabearrays in die sog. *bit-reversed order*
- Kombiniere benachbarte Elemente mit den Einheitswurzeln W zu DFTs der Größe 2
- Kombiniere benachbarte Paare zu DFTs der Größe 4

  $\vdots$
- Kombiniere erste Hälfte des Arrays mit zweiter Hälfte zu DFT der Größe N

Fast Fourier Transformation

Visualisierung durch sog. *Schmetterlingsdiagramm*:



Implementierungen der FFT

- Jede mathematische Funktionenbibliothek beinhaltet FFT-Routinen
- Naive Implementierung des Algorithmus benötigt $N = 2^p$
- Quasi jede Bibliothek implementiert Variante für beliebige N
- Algorithmus dann optimal, wenn N aus 'kleinen' Primzahlen besteht

Implementierungen der FFT

Hier: *Fastest Fourier Transform in the West* (www.fftw.org)

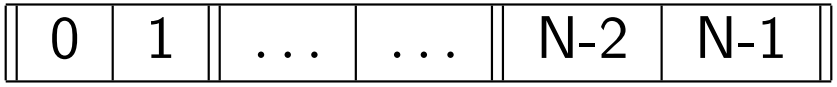
- Open Source
- *Immer* $\mathcal{O}(N \log N)$
- Optimiert Berechnung je nach Prozessorarchitektur
- Gehört zu den schnellsten Implementierungen[1]
- Unterstützt Parallelrechner

[1] http://www.fftw.org/benchfft/

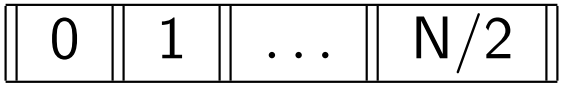
Implementierungen der FFT

Falls Eingabedaten reell, gilt $\hat{a}_k = \overline{\hat{a}_{N-k}}$
$\rightarrow$ Nur $N/2+1$ Fourierkomponenten müssen gespeichert werden
$\rightarrow$ *in-place*–Transformation Komplex$\leftrightarrow$Reell möglich

Anordnung der Fourierkomponenten:
Reell:



Komplex:

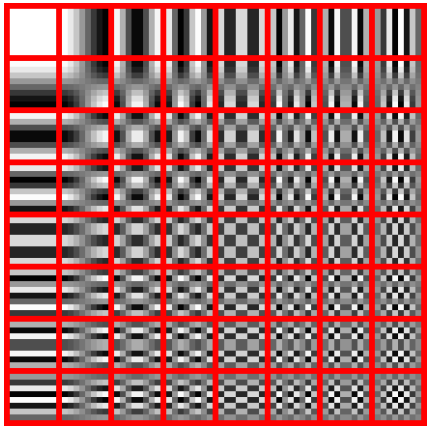


Achtung: FFTW führt nichtnormierte FT durch $\rightarrow$ Nach Hin- und Rücktrafo sind Daten mit N skaliert.

Kompression

Signale werden im Frequenzraum manipuliert, um sie zu komprimieren, z.B.:

- MP3-Format: Menschliches Hörvermögen im Bereich 20 Hz - 18 kHz $\rightarrow$ Ausserhalb liegende Frequenzen werden herausgefiltert.
- JPEG-Format: Bild wird in 8x8 Pixel große Blöcke unterteilt, die dann fouriertransformiert werden.



Signaltechnik

Hochpass- / Tiefpass- / Bandpassfilter eines Signals durch Manipulation des Frequenzspektrums

$\rightarrow$ z.B. Bildbearbeitung im Frequenzraum:

- Weichzeichner $\rightarrow$ Tiefpassfilter
- Kantenerkennung $\rightarrow$ Hochpassfilter

Ableiten mittels FT

Es gilt

$$\mathcal{F}\left[\frac{\mathrm{d}^n f}{\mathrm{d}x^n}\right] = (ik)^n \mathcal{F}[f(x)]$$

bzw.

$$\frac{\mathrm{d}^n f}{\mathrm{d}x^n} = \mathcal{F}^{-1}[(ik)^n \mathcal{F}[f(x)]]$$

$\rightarrow$ Ableitungen werden zu Multiplikationen im Frequenzraum!

Zeitaufwand $\mathcal{O}(N \log N)$ statt z.B. $\mathcal{O}(N)$ für finite Differenzen.

Ableiten mittels FT

Quellcode:
Grober Ablauf mit *FFTW*

DGLs und FT

Mittels FT lassen sich partielle Differentialgleichungen lösen!

Beispiel 1 (Spektralverfahren):

Wärmeleitungsgleichung in 1D

$$\partial_t u = \nu \partial_x^2 u$$

Anwenden der FT:

$$\partial_t \mathcal{F}[u] = \nu \mathcal{F}\left[\partial_x^2 u\right] = -\nu k^2 \mathcal{F}[u]$$

$\Rightarrow$ gewöhnliche DGL!

DGLs und FT

Beispiel 2 (Pseudospektralverfahren):

Burgersgleichung in 1D

$$\partial_t u = -u \cdot \partial_x u + \nu \partial_x^2 u$$

(Naives) Anwenden der FT:

$$\begin{aligned}
\partial_t \mathcal{F}[u] &= -\mathcal{F}[u \cdot \partial_x u] + \nu \mathcal{F}\left[\partial_x^2 u\right] \\
&= -\mathcal{F}[u] * \mathcal{F}[\partial_x u] - \nu k^2 \mathcal{F}[u] \\
&= -\mathcal{F}[u] * (ik\mathcal{F}[u]) - \nu k^2 \mathcal{F}[u]
\end{aligned}$$

DGLs und FT

Nichtlinearität wird zum Problem:

Aufwand für (F)FT in $\mathcal{O}(N \log N)$, aber Aufwand für Faltung in $\mathcal{O}(N^2)$!

Deshalb Multiplikation im Ortsraum durchführen, um Faltung zu verhindern:

$$\mathcal{F}\left[u \cdot \partial_x u\right] = \mathcal{F}\left[u \cdot \mathcal{F}^{-1}\left[\mathcal{F}\left[\partial_x u\right]\right]\right] = \mathcal{F}\left[u \cdot \mathcal{F}^{-1}\left[ik\mathcal{F}\left[u\right]\right]\right]$$

DGLs und FT

$\mathcal{O}(N^2)$-Faltung wird durch $\mathcal{O}(N \log N)$-FFT ersetzt
$\rightarrow$ Gesamter Algorithmus in $\mathcal{O}(N \log N)$:

$$\partial_t \mathcal{F}[u] = \mathcal{F}\left[u \cdot \mathcal{F}^{-1}[ik\mathcal{F}[u]]\right] - \nu k^2 \mathcal{F}[u]$$

Gewöhnliche DGL, lässt sich z.B. mittels RK4 lösen.

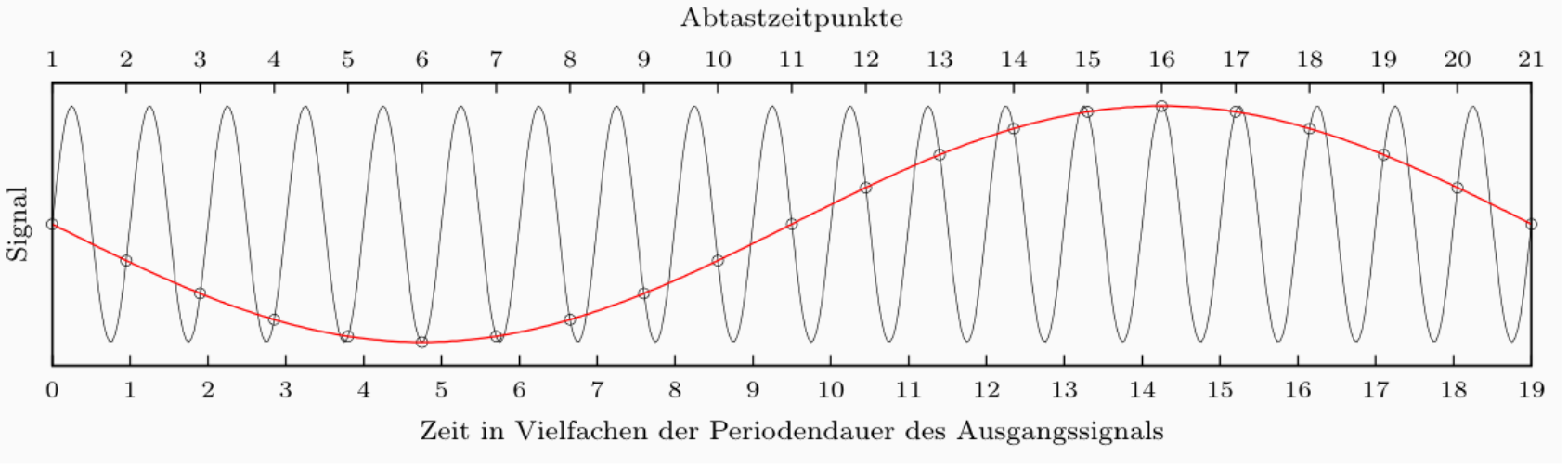
Weiterer Trick: Man führt die Zeitintegration im Fourierraum durch!
Mit $\hat{u} = \mathcal{F}[u]$:

$$\partial_t \hat{u} = \mathcal{F}\left[\mathcal{F}^{-1}[\hat{u}] \cdot \mathcal{F}^{-1}[ik\hat{u}]\right] - \nu k^2 \hat{u}$$

$\rightarrow$ Nur 3 Fouriertransformationen pro Auswertung der RHS!

Aliasing bei DGLs

Diskretisierung auf ein Gitter $\longrightarrow$ es gibt eine kleinste darstellbare räumliche Struktur bzw. größte Wellenzahl k_{max}.
Kleinere Stukturen werden fehlinterpretiert:



Aliasing bei DGLs

Entstehen während der Simulation zu hohe Wellenzahlen, kommt es zu Fehlern

Lineare Operationen 'mischen' Wellenzahlen nicht $\rightarrow$ unproblematisch

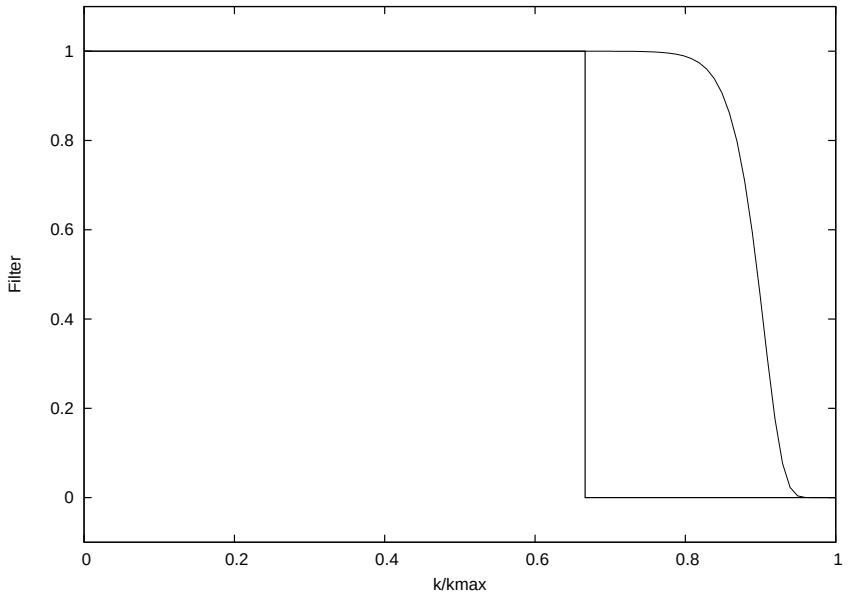
Aliasing tritt nur bei nichtlinearen Gleichungen auf

Lösung: 'Problematische' Wellenzahlen 'wegdämpfen', bevor sie durch Nichtlinearität zu groß werden

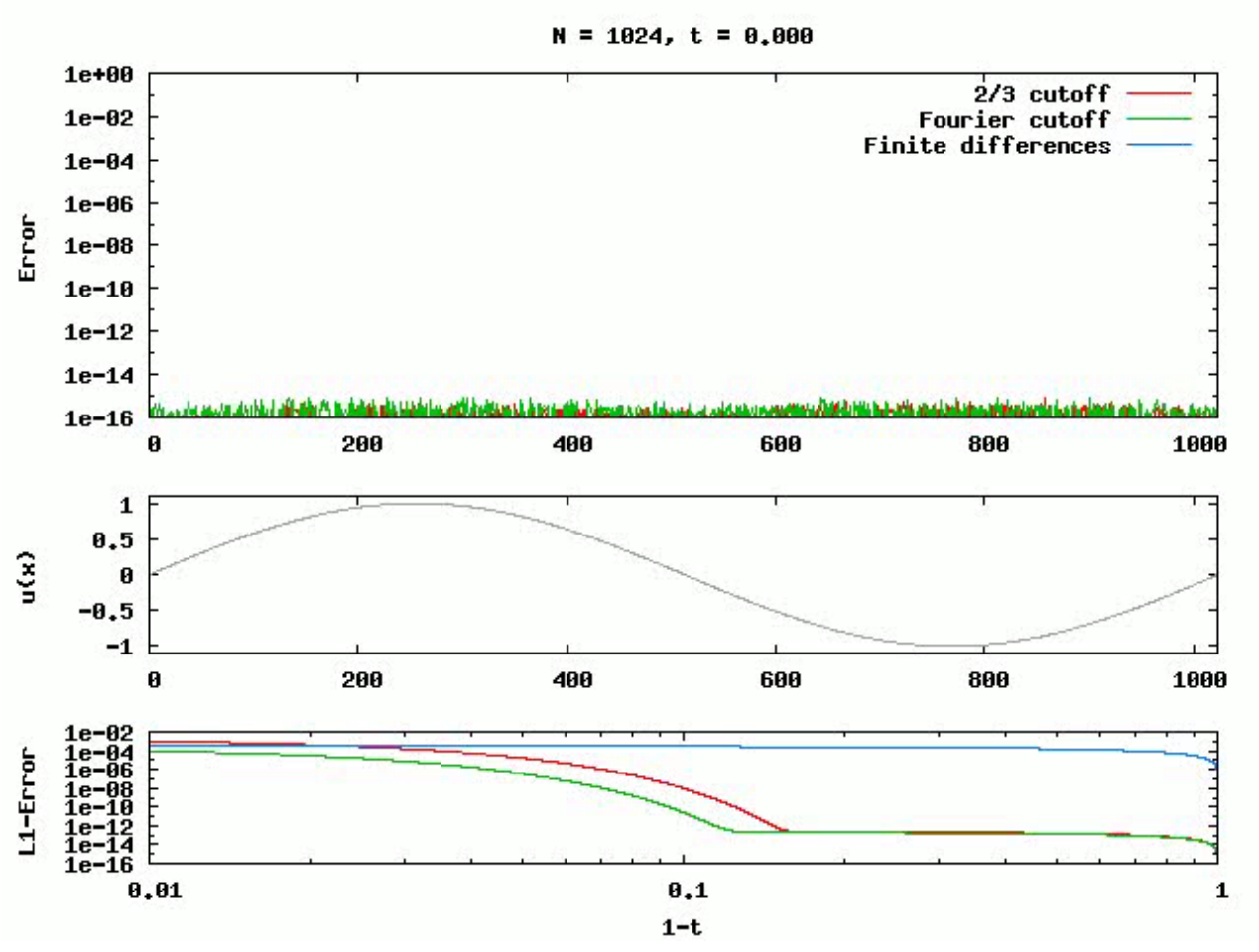
Aliasing bei DGLs

Beispiel: Konvektive Nichtlinearität $u \cdot \partial_x u$

- $2/3$-Regel von Orszag: Setze Wellenzahlen mit $k > \frac{2}{3} k_{max}$ zu null
- Glatter Fourier-Filter: Benutze $f(k) = \exp\left\{-36 \left(\frac{k}{k_{max}}\right)^{36}\right\}$ als 'Filterfunktion'



Aliasing bei DGLs



Periodizität

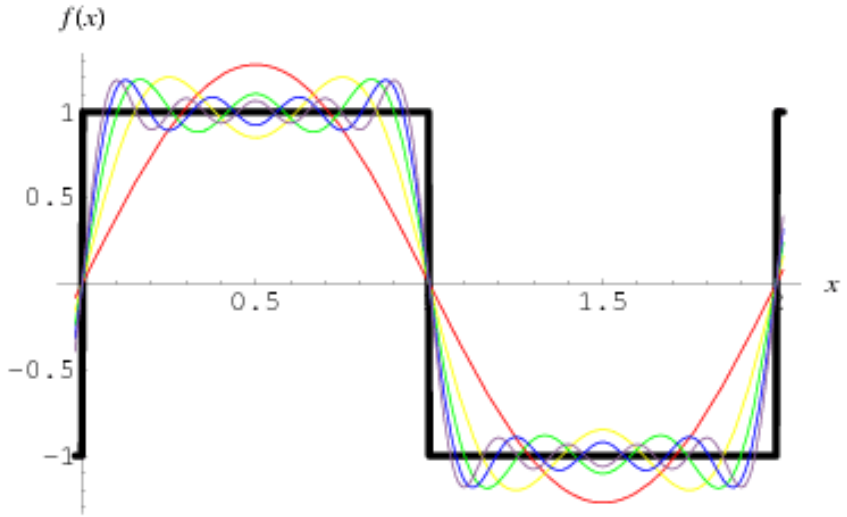
Wichtigste Einschränkung bei Benutzung von FFT: Signal und Spektrum sind periodisch!

$\rightarrow$ Für DGLs sind nur periodische Randbedingungen einfach zu handhaben

Andere Randbedingungen (z.B. Dirchlet, no-slip bei DGLs) möglich, aber schwierig zu implementieren

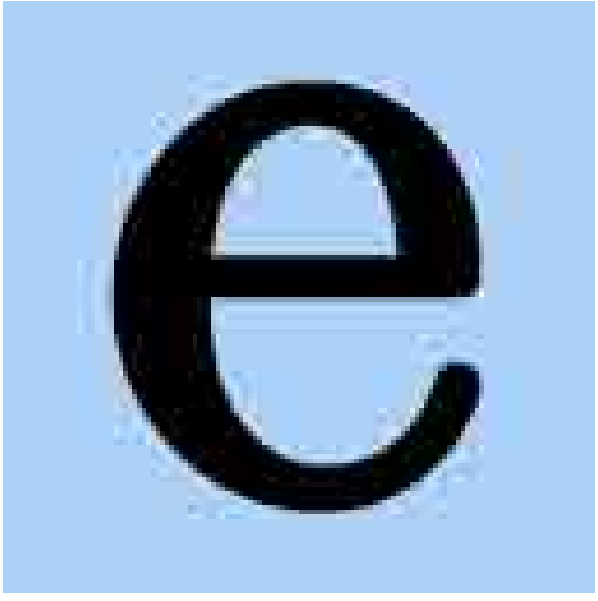
Gibb'sches Phänomen

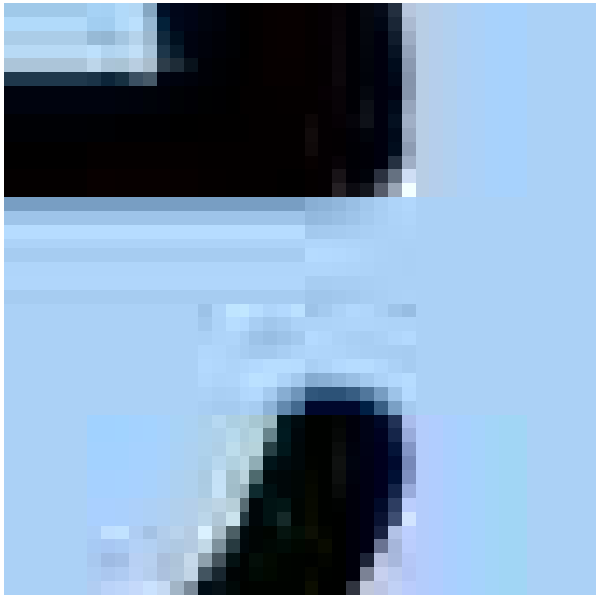
Bei Unstetigkeiten gibt es 'Überschwinger'



Gibb'sches Phänomen

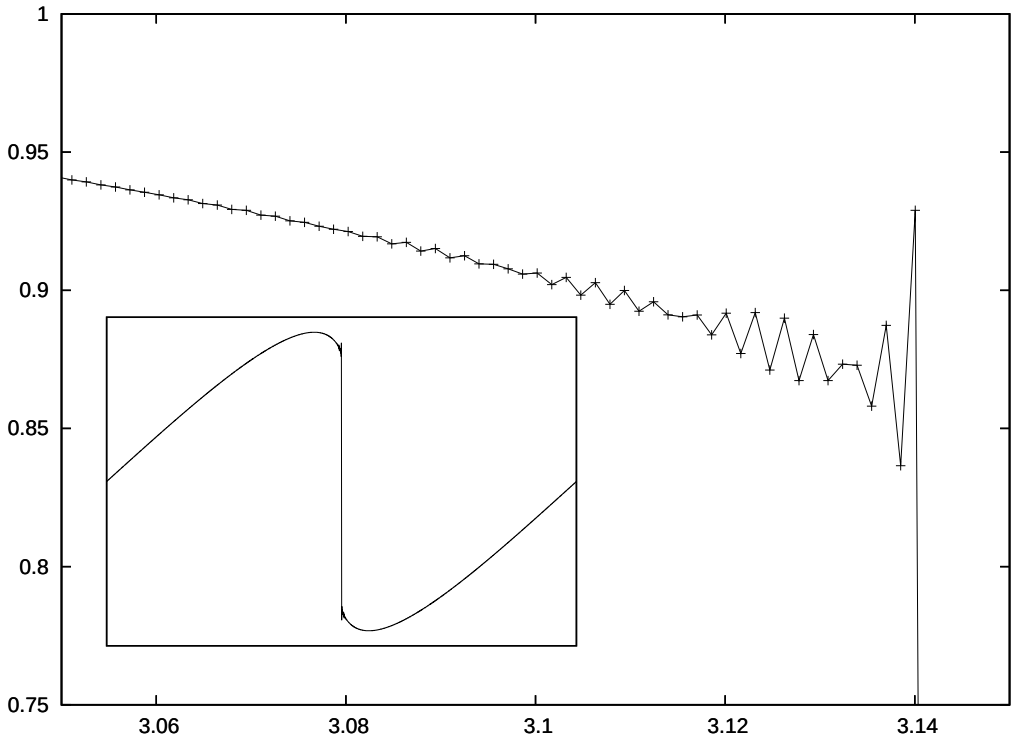
'Überschwinger' erzeugen Artefakte im JPEG-Dateiformat:





Gibb'sches Phänomen

Im Lösen von DGLs entstehen Probleme, falls sich Unstetigkeiten entwickeln:



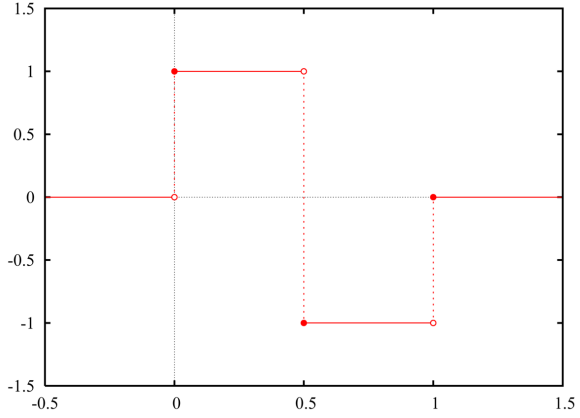
Wavelets

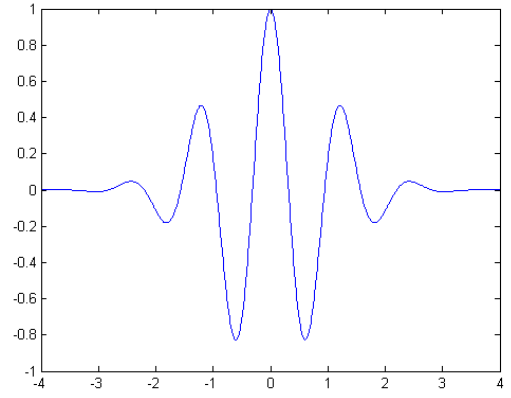
Basisfunktionen der FT sind ebene Wellen

→ 'Extreme' / Singuläre Daten können nur schlecht durch ebene Wellen approximiert werden

→ Benutze stärker lokalisierte Basisfunktionen, sog. *Wavelets*

→ Schnelle Wavelet-Transformation möglich





Zusammenfassung

- FT transformiert zwischen dem Orts- und dem Frequenzraum
- FFT ermöglicht schnelle Berechnung
- Ableitungen lassen sich durch FT berechnen
- DGLs kann man mittels FT behandeln
- Wegen der Diskretisierung wird Aliasing ein Problem
- FT beschränkt auf periodische RB, und Unstetigkeiten sind problematisch

Literatur

Beschreibung des FFT-Algorithmus:
Numerical Recipes. The Art of Scientific Computing.

Dokumentation zur *FFTW*:
http://www.fftw.org/fftw3_doc/

Paper zum Fourier-Filter:
Thomas Y. Hou, Ruo Li, *Computing nearly singular solutions using pseudo-spectral methods*

Weitere Informationen (und Bildquellen):
http://en.wikipedia.org/

Vielen Dank