Blatt 6

Frank Wübbeling

1. Juni 2006

1 Aufgabe 20

Nach Vorlesung gilt $\widehat{f}(\xi)\widehat{g}(\xi) = \widehat{f * g}(\xi)$. Wegen $\widehat{f}(\xi) = \widetilde{f}(-\xi)$ gilt damit auch $\widetilde{f}(\xi)\widetilde{g}(\xi) = (\widetilde{f * g})(\xi)$. Ersetze nun f durch $\widehat{f}$ und g durch $\widehat{g}$. Fouriertransformation liefert $\widehat{fg} = \widehat{f} * \widehat{g}$.
Wir zeigen nun, dass fg die Bandbreite 2Ω hat.

$$\begin{aligned} \widehat{fg}(\xi) &= (\widehat{f} * \widehat{g})(\xi) \\ &= \int_{\mathbf{R}} f(x)g(\xi - x)\mathrm{dx} \end{aligned}$$

f hat seinen Träger in $[-\Omega, \Omega]$, g hat seinen Träger in $[\xi - \Omega, \xi + \Omega]$. Für $|\xi| > 2\Omega$ haben die beiden Träger einen leeren Schnitt, d.h. die Bandbreite von (fg) ist $\leq 2\Omega$. Nach Vorlesung ist die Trapezregel damit exakt, wenn $h \leq \frac{2\pi}{2\Omega}$, also h die Nyquist–Bedingung erfüllt.

2 Aufgabe 21

Da f stetig und periodisch ist, konvergiert die Fourier–Reihe, und wir haben

$$f(x) = \sum \widehat{f}_k e^{ikx/A}.$$

Für $f \in L_2$ geht Teil a mit einer alten Aufgabe. Alternative Lösung für f stückweise stetig differenzierbar (dann konvergiert die Fourierreihe gleichmäßig und ist beschränkt):
(a) Sei $\varphi \in \mathcal{S}$. Sei ohne Einschränkung $A = \pi$.

$$\begin{aligned}
\widehat{f}(\varphi) &= \int_{\mathbf{R}} f(\xi)\widehat{\varphi}(\xi)d\xi \\
&= \int_{\mathbf{R}} \sum_k \widehat{f}_k e^{ik\xi}\widehat{\varphi}(\xi)d\xi \\
&= \sum_k \widehat{f}_k \int_{\mathbf{R}} e^{ik\xi}\widehat{\varphi}(\xi)d\xi \\
&= (2\pi)^{1/2} \sum_k \widehat{f}_k \varphi(k)
\end{aligned}$$

(b) Schreibe die Fourierreihe für den ersten Term auf der rechten Seite auf. Es gilt (wieder für $A = \pi$):

$$\begin{aligned}
\sum_{k=0}^{p-1} f(hk) &= \sum_{k=0}^{p-1} \sum_l \widehat{f}_l e^{iklh} \\
&= \sum_l \widehat{f}_l \sum_{k=0}^{p-1} e^{(ilh)k} \\
&= p \sum_l \widehat{f}_{lp}
\end{aligned}$$

Der Satz folgt durch Multiplikation mit $\frac{2A}{p}$ und Herausziehen des nullten Terms der Summe.

3 Aufgabe 22

$$\begin{aligned}
||G_{k,r}u||^2 &= \int_{|x|<r} \left| \int_{|y|<r} G_k(x-y)u(y)\mathrm{dy} \right|^2 \mathrm{dx} \\
&= r^9 \int_{|x|<1} \left| \int_{|y|<1} G_k(r(x-y))u(ry)\mathrm{dy} \right|^2 \mathrm{dx} \\
&= r^7 \int_{|x|<1} \left| \int_{|y|<1} G_{kr}((x-y))u_r(y)\mathrm{dy} \right|^2 \mathrm{dx} \\
&= r^7 ||G_{k,1}u_r||^2
\end{aligned}$$

Hier ist $u_r(y) = u(ry)$. Es wurde benutzt $G_k(rx) = \frac{1}{r}G_{kr}(x)$.
Andererseits gilt

$$||u_r||^2 = \int_{|x|<1} |u_r(x)|^2 \mathrm{dx} = \frac{1}{r^3} \int_{|x|<r} |u(x)|^2 \mathrm{dx} = \frac{1}{r^3} ||u||^2$$

und damit

$$||G_{k,r}|| = \sup_u \frac{||G_{k,r}u||}{||u||} = r^2 \sup_{u_r} \frac{||G_{kr,1}u_r||}{||u_r||} = r^2 ||G_{kr,1}||.$$

4 Aufgabe 23

Nach der Poissonschen Formel gilt für bandbeschränktes f, das die Nyquistbedingung erfüllt, im $\mathbb{R}^2$

$$\widehat{f}(\xi) = \frac{1}{2\pi} h^2 \sum_{k_1 \in \mathbb{Z}} \sum_{k_2 \in \mathbb{Z}} f(hk) e^{-ihk\xi}.$$

Durch zweimalige eindimensionale inverse Fouriertransformation (erst nach ξ_1, dann nach ξ_2) erhält man die Sincreihe im $\mathbb{R}^2$

$$f(x) = \sum_{k \in \mathbb{Z}^2} f(hk) \mathrm{sinc}((x - hk)\frac{\pi}{h})$$

mit der Definition $\mathrm{sinc}(x_1, \ldots, x_n) = \mathrm{sinc}(x_1) \cdots \mathrm{sinc}(x_n)$.
Diese Formel definiert eine Interpolationsformel: Gegeben $f(hk)$, kann $f(x)$ für $x \in \mathbb{R}^2$ berechnet werden. Das spezielle an dieser Formel: Sie ist exakt, falls f bandbeschränkt ist.
In der Aufgabe soll diese Interpolationsformel benutzt werden, um ein gegebenes Bild auf die p-fache Größe hochzusamplen. Klar: So, wie f definiert ist, sind die Summen endlich. Außerdem: Die Abtastbedingung kann keinesfalls eingehalten werden (da f endlichen Träger hat, hat $\widehat{f}$ keinen endlichen Träger und ist damit nicht bandbeschränkt). Wir erhalten also nur eine Näherung, keine exakte Interpolation.
Sei $h = \frac{1}{N}$. Dann werden zur naiven Auswertung von $f(x)$ $(2N+1)^2$ Rechenoperationen benötigt. Falls die Interpolationsformel auf einem Gitter mit pN Abtastpunkten ausgewertet werden soll, macht das insgesamt $(2pN+1)^2(2N+1)^2$ Rechenoperationen, also $O(N^4)$. Mit der Vorstellung, dass diese Formel zum Beispiel zur Vergrößerung eines Films benutzt werden soll (mit 50 Bildern pro Sekunde), ist das deutlich zu lang.
Die entscheidende Idee ist, die Operation als Faltung zu schreiben. Wir führen das der Einfachheit halber nur auf $\mathbb{R}$ durch, die Verallgemeinerung auf $\mathbb{R}^2$ ist einfach (und gar nicht nötig, siehe unten).
Gegeben y_p, $p = -m..m$. y soll auf p-fache Länge verlängert werden mit Hilfe der Sincreihe. Für das Resultat z_j, $j = -mp..mp$ gilt

$$z_j = \sum_{k=-m}^{m} y_k \mathrm{sinc}(j\frac{\pi}{p} - k\pi).$$

Wir wollen diese Formel als Faltung schreiben. Definiere dazu $\tilde{x} \in \mathbb{R}^{2mp+1}, \tilde{y} \in \mathbb{R}^{4mp+1}$:

$$\tilde{y}_{kp} = y_k,\ k = -m \ldots m,\ y_k = 0 \text{ sonst.}$$

$$\tilde{x}_k = \mathrm{sinc}(k\frac{\pi}{p}),\ k = -2mp \ldots 2mp.$$

Für die Faltung $z = \tilde{y} * \tilde{x}$ gilt dann für $j = -mp \dots mp$:

$$\begin{aligned} z_j &= \sum_{k=-mp}^{mp} \tilde{y}_k \tilde{x}_{j-k} \\ &= \sum_{k=-m}^{m} \tilde{y}_{pk} \tilde{x}_{j-pk} \\ &= \sum_{k=-m}^{m} y_k \mathrm{sinc}(j\pi/p - k\pi) \end{aligned}$$

oder z ist die Faltung von x und y. In zwei Dimensionen gilt der entsprechende Algorithmus: Vergrößere das Bild B durch Einschieben von $p-1$ Nullen zwischen zwei Bildpunkten, definiere eine Matrix X als Wertetafel des zweidimensionalen sinc, dann ist die Sincreihe die Faltung dieser beiden Matrizen.
Wir bemerken nun noch folgenden einfachen Satz: Für die Matrix X gelte $X_{k_1,k_2} = x_{k_1} x'_{k_2}$. Dann gilt für die Faltung Z der Matrizen Y und X:

$$\begin{aligned} Z_j &= \sum_{k_1=-N}^{N} \sum_{k_2=-N}^{N} Y_k X_{k-j} \\ &= \sum_{k_1=-N}^{N} \left(\sum_{k_2=-N}^{N} Y_k x'_{k_2-j_2} \right) x_{k_1-j_1} \end{aligned}$$

Die innere Summe ist eine Faltung auf den Zeilen von Y, die äußere Summe ist eine Faltung auf den Spalten des Ergebnisses. Zur Durchführung: Es muss auf $2N+1$ Zeilen/Spalten eine eindimensionale Faltung der Länge $2N+1$ durchgeführt werden, das kostet jedesmal offensichtlich $O(N^3)$ (naiv durchgeführt). Mit der Bemerkung, dass unser Faltungskern aus der Sincreihe die Bedingung erfüllt, haben wir die Komplexität bereits um einen Faktor N verrringert.
Führen wir nun die Faltung mit Hilfe des Faltungssatzes durch, bekommen wir eine Reduktion auf $N^2 \log N$, also fast um den Faktor N^2.
Tatsächlich lässt sich für praktische Zwecke die Komplexität noch weiter verringern. Um den Faltungssatz anzuwenden, müssen wir die Fouriertransformierte von $\tilde{y}$ berechnen. Setzt man die spezielle Struktur ein, sieht man sofort, dass die FT von $\tilde{y}$ die von y ist, geeignet periodisch fortgesetzt. Es reicht also, die Fouriertransformierte von y zu berechnen. Die Berechnung der Fouriertransformierte von $\tilde{x}$ kann man sich ganz sparen: Die analytische inverse Fouriertransformierte wäre die charakteristische Funktion eines Intervalls, punktweise Multiplikation entfernt gerade die periodischen Fortsetzungen in $\tilde{y}$. Setzen wir diese analytische Inverse hier ein, ergibt sich folgender Algorithmus zum Upsamplen eines Bildes Y:

1. Berechne die diskrete Fouriertransformierte $\widehat{Y}$ von Y.

2. Vergrößere $\widehat{Y}$ um den Faktor p durch Ergänzen mit Nullen.

3. Berechne die inverse Fouriertransformierte von $\widehat{Y}$.

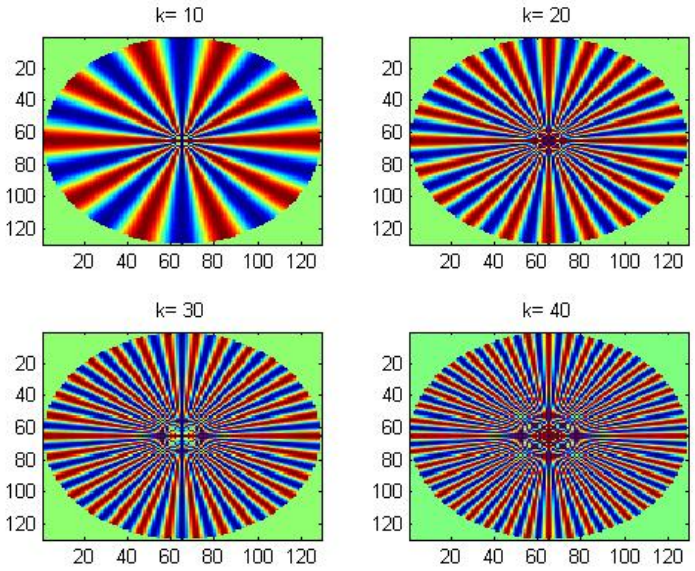
Das ist natürlich genau der Algorithmus, den man vielleicht von Anfang an vermutet hätte: Durch die Fouriertransformation lassen sich die Fourierkoeffizienten zu den Frequenzen $k = -m \dots m$ berechnen (nach Abtasttheorem). Wir samplen einfach hoch, indem wir die zugehörigen trigonometrischen Funktionen bei der inversen Fouriertransformation auf einem engeren Gitter auswerten.
Hier die zugehörigen matlab-codes. Aufgabe 23 a):

```
function aufg23a
%aufg23a Wertetafel von real(z^k/abs(z)^k)
N=64;
[x,y]=meshgrid((-N:N)/N,(-N:N)/N);
z=x+i*y;

makebild(z,10,1);
makebild(z,20,2);
makebild(z,30,3);
makebild(z,40,4);

function makebild(z,k,p)
subplot(2,2,p);
imagesc(f(z,k));
title(['k= ' num2str(k)]);

function erg=f(z,k)
erg=z.^k./(abs(z).^k);
erg(abs(z)>1)=0;
erg=real(erg);
```



Aufgabe 23a

fourconv2, Faltung zweier Matrizen mit Hilfe der Fouriertransformation. Ergebnis ist dasselbe wie bei conv2.

```
function z = fourconv2( x,y )
%Unsymmetrische Faltung von x und y (zero-padded).
m=size(x,1);
n=size(x,2);
m1=size(y,1);
n1=size(y,2);
tmp=zeros(m+m1-1,n+n1-1);
tmp(1:m,1:n)=x;
x=tmp;
tmp=zeros(m+m1-1,n+n1-1);
tmp(1:m1,1:n1)=y;
y=tmp;
z=ifft2(fft2(x).*fft2(y));
```

zweidfalt, Lösung der Aufgabe durch zweidimensionale Faltung. Upsamplen eines gegebenen Bildes um den Faktor p.

```
function z = zweidfour( bild ,p )
%Upsample bild um Faktor p.
%Annahme: bild ist quadratisch, Dimension ist ungerade.
```

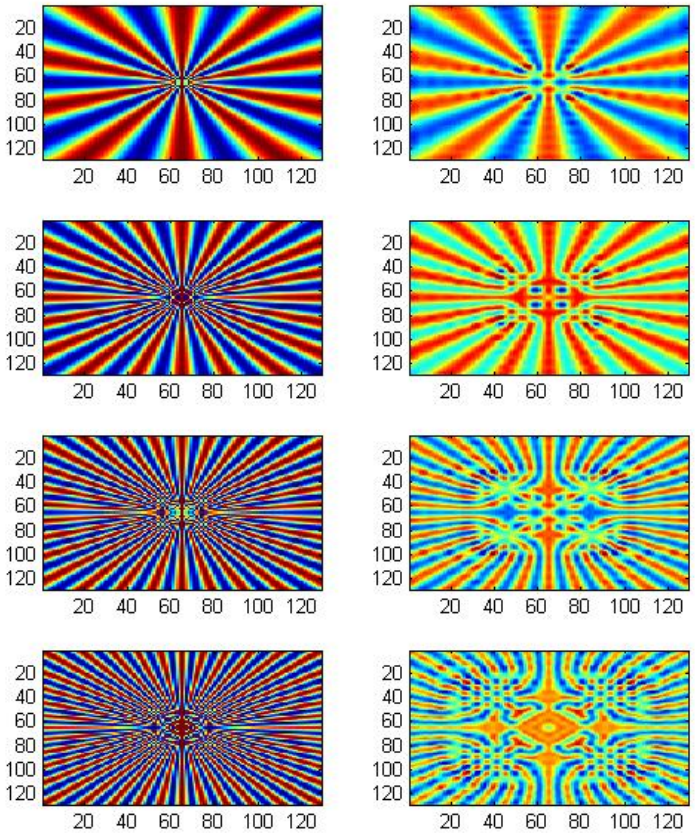
```
%TODO: Funktioniert auch für m=size(bild,2).
m=(size(bild,2)-1)/2;
n=p*m;
b=zeros(2*n+1,2*n+1);
b(1:p:2*n+1,1:p:2*n+1)=bild;
x=sinc((-2*n:2*n)*pi/p)'*ones(1,4*n+1);
z=fourconv2(b,x.*x');
z=z(2*n+1:4*n+1,2*n+1:4*n+1);
```

aufg23b, Treiber für die Lösung.

```
function erg = Untitled1( input_args )
makebild(10,1);
makebild(20,2);
makebild(30,3);
makebild(40,4);

function makebild(k,r)
N=16;
M=64;
p=M/N;
subplot(4,2,2*r-1);
x=[-N:N]'/N*ones(1,2*N+1);
y=x';
z=x+i*y;
x1=[-M:M]'/M*ones(1,2*M+1);
y1=x1';
z1=x1+i*y1;
imagesc(real(f(z1,k)));
subplot(4,2,2*r);
imagesc(real(zweidfalt(f(z,k),4)));

function erg=f(z,k)
erg=z.^k./(abs(z).^k+1e-300);
```


20 40 60 80 100 120
20 40 60 80 100 120