AUSGEWÄHLTE KAPITEL AUS "NUMERISCHE METHODEN IN DER PHYSIK"
H. Sormann SS 2007

Erstes Thema: Computertomographie

Rekonstruktion der Masseverteilung eines zweidimensionalen Objektes mittels der FFT unter Verwendung eindimensionaler Röntgenprofile

1. Einleitung

Die Fourier-Analyse spielt in vielen Gebieten der Physik, speziell der Angewandten Physik, eine überragende Rolle. Im folgenden soll das Problem der *Computertomographie* diskutiert werden, bei dem es darum geht, aus einer endlichen Zahl von eindimensionalen Profilen mittels diskreter Fourier-Transformationen zwei- oder dreidimensionale Verteilungen bestimmmter physikalischer Größen zu rekonstruieren. Solche Rekonstruktionsmethoden werden z. B. in der Festkörperphysik verwendet, um aus eindimensionalen *Comptonprofilen* die dreidimensionale Impulsverteilung von Elektronen in einem Festkörper zu gewinnen. Eine besondere Bedeutung kommt diesen Methoden in der diagnostischen Medizin zu, wo es darum geht, aus einer Reihe von zweidimensionalen Röntgenbildern dreidimensionale Darstellungen vom Inneren des menschlichen Körpers zu erhalten. Die wichtigsten mathematischen Grundlagen für diese *Computertomographie* (CT) genannte Methode wurden von *Cormack* und *Hounsfield* geschaffen, die dafür 1979 mit dem Nobelpreis für Medizin ausgezeichnet wurden.

Es gibt eine Reihe von mathematisch unterschiedlichen Rekonstruktionsmethoden, wobei die wichtigste von ihnen auf der Fourier-Transformation beruht. Da in allen im Rahmen dieser Übung enthaltenen Aufgaben die

Inputdaten als *diskrete* Datenpunkte vorliegen, wird die im folgenden verwendete Methode die *Fourier-Analyse diskreter Daten* sein.

Ein Kapitel zu diesem Thema finden Sie in meinem Vorlesungsskriptum *Numerische Methoden in der Physik*. Um Ihnen den Zugang zu diesem Thema zu erleichtern, habe ich die wichtigsten Teile dieses Kapitels an den Anfang dieser Übungsbeschreibung gestellt.

2. Grundlagen der Fourier-Analyse diskreter Daten

Gegeben sei eine Menge von n äquidistant verteilten Stützpunkten

$$(x_j = x_0 + j\Delta | y_i) \qquad j = 0, 1, \ldots, n-1 \quad , \tag{1}$$

wobei die x reellwertig sind und die y auch komplexwertig sein können. Um die Formeln zu vereinfachen, wird im folgenden $x_0 = 0$ angenommen.

Wie Sie sicher wissen, bedeutet Fourier-Transformation (FT) in der Physik gewöhnlich eine Transformation aus dem *Ortsraum* in den *Wellenzahlraum* bzw. aus dem *Zeitraum* in den *Frequenzraum*. Dementsprechend steht die hier verwendete Größe x manchmal für eine Ortskoordinate und manchmal für eine Zeitkoordinate.

Die Punktmenge (1) soll nun unter Verwendung exponentieller Basisfunktionen interpoliert werden:

$$I(x) = \frac{1}{n} \sum_{q=0}^{n-1} Y_q \, \mathrm{e}^{-i\alpha q x} \,, \tag{2}$$

wobei die Entwicklungskoeffizienten Y_q, die *Fourierkoeffizienten*, zu bestimmen sind.

Gleichzeitig soll angenommen werden, daß die n gegebenen Punkte Teil einer periodischen Funktion mit der Periode

$$P = \Delta\, n$$

sind.[1] Demzufolge muß die Interpolationsfunktion $I(x)$ ebenfalls bzgl. P periodisch sein, d.h. es muß gelten:

$$I(x + P) = I(x)\,.$$

Wie man leicht zeigen kann, ist diese Bedingung äquivalent mit einer speziellen Wahl der bisher freien (reellen) Konstanten α:

$$\frac{1}{n} \sum_{q=0}^{n-1} Y_q \, \mathrm{e}^{-i\alpha q (x+P)} = \frac{1}{n} \sum_{q=0}^{n-1} Y_q \, \mathrm{e}^{-i\alpha q x}\,.$$

[1] Es spielt dabei keine Rolle, ob die zu interpolierenden Funktionswerte *tatsächlich* eine periodische Funktion beschreiben!

Die obige Identität gilt offenbar unter der Bedingung

$$e^{-i\alpha qP} = e^{-i\alpha qn\Delta} = 1\,,$$

wodurch sich für α der Wert

$$\alpha = \frac{2\pi}{\Delta n}$$

ergibt. Damit lautet die Interpolationsfunktion

$$I(x) = \frac{1}{n}\sum_{q=0}^{n-1} Y_q \exp\left[-i\frac{2\pi qx}{\Delta n}\right]\,. \tag{3}$$

Nach der Interpolationsbedingung gilt nun

$$I(x_j) = \frac{1}{n}\sum_{q=0}^{n-1} Y_q \exp\left(-i\frac{2\pi qj}{n}\right) = y_j \quad (j = 0, 1, \ldots, n-1)\,, \tag{4}$$

und die Lösung dieses linearen, inhomogenen Gleichungssystems sind die i. a. komplexwertigen Fourierkoeffizienten Y_q.

Selbstverständlich könnten die Y_q direkt aus dem Gleichungssystem (4) ermittelt werden. Im Falle der Fourier-Entwicklung geht das aber sehr viel einfacher. Multipliziert man nämlich die Glg. (4) mit $\exp(i2\pi q'j/n)$ und summiert über den Index j, so ergibt sich

$$\sum_{j=0}^{n-1} y_j \exp\left(i\frac{2\pi q'j}{n}\right) = \frac{1}{n}\sum_{q=0}^{n-1} Y_q \underbrace{\sum_{j=0}^{n-1} \exp\left(i\frac{2\pi j}{n}(q'-q)\right)}_{(A)}\,. \tag{5}$$

o.B.: der Ausdruck (A) hat im Falle äquidistanter Stützpunkte das einfache Ergebnis

$$(A) = n \cdot \delta_{q,q'} \qquad \text{'Kronecker-Delta': } 1 \quad \text{für} \quad q = q' \quad \text{sonst Null.}$$

Die Ursache dafür ist, daß die Fourier-Entwicklungsfunktionen bzgl. einer äquidistanten Punktmenge *orthogonal* sind.

Somit erhält man aus Glg. (5) unmittelbar

$$\sum_{j=0}^{n-1} y_j \exp\left(i\frac{2\pi q'j}{n}\right) = \frac{1}{n}\sum_{q=0}^{n-1} Y_q\, n\, \delta_{q,q'} = Y_{q'}\,.$$

Es ergibt sich somit für die *Fourierkoeffizienten* Y_q die einfache Bestimmungsgleichung

$$Y_q = \sum_{j=0}^{n-1} y_j \exp\left(i\frac{2\pi qj}{n}\right) \qquad (q = 0, 1, \ldots, n-1)\,. \tag{6}$$

Man nennt die Auswertung (6), also die Berechnung der Y_q aus den y_j, die diskrete Fourier-Transformation (DFT) der y_j; die Auswertung in die umgekehrte Richtung, also die Berechnung der y_j aus den Y_q, heißt dementsprechend die inverse DFT.

2.1 Numerische Berechnung der Fourierkoeffizienten

Die numerische Auswertung von (6) ist ein Prozess der Ordnung n^2, d.h. es sind n Summenterme für jeden der n Koeffizienten zu berechnen. Dies bedeutet für große n einen sehr zeitintensiven Rechenprozess.

In bezug auf die Rechenzeit bietet die sogenannte *Fast Fourier Transform FFT*, die auf einer rekursiven Berechnung der in (6) vorkommenden Summe beruht, einen wesentlichen Fortschritt. Dieser Algorithmus, der auf den Ergebnissen einer Arbeit von Danielson und Lanczos (1942) beruht, ist zwar im Prinzip einfach, seine geschickte Programmierung ist jedoch im Detail etwas trickreich. Er soll deshalb hier nur prinzipiell erläutert werden, und zwar für das Beispiel von $n = 8$ Stützpunkten. Spaltet man die Summe (6) in zwei Teilsummen auf, und zwar in der Form

$$Y_q = \sum_{j=0}^{7} y_j e^{i\cdot 2\pi q j/8} = \sum_{j=0(2)}^{6} y_j e^{i\cdot 2\pi q j/8} + \sum_{j=1(2)}^{7} y_j e^{i\cdot 2\pi q j/8} \quad ,$$

so erhält man durch geeignete Transformation der Summenindizes den Ausdruck

$$Y_q = \sum_{j=0}^{3} y_{2j} e^{i\cdot 2\pi q j/4} + e^{i\cdot 2\pi q/8} \sum_{j=0}^{3} y_{2j+1} e^{i\cdot 2\pi q j/4}$$

bzw. unter Verwendung der Abkürzung $W_m = \exp(i \cdot 2\pi/m)$

$$Y_q = \sum_{j=0}^{7} W_8^{j\cdot q} y_j = \underbrace{\sum_{j=0}^{3} W_4^{j\cdot q} y_{2j}}_{Y_q^0} + W_8^k \underbrace{\sum_{j=0}^{3} W_4^{j\cdot q} y_{2j+1}}_{Y_q^1}$$

Die beiden Teilsummen Y_q^0 und Y_q^1 können nun ihrerseits wieder zerlegt werden:

$$Y_q^0 = \underbrace{\sum_{j=0}^{1} W_2^{j\cdot q} y_{4j}}_{Y_q^{00}} + W_4^q \underbrace{\sum_{j=0}^{1} W_2^{j\cdot q} y_{4j+2}}_{Y_q^{01}}$$

$$Y_q^1 = \underbrace{\sum_{j=0}^{1} W_2^{j\cdot q} y_{4j+1}}_{Y_q^{10}} + W_4^q \underbrace{\sum_{j=0}^{1} W_2^{j\cdot q} y_{4j+3}}_{Y_q^{11}}$$

Die letzte Aufspaltung der vier Größen Y_q^{00}, Y_q^{01}, Y_q^{10} und Y_q^{11} lautet:

$$Y_q^{00} = y_0 + W_2^q y_4 \qquad Y_q^{01} = y_2 + W_2^q y_6$$

$$Y_q^{10} = y_1 + W_2^q y_5 \qquad Y_q^{11} = y_3 + W_2^q y_7$$

Es sind somit die Fourierkoeffizienten 'auf die Ebene der y-Werte reduziert'. Davon ausgehend, kann der gewünschte Koeffizient Y_q schrittweise aufgebaut werden, wie dies schematisch in der Abbildung 1 dargestellt ist.
Die hier präsentierte Vorgangsweise funktioniert jedoch nur dann, wenn die bei dem Aufbau der Koeffizienten gemäß Abb. 1 berücksichtigten Stützpunkte von Schritt zu Schritt verdoppelt werden können, d.h. *wenn die Stützpunktanzahl n eine Potenz von 2 ist* [2].

[2] Diese Bedingung ist in einigen modernen FFT-Programmen *nicht mehr* gegeben.

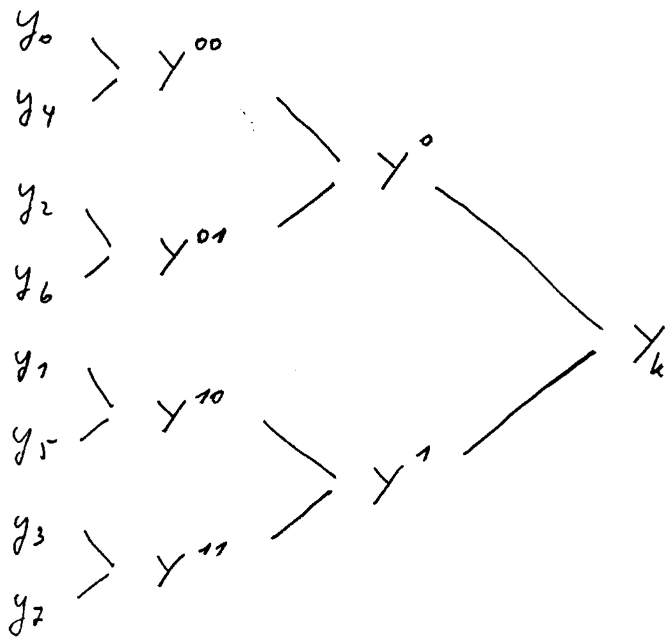


Figure 1: Grundprinzip der 'Fast Fourier Transform'.

Wie ebenfalls aus Abb. 1 hervorgeht, müssen die gegebenen y_j-Werte ($j = 0, \ldots, n-1$) am Beginn der Kaskade (links in der Abb.) 'geeignet' geordnet werden. Es zeigt sich, daß diese Indexordnung sich einfach aus einer *Bitumkehr* der ursprünglichen Indizes in dualer Schreibweise ergibt:

ursprüngliche Folge (Index)	Dual	BITUMKEHR	neue Folge (Index)
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Der große Vorteil der FFT gegenüber der direkten Auswertung von (6) liegt in der beträchtlichen Einsparung an Rechenzeit.

<u>o.B.:</u> Während nämlich bei der Berechnung der Fourierkoeffizienten gemäß (6) die Auswertung von n^2 Summentermen nötig ist, reduziert sich diese Zahl bei der FFT auf $n \ln_2(n)$:

n	n^2	$n\ln_2(n)$
128	16384	896
1024	1048576	10240
.	.	.
.	.	.
1048576	$\approx 10^{12}$	$\approx 2.1 \cdot 10^7$

Bei sehr großen Datenmengen ist eine Fourier-Transformation mittels kleiner Computer (PC) überhaupt nur mittels der FFT möglich!

2.2 FFT-Programme in C, F90 und MATLAB

FFT-Programme sind häufig recht raffiniert programmiert. Es erscheint im Rahmen dieser LV nicht sinnvoll, auf alle Details der Algorithmen einzugehen. Aus diesem Grund werden im folgenden (ausnahmsweise) keine Struktogramme, sondern direkt FFT-Programmlistings in den Sprachen C, F90 und MATLAB präsentiert. Es handelt sich dabei um Programme zur schnellen Fourier-Transformation einer diskreten komplexwertigen Punktmenge mit 2^m Werten (m = positive Integerzahl) und mit äquidistanten x-Komponenten.

Das C-Programm FOUR1

Quelle: Press et al., *Numerical Recipes in C*, Cambtidge Uni Press 1992, S. 507f.

INPUT-Parameter:

DATA( ): eindimensionales Real-Feld, das die Real- und Imaginärteile der Größen enthält, die Fourier-transformiert werden sollen.

Die DATA-Werte müssen wie folgt abgespeichert sein:

```
RT{y0}        DATA(1)
IT{y0}        DATA(2)

RT{y1}        DATA(3)
IT{y1}        DATA(4)
   .
   .
RT{y(n-1)}    DATA(2n-1)
IT{y(n-1)}    DATA(2n)
```

DATA muß also mindestens $2n$ Werte aufnehmen können!

NN: Anzahl der n (i. a. komplexen) Werte.

ISIGN=+1: Fourier-Transformation.

ISIGN=−1: inverse Fourier-Transformation (aber ohne den Normierungsfaktor $1/n$).

OUTPUT-Parameter:

DATA( ): Real-Feld mit den Real- und Imaginärteilen der Fourier-Transformierten.

```
#include <math.h>
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

void four1(double data[], unsigned long nn, int isign)
{
unsigned long n,mmax,m,j,istep,i;
double wtemp,wr,wpr,wpi,wi,theta;
double tempr,tempi;

n=nn << 1;
j=1;
for (i=1;i<n;i+=2) {
if (j > i) {
SWAP(data[j],data[i]);
SWAP(data[j+1],data[i+1]);
}
m=n >> 1;
while (m >= 2 && j > m) {
j -= m;
m >>= 1;
}
j += m;
}
mmax=2;
while (n > mmax) {
istep=mmax << 1;
theta=isign*(6.28318530717959/mmax);
wtemp=sin(0.5*theta);
wpr = -2.0*wtemp*wtemp;
wpi=sin(theta);
wr=1.0;
wi=0.0;
for (m=1;m<mmax;m+=2) {
for (i=m;i<=n;i+=istep) {
j=i+mmax;
tempr=wr*data[j]-wi*data[j+1];
tempi=wr*data[j+1]+wi*data[j];
```

```
data[j]=data[i]-tempr;
data[j+1]=data[i+1]-tempi;
data[i] += tempr;
data[i+1] += tempi;
}
wr=(wtemp=wr)*wpr-wi*wpi+wr;
wi=wi*wpr+wtemp*wpi+wi;
}
mmax=istep;
}
}
#undef SWAP
/* (C) Copr. 1986-92 Numerical Recipes Software +. */
```

Das F90-Programm FFT

Quelle: DeVries, *Computerphysik*, Spektrum Akademischer Verlag 1995, S. 322f.

INPUT-Parameter:

A(): eindimensionables Datenfeld vom Typ

`COMPLEX(KIND=KIND(0.0d0))` ,

das die komplexwertigen Größen enthält, die Fourier-transformiert werden sollen.

M: $2 * *M$ ist die Zahl n der (äquidistanten) Datenwerte.

INV=+1: Fourier-Transformation.

INV$\neq -1$: inverse Fourier-Transformation.

OUTPUT-Parameter:

A( ): COMPLEX-Feld mit den Real- und Imaginärteilen der Fourier-Transformierten.

```
      SUBROUTINE fft(a,m,inv)
! Paul L. DeVries, Department of Physics, Miami University

! This subroutine performs the Fast Fourier Transform by
! the method of Cooley and Tukey

!        start:                 1965
!  last modified:               1993
!  Modifications H. Sormann:    2000
!
```

```
!  Parameter description by H. Sormann 2000
!  PARAMETERS:   a     vector of complex data which are to be
!                      Fourier-transformed
!                      this vector is VARIABLY dimensioned (see below)

!                      On return, a  contains the FT coefficients.
!
!                m     2**m  is the number of data points
!
!                inv equal 1        Fourier transform
!                inv not equal 1    INVERSE Fourier transform
!
      IMPLICIT NONE


      INTEGER                      :: m,inv,n,nd2,i,j,k,l,le,le1,ip
      COMPLEX(KIND=KIND(0.0d0))              :: u,w,t
      COMPLEX(KIND=KIND(0.0d0)),DIMENSION(*) :: a   ! * = var. dimension
      DOUBLE PRECISION             :: ang,pi
!      PARAMETER (pi=3.141592653589793d0)
      pi=4.d0*atan(1.d0)

      n=2**m
      nd2=n/2
      j=1
      DO i=1,n-1
        IF(i .lt. j)THEN
          t=a(j)
          a(j)=a(i)
          a(i)=t
        ENDIF
        k=nd2
100     IF(k .lt. j)THEN
          j=j-k
          k=k/2
          goto 100
        ENDIF
        j=j+k
      END DO
      le=1
      DO l=1,m
        le1=le
        le=le+le
        u=(1.d0,0.d0)
        ang=pi/dble(le1)
        w=cmplx(cos(ang), -sin(ang),KIND(0.d0))
        IF(inv .eq. 1)w=conjg(w)
```

```
    DO j=1,le1
      DO i=j,n,le
        ip=i+le1
        t=a(ip)*u
        a(ip)=a(i)-t
        a(i)=a(i)+t
      END DO
      u=u*w
    END DO
  END DO
  IF(inv .ne. 1)THEN
    DO i=1,n
      a(i)=a(i)/dble(n)
    END DO
  ENDIF
  END SUBROUTINE fft
```

Das MATLAB-Programm fft

Es gibt eine interne MATLAB-Routine mit dem Funktionsnamen *fft.m* für die DFT und eine MATLAB-Routine mit dem Namen *ifft.m* für die inverse DFT.

Wenn Sie sich z. B. mittels *help fft* über diese Routine informieren, erhalten Sie den folgenden Text:

```
%FFT Discrete Fourier transform.
%   FFT(X) is the discrete Fourier transform (DFT) of vector X.  If the
%   length of X is a power of two, a fast radix-2 fast-Fourier
%   transform algorithm is used.  If the length of X is not a
%   power of two, a slower non-power-of-two algorithm is employed.
%   For matrices, the FFT operation is applied to each column.
%   For N-D arrays, the FFT operation operates on the first
%   non-singleton dimension.
%
%   FFT(X,N) is the N-point FFT, padded with zeros if X has less
%   than N points and truncated if it has more.
%
%   FFT(X,[],DIM) or FFT(X,N,DIM) applies the FFT operation across the
%   dimension DIM.
%
%   For length N input vector x, the DFT is a length N vector X,
%   with elements
%                    N
%      X(k) =       sum  x(n)*exp(-j*2*pi*(k-1)*(n-1)/N), 1 <= k <= N.
%                   n=1
%   The inverse DFT (computed by IFFT) is given by
%                    N
```

```
%      x(n) = (1/N) sum  X(k)*exp( j*2*pi*(k-1)*(n-1)/N), 1 <= n <= N.
%                    k=1
%
%   The relationship between the DFT and the Fourier coeffs a and b in
%            N/2
%   x(n)=a0+ sum a(k)*cos(2*pi*k*t(n)/(N*dt))+b(k)*sin(2*pi*k*t(n)/(N*dt))
%            k=1
%   is
%      a0 = X(1)/N, a(k) = 2*real(X(k+1))/N, b(k) = -2*imag(X(k+1))/N,
%   x is a length N discrete signal sampled at times t with spacing dt.
%
%   See also IFFT, FFT2, IFFT2, FFTSHIFT.

%   Copyright (c) 1984-98 by The MathWorks, Inc.
%   $Revision: 5.9 $  $Date: 1998/06/30 18:43:41 $

%   Built-in function.
```

Ein MATLAB-Programm, welches das Testproblem im Abschnitt 2.3 lösen soll, könnte etwa so aussehen:

```
%Eingabe der Testdaten von Skriptum:
n=8;
delta=1.0;
period=n*delta;

index=0:1:n-1;

werte=[0.7013+0.0437i -0.0724+0.5133i 0.0988-0.2688i 0.0715-0.1162i ...
       0.4013+0.1188i -0.0901-0.1408i -0.1263-0.0688i 0.2660-0.3813i];


yy=fft(werte,n);
% Die folgende Umordnung der von fft berechneten Koeff. ist noetig,
% um dieselbe Reihenfolge wie bei den C- und F90-Programmen zu
% erhalten (Sormann 2000):
daten(1)=yy(1);
for ind=2:n
  daten(ind)=yy(n+2-ind);
end
.
.
.
```

2.3 Ein Test für die FFT-Programme

Dieses Beispiel beschreibt die Fourier-Transformation der folgenden Testdaten ($n = 8$, $\Delta = 1.$):

j	RT{y(j)}	IT{y(j)}
0	0.7013	0.0437
1	-0.0724	0.5133
2	0.0988	-0.2688
3	0.0715	-0.1162
4	0.4013	0.1188
5	-0.0901	-0.1408
6	-0.1263	-0.0688
7	0.2660	-0.3813

Die gegebenen Stützstellen repräsentieren somit eine periodische Funktion mit der Periode $P = 8$, und das Programm FOUR1 liefert die folgenden Fourierkoeffienten:

q	RT{Y(q)}	IT{Y(q)}
0	1.2501	-0.3001
1	0.0001	0.3000
2	0.2601	0.0001
3	-0.7000	-0.7003
4	0.9001	-0.0501
5	0.9999	0.0000
6	2.0001	1.0001
7	0.9000	0.0999

2.4 Frequenz- bzw. Wellenzahl-Analyse

Die folgenden Ausführungen beschreiben die sogenannte *Frequenz-Optimierung* einer Fourier-Interpolation. Sämtliche Gleichungen und Ergebnisse dieses Abschnittes gelten auch für eine *Wellenzahl-Optimierung*. Dazu müssen Sie nur die Zeitfunktionen $f(t)$ und $I(t)$ durch die entsorechenden Ortsfunktionen $f(x)$ und $I(x)$ ersetzen, und an die Stelle der diskreten Frequenzen ν_q treten die diskreten Wellenzahlen k_q.

Ausgangspunkt ist die Interpolationskurve (3)

$$I(t) = \frac{1}{n} \sum_{q=0}^{n-1} Y_q \exp\left[-i\frac{2\pi qt}{\Delta n}\right] .$$

Weiters soll angenommen werden, daß nicht nur die gegebenen (gemessenen) Werte y_j, sondern die gesamte Funktion $f(t)$, welche 'hinter diesen Werten steht', reell ist. In diesen Fall interessiert nur der Realteil der Interpolationsfunktion, nämlich

$$\Re\{I(t)\} = \sum_{q=0}^{n-1} \left[\Re\left\{\frac{Y_q}{n}\right\}\cos\left(\frac{2\pi qt}{P}\right) + \Im\left\{\frac{Y_q}{n}\right\}\sin\left(\frac{2\pi qt}{P}\right)\right] . \tag{7}$$

Man kann nun sagen: die Funktion $f(t)$ besteht aus *cosinus*-Anteilen mit den Amplituden 'Realteil von Y_q/n' und aus *sinus*-Anteilen mit den Amplituden 'Imaginärteil von Y_q/n'.
Die Frequenzen, die in die obige Entwicklung einbezogen werden, lauten

$$\nu_q = \frac{q}{P} \qquad \text{mit} \quad q = 0, \ldots, n-1$$

mit n als Zahl der Meßpunkte und P als Periode des Meß-Signals.

Es stellt sich nun die Frage, ob die Interpolation Glg. (7) die bestmögliche ist, d.h., ob diese Funktion die Meßwerte wirklich so glatt als möglich verbindet (vgl. Abschnitt 3.1).
Die Antwort ist <u>nein</u>, und die Begründung dafür finden Sie in der Abb. 2. Wegen der Periodizität der Interpolationsfunktion sind auch die Fourierkoeffizienten Y_q periodisch, und zwar mit der Periode n, d.h. es gilt:

$$Y_q = Y_{q+\mu n} \tag{8}$$

für jede beliebige ganze Zahl μ.

Man kann nun zeigen, daß *jede zusammenhängende Gruppe von n Fourierkoeffizienten mit den zugehörigen Frequenzen eine Interpolationsfunktion durch die gegebenen Punkte ergibt*, egal in welchem Bereich der Frequenzachse diese Gruppe liegt. Natürlich erhält man dabei jedesmal eine verschiedene Interpolationsfunktion!

Wenn man nun fragt, welche der möglichen Interpolationen diejenige sein wird, welche die geringste Neigung zur Oszillation zwischen den Stützpunkten hat, so ist die Antwort einfach: natürlich jene, welche die (dem Betrag nach) kleinsten Frequenzen enthält. Wenn man also n Koeffizienten benötigt, wird der ideale Frequenzbereich "um die Frequenz Null" liegen, also im Bereich $-n/2 \leq q < +n/2$.
Um diese ideale Interpolation formelmäßig darzustellen, bedarf es nur einer einfachen Umformung der Glg. (7):

$$\begin{aligned}\Re\{I(t)\} = & \sum_{q=-n/2}^{-1} \left[\Re\left\{\frac{Y_q}{n}\right\}\cos\left(\frac{2\pi qt}{P}\right) + \Im\left\{\frac{Y_q}{n}\right\}\sin\left(\frac{2\pi qt}{P}\right)\right] + \\ & + \sum_{q=0}^{(n/2)-1} \left[\Re\left\{\frac{Y_q}{n}\right\}\cos\left(\frac{2\pi qt}{P}\right) + \Im\left\{\frac{Y_q}{n}\right\}\sin\left(\frac{2\pi qt}{P}\right)\right] .\end{aligned}$$

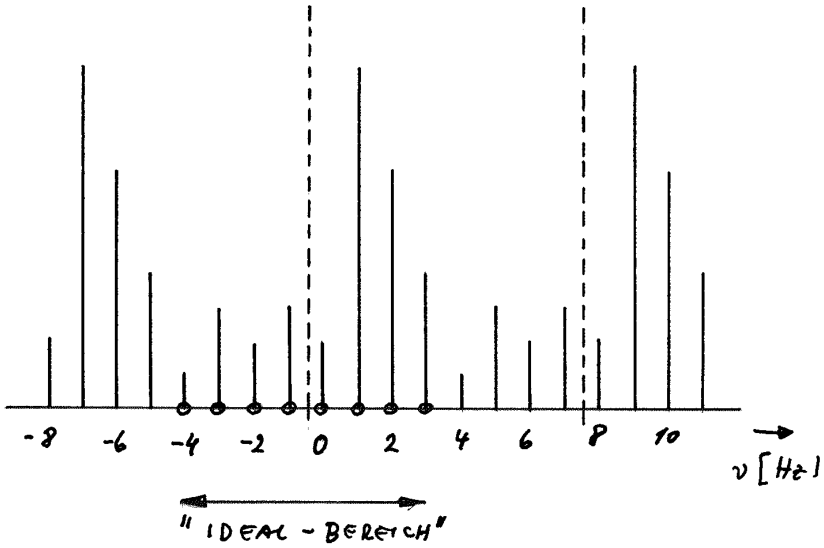


Figure 2: Periodizität der Fourierkoeffizienten im q=Raum für n=8. Es sind zwei Bereiche markiert: 'FOUR1' bedeutet den Bereich der Fourierkoeffizienten, den das Programm FOUR1 liefert (q=0...7), und 'IDEAL' bedeutet den Bereich q=–4,–3,–2,–1,0,1,2,3, der die 'glattest-mögliche' Interpolationsfunktion ergibt.

Führt man nun im ersten Term der obigen Gleichung die Index-Transformation

$$q' = n + q$$

durch, so ergibt sich weiter

$$\Re\{I(t)\} = \sum_{q'=n/2}^{n-1} \left[\Re\left\{\frac{Y_{q'-n}}{n}\right\} \cos\left(\frac{2\pi(q'-n)t}{P}\right) + \Im\left\{\frac{Y_{q'-n}}{n}\right\} \sin\left(\frac{2\pi(q'-n)t}{P}\right) \right] + \\ + \sum_{q=0}^{(n/2)-1} \left[\Re\left\{\frac{Y_q}{n}\right\} \cos\left(\frac{2\pi qt}{P}\right) + \Im\left\{\frac{Y_q}{n}\right\} \sin\left(\frac{2\pi qt}{P}\right) \right] .$$

Unter Berücksichtigung der Periodizität der Y_q (8) gilt

$$Y_{q'-n} = Y_{q'} ,$$

und man erhält (mit $q' \to q$) das Ergebnis

$$\Re\{I(t)\}_{\text{ideal}} = \sum_{q=0}^{n-1} \left[\Re\left\{\frac{Y_q}{n}\right\} \cos(2\pi\nu_q t) + \Im\left\{\frac{Y_q}{n}\right\} \sin(2\pi\nu_q t) \right] \tag{9}$$

mit

$$\nu_q = \frac{q}{P} \qquad \text{für} \quad q = 0, \ldots, \frac{n}{2} - 1 \tag{10}$$

und

$$\nu_q = \frac{q-n}{P} \qquad \text{für} \quad q = \frac{n}{2}, \ldots, n-1 \quad , \tag{11}$$

wobei die beiden Gleichungen (10) und (11) im engeren Sinne als Frequenz-Optimierung bezeichnet werden.

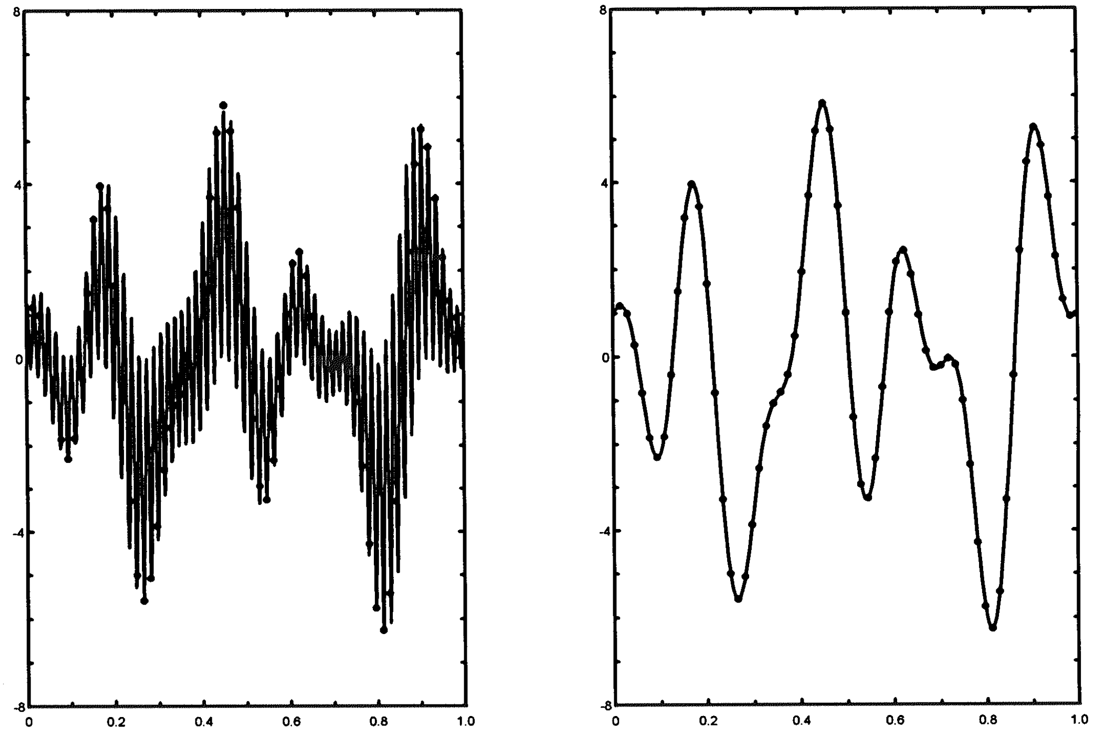


Figure 3: (a) FT-Frequenz-Analyse ohne Frequenz-Optimierung, (b) FT-Frequenz-Analyse mit Frequenz-Optimierung. Die Kreise sind die Stützpunkte.

Die Darstellung (9)-(11) hat den Vorteil, daß in ihr nur die Y_q-Werte vorkommen, die von den im Abschnitt 2.2 beschriebenen Programmen geliefert werden!

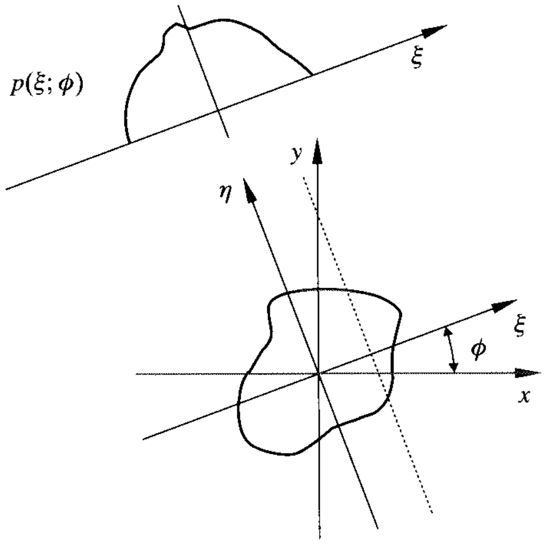


Figure 4: CT-Geometrie: ein Projektions-Profil $p(\xi;\phi)$ des Objekts entsteht, wenn rechtwinkelig zu einer ξ-Achse, die gegenüber der festen x-Achse einen Winkel ϕ einschließt, eine Reihe paralleler Röntgenstrahlbündel das Objekt in η-Richtung durchdringen.

3. Theoretische Grundlagen der Computertomographie

Die folgenden, speziell die CT betreffenden Ausführungen stammen im wesentlichen aus dem Buch von P. L. DeVries, *Computerphysik*, Spektrum Akademischer Verlag, Heidelberg, 1995, Kap. 6.11.

Das Prinzip der CT-Geometrie ist in der Abb. 4 dargestellt. In der xy-Ebene liege ein Körper mit der Masseverteilung $f(x,y)$. Dieser Körper wird nun von einem Röntgenstrahlbündel durchstrahlt (gepunktete Linie in Abb. 4). Die Position bzw. die Richtung des Röntgenstrahls werden durch den Winkel ϕ sowie durch den Minimalabstand ξ des Strahls vom Koordinatenursprung beschrieben. Beim Durchgang durch den Körper wird der Röntgenstrahl geschwächt, d.h. er erfährt eine Reduktion seiner ursprünglichen Intensität I_0 auf die Intensität I:

$$I = I_0 \, \mathrm{e}^{-\int d\eta \, f(x,y)} \, ,$$

wobei $d\eta$ ein Element des Weges ist, den der Strahl durch den Körper nimmt. Man nennt den (negativen) Logarithmus des Verhältnisses I/I_0,

$$p(\xi;\phi) = -\ln\left(\frac{I}{I_0}\right) = \int d\eta \, f(x,y) \, , \tag{12}$$

die *Projektion* p bei ξ und ϕ. Wenn man nun bei festem ϕ viele parallele Strahlenbündel mit verschiedenen, äquidistant im Intervall $[-\xi_{max}, +\xi_{max}]$ verteilten ξ verwendet, erhält man ein aus diskreten Punkten bestehendes, eindimensionales *Projektions-Profil* des Objekts für den Winkel ϕ entlang der ξ-Achse.

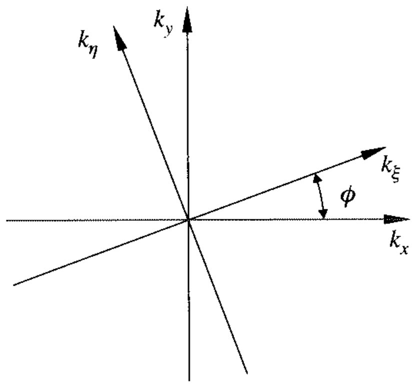


Figure 5: Die mit Abb. 4 korrespondierenden Koordinatensysteme im Wellenzahlraum.

Unser Ziel ist es, aus einer Reihe von Profilen $p(\xi;\phi)$, die bei verschiedenen Winkeln ϕ erhalten wurden, die zweidimensionale Masseverteilung (auch *Dämpfungsfunktion* genannt) $f(x,y)$ zu rekonstruieren.

Wir haben also zwei Koordinatensysteme: das fest mit dem *gescannten* Objekt verbundene System $(x;y)$ und das um den Winkel ϕ rotierende System $(\xi;\eta)$. Der Zusammenhang zwischen diesen Systemen ist einfach

$$\xi = x\cos\phi + y\sin\phi \qquad \text{und} \qquad \eta = -x\sin\phi + y\cos\phi\,. \tag{13}$$

Der erste Schritt der Rechnung ist eine Fourier-Transformation (FT) der Dämpfungsfunktion aus dem $(x;y)$-Raum ("Ortsraum") in den $(k_x;k_y)$-Raum ("Wellenzahl-Raum"):

$$F(k_x,k_y) = \frac{1}{2\pi}\int\int_{-\infty}^{+\infty} dxdy\, f(x,y)\, \mathrm{e}^{i(k_x x + k_y y)}\,. \tag{14}$$

Genau wie man aus dem $(x;y)$-System durch Drehung um ϕ entgegen dem Uhrzeigersinn in das (ξ,η)-System gelangt, kann man aus dem (k_x,k_y)-System in das $(k_\xi;k_\eta)$-System übergehen:

$$k_\xi = k_x\cos\phi + k_y\sin\phi \qquad \text{und} \qquad k_\eta = -k_x\sin\phi + k_y\cos\phi\,. \tag{15}$$

Man kann sehr leicht zeigen, daß der in der FT [Glg. (14)] vorkommende Exponentialterm die Eigenschaft

$$\mathrm{e}^{i(k_x x + k_y y)} = \mathrm{e}^{i(k_\xi \xi + k_\eta \eta)} \tag{16}$$

hat.

Nun kommt eine Festlegung: wir berechnen die FT von $f(x,y)$ nicht für den gesamten $(k_x;k_y)$-Raum, sondern nur entlang der k_ξ-Achse (also für $k_\eta = 0$), d. h. für die Argumente

$$k_x = k_\xi\cos\phi \qquad \text{und} \qquad k_y = k_\xi\sin\phi\ :$$

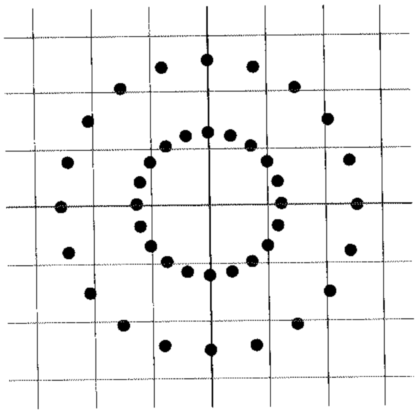


Figure 6: Verteilung der mittels Glg. (18) erhaltenen Fourierkoeffizienten der Dämpfungsfunktion im $(k_x; k_y)$-Raum.

$$F(k_\xi \cos\phi, k_\xi \sin\phi) = \frac{1}{2\pi} \int\int_{-\infty}^{+\infty} dxdy\, f(x,y)\, \mathrm{e}^{ik_\xi \xi}\,.$$

Dies kann unter Berücksichtigung von $dxdy = d\xi d\eta$ umgeformt werden in

$$F(k_\xi \cos\phi, k_\xi \sin\phi) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} d\xi \left(\int_{-\infty}^{+\infty} d\eta\, f(x,y) \right) \mathrm{e}^{ik_\xi \xi}\,. \tag{17}$$

Durch einen Vergleich von (17) mit (12) ergibt sich sofort der wichtige Zusammenhang

$$F(k_\xi \cos\phi, k_\xi \sin\phi) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} d\xi\, p(\xi;\phi)\, \mathrm{e}^{ik_\xi \xi}\,. \tag{18}$$

Wenn man also die Projektionsfunktion für verschiedene Winkel ϕ gemessen hat, kann man im Prinzip durch numerische Auswertung dieser Gleichung die Fourierkoeffizienten der gesuchten Dämpfungsfunktion entlang *radialer Linien* im $(k_x; k_y)$-Raum berechnen (s. Abb. 6).

Hier ergibt sich nun ein Problem: um aus den Fourierkoeffizienten $F(k_x, k_y)$ die uns primär interessierenden Werte der Dämpfungsfunktion (= Masseverteilungsfunktion) $f(x,y)$ zu erhalten, muß eine zu (14) inverse FT durchgeführt werden. Eine solche Rechnung ist jedoch nur dann einfach durchzuführen, wenn die Fourierkoeffizienten in Form eines kartesischen Punktnetzes vorliegen. Dies ist aber offensichtlich nicht der Fall (s. Abb. 6)! Man könnte nun unter Verwendung der "an den Polarkoordinaten-Positionen" vorliegenden Fourierkoeffizienten F die benötigten Werte durch Interpolation gewinnen. Dies ist aber aus folgendem Grund problematisch: eine Interpolation involviert immer einen lokalen Fehler. Bei der darauf folgenden inversen FT verbreitet sich dieser Fehler unkontrolliert über den gesamten Ortsraum!

Wegen dieses Problems wird die "direkte inverse FT" selten angewendet. Statt dessen wird eine Rücktransformation durchgeführt, bei welcher die (notwendige!) Interpolation erst im Ortsraum und nicht bereits im Wellenzahlraum durchgeführt wird.

Wie kann das funktionieren?

Die zu Glg. (14) gehörende inverse Fourier-Transformation lautet

$$f(x,y) = \frac{1}{2\pi} \int\int_{-\infty}^{+\infty} dk_x dk_y \, F(k_x,k_y) \, \mathrm{e}^{-i(k_x x + k_y y)} \, . \tag{19}$$

Nun werden sowohl der Vektor $\mathbf{r} = (x/y)$ als auch der Vektor $\mathbf{k} = (k_x/k_y)$ in Polarkoordinaten geschrieben, also

$$k_x = k\cos\phi \qquad \text{und} \qquad k_y = k\sin\phi$$

sowie

$$x = r\cos\Theta \qquad \text{und} \qquad y = r\sin\Theta \, .$$

Mit Hilfe der Relation

$$k_x x + k_y y = kr\cos\phi\cos\Theta + kr\sin\phi\sin\Theta = kr\cos(\Theta - \phi) \tag{20}$$

ergibt sich dann der Ausdruck

$$f(r\cos\Theta, r\sin\Theta) = \frac{1}{2\pi}\int_0^\infty \int_0^{2\pi} d\phi dk \, k \, F(k\cos\phi, k\sin\phi) \, \mathrm{e}^{-ikr\cos(\Theta-\phi)} \, .$$

Die nun folgenden Umformungen sind elementar und führen zu

$$\begin{aligned}
f(r\cos\Theta, r\sin\Theta) = & \frac{1}{2\pi}\int_0^\infty \int_0^{\pi} d\phi dk \, k \, F(k\cos\phi, k\sin\phi) \, \mathrm{e}^{-ikr\cos(\Theta-\phi)} + \\
& + \frac{1}{2\pi}\int_0^\infty \int_\pi^{2\pi} d\phi dk \, k \, F(k\cos\phi, k\sin\phi) \, \mathrm{e}^{-ikr\cos(\Theta-\phi)} \\
= & \frac{1}{2\pi}\int_0^\infty \int_0^{\pi} d\phi dk \, k \, F(k\cos\phi, k\sin\phi) \, \mathrm{e}^{-ikr\cos(\Theta-\phi)} + \\
& + \frac{1}{2\pi}\int_0^\infty \int_0^{\pi} d\phi dk \, k \, F(-k\cos\phi, -k\sin\phi) \, \mathrm{e}^{+ikr\cos(\Theta-\phi)} \\
= & \frac{1}{2\pi}\int_0^\pi d\phi \left\{ \int_0^\infty dk \, k \, F(k\cos\phi, k\sin\phi) \, \mathrm{e}^{-ikr\cos(\Theta-\phi)} + \right. \\
& \left. + \int_0^\infty dk \, k \, F(-k\cos\phi, -k\sin\phi) \, \mathrm{e}^{+ikr\cos(\Theta-\phi)} \right\} .
\end{aligned} \tag{21}$$

Nun kennt man aber gemäß Glg.(17) nur die Fourierkoeffizienten, welche auf der positiven k_ξ-Achse liegen, d.h. in Glg. (21) gibt es in ersten Integral nur Nicht-Null-Beiträge für $k = k_\xi$, und im zweiten Integral nur für $k = -k_\xi$. Daraus ergibt sich

$$f(r\cos\Theta, r\sin\Theta) = \frac{1}{2\pi}\int_0^\pi d\phi \int_{-\infty}^{+\infty} dk_\xi \, |k_\xi| \, F(k_\xi\cos\phi, k_\xi\sin\phi) \, \mathrm{e}^{-ik_\xi r\cos(\Theta-\phi)} \, ,$$

bzw. unter Verwendung der aus Abb. 7 hervorgehenden Beziehung

$$\xi = r\cos(\Theta - \phi) \tag{22}$$

das Ergebnis

$$f(r\cos\Theta, r\sin\Theta) = \int_0^\pi d\phi \left\{ \frac{1}{2\pi}\int_{-\infty}^{+\infty} dk_\xi \, |k_\xi| \, F(k_\xi\cos\phi, k_\xi\sin\phi) \, \mathrm{e}^{-ik_\xi \xi} \right\} . \tag{23}$$

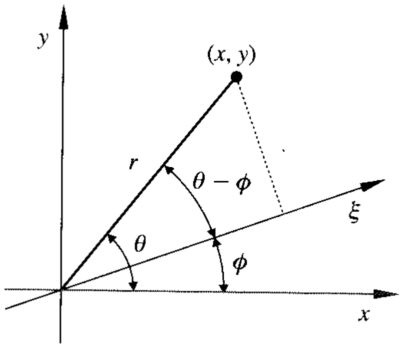


Figure 7: Erklärung der Gleichung (22).

Die in der geschwungenen Klammer definierte Größe

$$\tilde{p}(\xi;\phi) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} dk_\xi\, |k_\xi|\, F(k_\xi \cos\phi, k_\xi \sin\phi)\, \mathrm{e}^{-ik_\xi \xi} \tag{24}$$

nennt man die "modifizierte Projektion" für eine Winkelrichtung ϕ. Wie Sie unmittelbar sehen, handelt es sich dabei um die (eindimensionale) inverse Fouriertransformierte der Funktion

$$|k_\xi|\, F(k_\xi \cos\phi, k_\xi \sin\phi)\,,$$

und der gewünschte Werte der Dichtefunktion für einen bestimmten Punkt im Ortsraum ergibt sich gemäß Glgen. (23) und (24) zu

$$f(x,y) = f(r\cos\Theta; r\sin\Theta) = \int_0^\pi d\phi\, \tilde{p}(\xi;\phi)\,. \tag{25}$$

Als letztes muß nur noch eine Beziehung zwischen der Größe ξ und dem gewünschten Ortpunkt (x/y) hergestellt werden. Dies gelingt durch Kombination der Gleichungen (22) und (20):

$$\xi = r\cos(\Theta - \phi) = x\cos\phi + y\sin\phi\,. \tag{26}$$

4. Programmierung

Achtung: als Hilfestellung für die Programmerstellung finden Sie auf der Website dieser LV den Textfile *programmtest.txt*.

- Angenommen, die Masseverteilungsfunktion $f(x,y)$ eines Objektes soll aus einer Reihe von n_ϕ eindimensionalen Röntgenprofilen rekonstruiert werden: dann gehört jedes Profil zu einem bestimmten Winkel

$$\phi_i = i\,\frac{\pi}{n_\phi} \quad \text{mit} \quad i = 1,\ldots,n_\phi$$

und setzt sich aus n gemessenen Werten ξ zusammen, die im Intervall $[-\xi_{max}, +\xi_{max}]$ folgendermaßen äquidistant verteilt sind:

$$\xi_j = -\xi_{max} + (j-1)\,\Delta\xi \quad \text{mit} \quad \Delta\xi = \frac{2\xi_{max}}{n} \quad \text{und} \quad j = 1, \ldots, n$$

Diese Meßdaten werden nun eingelesen, wobei die äußerste Programmschleife über die verschiedenen Winkel geht: $i = 1, \ldots, n_\phi$.

- Als erstes wird nun die Glg. (18) ausgewertet: wie bereits erläutert, ist die Größe $F(k_\xi \cos\phi, k_\xi \sin\phi)$ die Fouriertransformierte der (gegebenen) Profilfunktion $p(\xi;\phi)$. Da diese Funktion nur als Menge diskreter Punkte ($\xi = \xi_j, j = 1, \ldots, n$) gegeben ist, muß eine diskrete Fourier-Transformation durchgeführt werden. Um die effiziente FFT-Methode anwenden zu können, muß n eine Potenz von 2 sein! Selbstverständlich ist damit auch $F(k_\xi \cos\phi, k_\xi \sin\phi)$ nur für diskrete Argumente bekannt, nämlich für die Werte

$$k_\xi^{(j)} = \frac{2\pi(j-1)}{n\Delta\xi}, \quad j = 1, \ldots, n$$

- Nun kommt ein wichtiges Detail, das ich bereits im Skriptum "Numerische Methoden in der Physik" im Abschnitt 3.4.3 ausführlich dargelegt habe[3]:
Die erhaltenen Wellenzahlen $k_\xi^{(j)}$ sind sozusagen die "Frequenzen" der Fourier-Entwicklung. Um nun eine Interpolationsfunktion zu erhalten, die möglichst geringe Oszillationen enthält, ist es sinnvoll, eine sogenannte "Frequenz-Verschiebung" (oder im gegebenen Fall "Wellenzahl-Verschiebung") durchzuführen. Dies geschieht mittels der Gleichungen[4]

$$k_\xi^{(j)} = \frac{2\pi(j-1)}{n\Delta\xi}, \qquad j = 1, \ldots, n/2, \tag{27}$$

$$k_\xi^{(j)} = \frac{2\pi(j-1-n)}{n\Delta\xi}, \qquad j = n/2+1, \ldots, n. \tag{28}$$

- Wie bereits ausführlich erklärt, können die so erhaltenen Werte F nicht direkt aus dem $(k;\phi)$-Raum in den Ortsraum rücktransformiert werden. Statt dessen berechnet man als nächstes die durch Glg. (24) definierten modifizierten Fourierkoeffizienten

$$|k_\xi^{(j)}|\, F(k_\xi^{(j)} \cos\phi; k_\xi^{(j)} \sin\phi). \tag{29}$$

Diese Koeffizienten werden dann unter Beachtung von (24) mittels einer inversen FFT in den Ortsraum rücktransformiert.

- Wie die Gleichung (25) zeigt, stellen die erhaltenen Werte der modifizierten Projektion $\tilde{p}$ den Beitrag zur Masseverteilungsfunktion f *für*

[3] **Achtung:** lesen Sie bitte zu diesem Punkt meine kurze Studie *k_optimierung.pdf*.
[4] Vergleichen Sie diese Ausdrücke mit Glg. (10) und (11) in dieser Übungsbeschreibung.

einen bestimmten Winkel ϕ dar. Dieser Beitrag kann nun (durch Interpolation) im Ortsraum (!) für jeden gewünschten (x, y)-Wert bestimmt werden, und zwar auf die folgende einfache Weise [s. Gleichungen (25) und (26)]:

Angenommen, es soll der Beitrag der Projektion für ein festes ϕ am Ort $x = x_0$ und $y = y_0$ bestimmt werden. Dazu berechnet man zuerst den entsprechenden Wert für ξ:

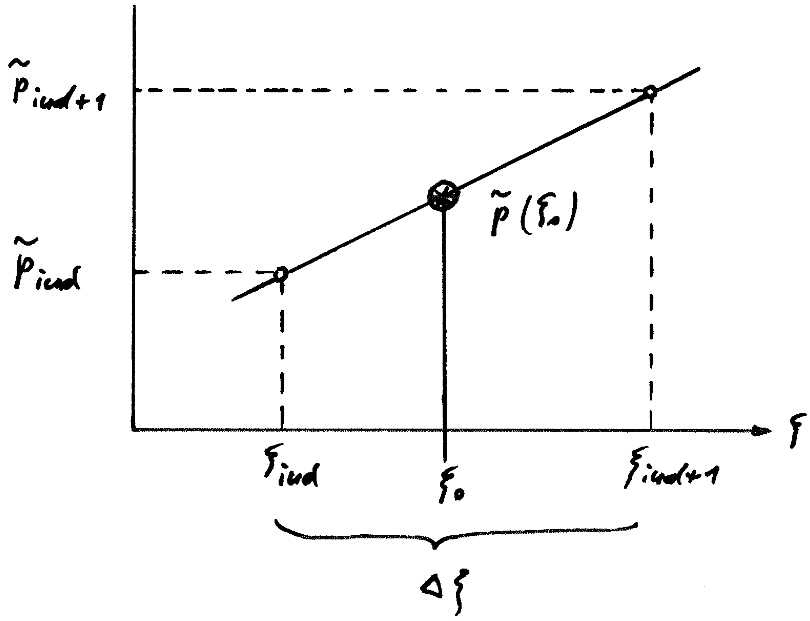
$$\xi_0 = x_0 \cos\phi + y_0 \sin\phi$$

und approximiert aus den bekannten Werten $\tilde{p}(\xi_i; \phi)$ den Wert $\tilde{p}(\xi_0; \phi)$ mittels der folgenden *linearen Interpolation*:

Als erstes sucht man jene Zahl *ind*, für welche

$$\xi_0 \in [\xi_{ind}, \xi_{ind+1}]$$

gilt. Die Interpolation ist dann trivial (s. Abb.)



und führt zu

$$\frac{\tilde{p}_{ind+1} - \tilde{p}_{ind}}{\Delta\xi} = \frac{\tilde{p}_{ind+1} - \tilde{p}(\xi_0)}{\xi_{ind+1} - \xi_0}$$

bzw. zu

$$\tilde{p}(\xi_0) = \frac{1}{\Delta\xi}\left[(\xi_0 - \xi_{ind})\tilde{p}_{ind+1} + (\xi_{ind+1} - \xi_0)\tilde{p}_{ind}\right] . \tag{30}$$

- Zuletzt müssen noch alle Beiträge verschiedener Winkel ϕ, die zum selben Punkt (x_0, y_0) gehören, im Sinne der Gleichung (25) zusammengefaßt werden. Da auch die $\tilde{p}$-Werte nur für diskrete ϕ bekannt sind, wird die Integration (25) durch die Summation

$$f(x_0; y_0) = \frac{1}{2n_\phi} \sum_{i=1}^{n_\phi} \tilde{p}(\xi_0; \phi_i) \tag{31}$$

ersetzt.

- Ausgabe der Ergebnisse:

  Die Ergebnisse der Rekonstruktion sollen auf einem kartesischen Netz in der $(x;y)$-Ebene dargestellt werden; die entsprechenden Netzpunkte sind in der Form

$$x_i = x_{min} + (i-1)\Delta x \quad (i = 1, \ldots, n_x + 1) \qquad \Delta x = \frac{x_{max} - x_{min}}{n_x} \quad (32)$$

und

$$y_j = y_{min} + (j-1)\Delta y \quad (j = 1, \ldots, n_y + 1) \qquad \Delta y = \frac{y_{max} - y_{min}}{n_y} \quad (33)$$

definiert. Betrachten Sie die Werte $f_{i,j} = f(x_i, y_j)$ als (i,j)-te Komponente einer Matrix, so können Sie diese Matrix bequem über Matlab graphisch ausgeben.

Angenommen, Ihre Ergebnismatrix heißt *fimage*. Dann ergibt

```
mat=load('fimage');
surf(mat');
```

eine 3D-Darstellung der rekonstruierten Masseverteilung; wählen Sie

```
mat=load('fimage');
contour(mat',clines);  % clines=Zahl der C-Linien (20-30)
axis('square');
```

so erhalten Sie einen *contour plot* dieser Verteilung.

VORSCHLAG FUER DIE GRUNDSTRUKTUR DES COMPUTER-TOMOGRAPHIE-PROGRAMMES:
===

H. Sormann Maerz 2007

Die Gleichungs-bzw. Seitennummern beziehen sich auf das Uebungsskriptum.

```
(1) Definition der Parameter: nphi  n  xi_max  delta-xi
    Nullsetzen des Ergebnisfeldes  IMAGE

(2) Aeusserste Schleife ueber die nphi Winkel PHI

(3)   Berechnung des Projektions-Profils fuer ein PHI
      durch Aufruf meines Programms  proj_test1(xi,phi)

(4)   FFT des Projektions-Profils (Glg. 18):
      Beachten Sie die Details beim Aufruf der FFT fuer die
      verschiedenen Programmiersprachen.
      EIN WICHTIGER PUNKT FUER C-USER:
        Das zweite Argument in der Parameterliste von FOUR1,
        naemlich die Zahl der Stuetzpunkte, muss den Typ
        "unsigned long" haben. Dieser Typ kann aber bei
        der Verwendung in mathematischen Ausdruecken
        Schwierigkeiten machen. Definieren Sie deshalb diese Groesse
        zweimal:
                  unsigned long  nn;
                  int            n;

                  n= 32; (z.B.)
                  nn=n;

(5)   Durchfuehrung der "Wellenzahl-Verschiebung", Glg. (27, 28).

(6)   Berechnung der modifizierten Fourierkoeffizienten, Glg. (29).

(7)   INVERSE FFT der modifizierten Fourierkoeffizienten fuehrt zu den
      modifizierten Projektionen, s. Glg. (24).
      ACHTUNG auf Normierungsfaktor 1/n !
      Die ruecktransformierten Werte sollten (bis auf etwaige Rundungsfehler)
      REELL sein.

(8)   Berechnung der modifizierten Projektion AN DEN KARTESISCHEN
      KOORDINATEN des x/y-Netzes durch lineare Interpolation (s. Seiten
      21 und 22).

(9)   Aufsummierung der interpolierten Werte gemaess Glg. (31).

(10)  Ende der Schleife ueber den Winkel PHI.

(11)  Graphische Ausgabe des Feldes "image"
```

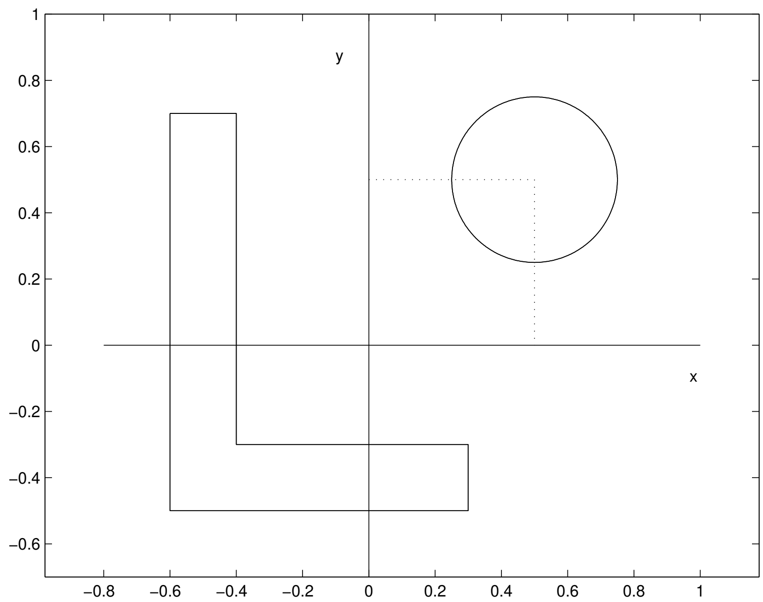


Figure 8: Testobjekt für die Computer-Tomographie.

5. Tests

Im folgenden sollen Sie die Leistungsfähigkeit bzw. Genauigkeit der CT-Methode an Hand eines einfachen Testbeispiels studieren. Für diese Rechnungen werden von mir via Internet die folgenden Programme zur Verfügung gestellt:
(1) Programme für die FFT für C- und F90-User mit den Namen

```
four1.c      fft.f90
```

Für Matlab-User gibt es für die FFT und die inverse FFT die internen Funktionen

```
fft.m          ifft.m
```

Eine genaue Beschreibung dieser Programme finden Sie im Abschnitt 2.2 dieser Übungsbeschreibung.

(2) Ein Programm, mit welchem Sie eindimensionale Röntgenprofile für das einfache Muster Abb. 8 per Computer simulieren können. Auch diese Routine kann in C-, F90- und Matlab-Version heruntergeladen werden; die Aufruf-Argumente sind einfach die Größen ξ und ϕ:

```
F90:      double precision function  proj_test1(xi,phi)
            implicit none
            double precision xi,phi
            .
            .
```

```
C:        double proj_test1(double xi, double phi)
          .
          .
```

```
Matlab:   function profilwert = proj_test1(xi,phi)
```

Achtung: Die Input-Größe *xi* kann auch ein Vektor sein; dementsprechend ist auch die Output-Größe *profilwert* ein Vektor!

(3) Parameter der Rechnungen:

$$n_\phi = 12$$

$$n = 32 = 2^5 \qquad \xi_{max} = 1.6$$

Ihr Programm könnte (z. B. in C) so beginnen:

```
nphi=12;
n=32;
xi_max=1.6;
delta_xi=2*xi_max/n;

for(iphi=1;iphi<=nphi;iphi++) {
  phi=iphi*M_PI/nphi;
  for(i=1;i<=n;i++) {
    xi[i]=-xi_max + (i-1)*delta_xi;
    projection[i]=proj_test1(xi[i],phi);
  }
  .
```

Die Ergebnis-Matrix [s. Glgen. (32, 33)] ist durch die folgenden Parameter bestimmt:

$$x_{min} = -1 \qquad x_{max} = +1 \qquad n_x = 100$$

$$y_{min} = -1 \qquad y_{max} = +1 \qquad n_y = 100$$

Anmerkung: Um die Kanten der Testobjekte möglichst scharf abbilden zu können, sollte das kartesische Netz in der xy-Ebene engmaschig sein: n_x=n_y=100 ist durchaus notwendig. Andererseits steigt die Rechenzeit mit steigender Netzdichte massiv an.

Ich rate Ihnen daher zur folgenden Strategie: verwenden Sie für die Programmentwicklung n_x=n_y=50, und gehen Sie erst bei den "Produktionsläufen" auf ein 100x100-Netz über.

(4) Testaufgaben:

- Stellen Sie Ihr rekonstruiertes Muster mit den auf S. 26 gegebenen Parametern sowohl als "Surf-Verteilung" als auch als "Contour-Plot" (Matlab) dar. Paßt Ihr Ergebnis zum Muster in Abb. 8?

- Demonstrieren Sie nun die Wichtigkeit der durch die Gleichungen (27), (28) beschriebenen "Wellenzahl-Verschiebungen", indem Sie Ihr Programm so gestalten, daß Sie diesen Teil des Algorithmus wahlweise ein- und ausschalten können.

- Welche Möglichkeiten gibt es, die *performance* des Programms zu verbessern?

  - mehr Projektionen messen $\longrightarrow n_\phi$ erhöhen
  - mehr Meßpunkte pro Projektionswinkel $\longrightarrow n$ erhöhen

  Vergleichen Sie Contour-Plots, die Sie mit den folgenden Parametern erhalten:

$$n = 32 = 2^5 \qquad \text{und} \quad n_\phi = 12 \quad 24 \quad 48 \quad 96$$

$$n_\phi = 48 \qquad \text{und} \quad n = 32 \quad 64 \quad 128 \quad 256$$

- Diskutieren Sie die Qualität dieser "Bilderserien".

  Zu den Bilderserien eine Anmerkung für Leute ohne Matlab-Kenntnisse:

  Für die Diskussion bei der Übungspräsentation ist es wichtig, daß alle 4 Bilder einer Serie nebeneinander auf dem Bildschirm zu sehen sind.

  Eine einfache Realisierung mittels Matlab-Grafik kann so geschehen: angenommen, die Daten für die 4 Bilder befinden sich auf den Files bild1.erg bis bild4.erg. In diesem Fall laden Sie die Inhalte der vier Files im Matlab-Fenster ein:

```
dat1=load('bild1.erg');     dat2=load('bild2.erg');
dat3=load('bild3.erg');     dat4=load('bild4.erg');
```

  und zeichen dann die vier Diagramme als "subplots":

```
subplot(2,2,1)            subplot(2,2,2)
  contour(dat1,20);         contour(dat2,20);
  axis('square');           axis('square');

subplot(2,2,3)            subplot(2,2,4)
  contour(dat3,20);         contour(dat4,20);
  axis('square');           axis('square');
```

6. Rekonstruktion eines unbekannten Objektes

Im folgenden soll nun, auf sehr vereinfachte Weise, eine typische Anwendungssituation für eine Computer-Tomographie im technischen Bereich studiert werden:

Hinter einer geschlossenen, undurchsichtigen Umrandung befinde sich eine bestimmte Struktur von 4 x 4 Rohrleitungen. Es soll nun überprüft werden, ob diese Rohre (die in unserem einfachen Querschnitt-Modell als Kreise erscheinen) intakt oder mehr oder weniger beschädigt sind, und zwar ohne die Umrandung zu durchbrechen.

Nehmen wir an, man hätte dieses Gehäuse im Sinne einer CT mit Röntgenstrahlen durchstrahlt, und die gemessenen Profilkurven seien auf dem File

`rohre.0`

abgespeichert.

Die Parameter dieses Datenfiles:

```
96 Profilkurven, d.h.:   nphi = 96
Jede Profilkurve besteht aus 2^7 Datenpunkten, d.h.:  n = 128
Die xi-Werte liegen wie beim Testbeispiel zwischen -1.6 und +1.6,
und fuer die Ergebnismatrix soll wieder nx=ny=100 gelten (Glg. 32,33).
```

<u>Einlesen des Datenfiles für C, F90 und Matlab:</u>

```
C:        int nphi, n, iphi, i;
          float *projection;
          FILE *inp;

          inp=fopen("rohre.0","r");
          nphi=96;  n=128;

          projection=vector(1,n);            // "vector" aus  nrutil.c

          for(iphi=1;iphi<=nphi;iphi++) {
            for(i=1;i<=n;i++) fscanf(inp,"%f ", &projection[i]);
            .
            .

F90:   program input
       implicit none

       integer  ::  nphi,log_2_n,n,iphi
       double precision,allocatable,dimension(:) :: projection
```

```
nphi=96
log_2_n=7
n=2**log_2_n

open(10,file='rohre.0',status='old',recl=10*n)
allocate (projection(n))

do iphi=1,nphi
  read(10,*)projection
  .
  .
```

```
Matlab:   dat=load('rohre.0');
              nphi=length(dat(:,1));
              n=length(dat(1,:));

              projection=zeros(1,n);
              for iphi=1:nphi
                projection=dat(iphi,:);
                .
```

- **Aufgabe 1:** CT-Rekonstruktion des Rohr-Systems innerhalb des (als unzugänglich angenommenen) Behälters unter Verwendung des Datenfiles

  `rohre.0`

- **Aufgabe 2:** Jede (jeder) von Ihnen erhält von mir in einer namentlich bezeichneten *directory* drei weitere Röntgenprofil-Files mit den Namen

  `rohre.1   rohre.2   rohre.3`

  Untersuchen Sie die Inhalte dieser drei Rohrbehälter und stellen Sie fest, welche von ihnen

  – einwandfrei sind bzw.
  – ein (oder mehrere) *schadhafte* Rohre enthalten.
  – Unterscheiden Sie zwischen starker und schwacher Beschädigung.
  – Stellen Sie alle Masseverteilungen als *contour plots* grafisch dar.

  Anmerkung: In diesem simulierten Problem werden die Zustände der einzelnen Rohre durch einen einfachen "Code" dargestellt:
  intakte Kreise repräsentieren unbeschädigte Rohre, schwach bzw. stark beschädigte Rohre werden durch Halbkreise mit dicken bzw. dünnen Wandstärken dargestellt.

7. Rekonstruktion der Elektronen-Impulsverteilung in Metallen

Zum Abschluß dieser Übung soll nun noch eine wichtige Anwendung der CT in der Festkörperphysik demonstriert werden, nämlich die Rekonstruktion der Impulsverteilung von Elektronen in einem Metall aus einer Reihe von (eindimensionalen) Profilen. Diese Profile werden experimentell dadurch erhalten, daß die Metallprobe von einem intensiven monochromatischen Röntgenstrahl (z. B. aus einem Synchrotron-Beschleuniger) durchdrungen wird. Dabei kommt es zu *Compton-Streuprozessen* der Röntgenphotonen mit den Metallelektronen, und die Winkelverteilung der erhaltenen Röntgen-Streustrahlung ergibt eine eindimensionale Projektion der räumlichen Impulsverteilung der Elektronen im Festkörper. Auf diese Weise erhält man ein sogenanntes *Compton-Profil*.

Wenn man nun den Röntgenstrahl aus verschiedenen Richtungen auf die Probe einwirken läßt, ergeben sich eine Anzahl verschiedener Compton-Profile, die man mit dem in dieser Übung eingesetzten Programm zu einer Dichteverteilung der Elektronen im Impulsraum zusammensetzen kann.

Bevor ich Ihnen nun die nötigen Informationen über die entsprechende Computer-Tomographie gebe, möchte ich wenigstens grundsätzlich über das Thema "Elektronenimpulse in einem kristallinen Festkörper" sprechen.

Wie Sie wahrscheinlich wissen, sind Kristalle aus lauter winzigen Volumseinheiten zusammengesetzt, welche man Einheitszellen nennt. Die Form dieser Einheitszellen entscheidet darüber, welche Art von Symmetrie ein Kristall aufweist, d.h., ob er eine kubische oder hexagonale oder tetragonale usw. Struktur hat.

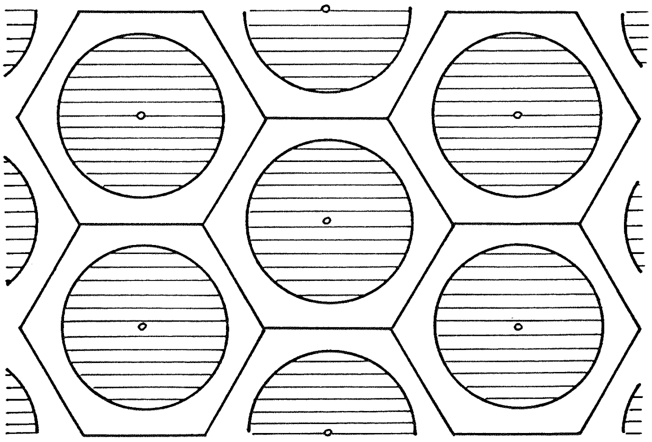
Metallische Festkörper haben nun häufig ein sehr einfaches Aufbauprinzip: im Zentrum jeder Einheitszelle sitzt ein Atomkern des entsprechenden metallischen Elementes, und die Elektronen (oder zumindest jene Elektronen, die man Valenzelektronen nennt) können sich innerhalb einer Einheitszelle, aber auch von Zelle zu Zelle relativ ungehindert bewegen.

So sieht - stark vereinfacht - der Aufbau eines Metalls im realen Raum (Ortsraum) aus. In der hier vorzunehmenden Untersuchung geht es aber nicht um die Physik der Metallelektronen im Ortsraum, sondern im Impulsraum, d.h. wir möchten gerne wissen, wie die Impulse der Metallelektronen verteilt sind.

Nun kann man die Physik eines Kristalls im Impulsraum im Prinzip ähnlich beschreiben wie im Ortsraum: auch im Impulsraum ist der Kristall aus vielen Einheitszellen lückenlos zusammengesetzt, welche man nach dem im 20. Jh. wirkenden bedeutenden französischen Physiker L. Brillouin Brillouinzonen nennt.

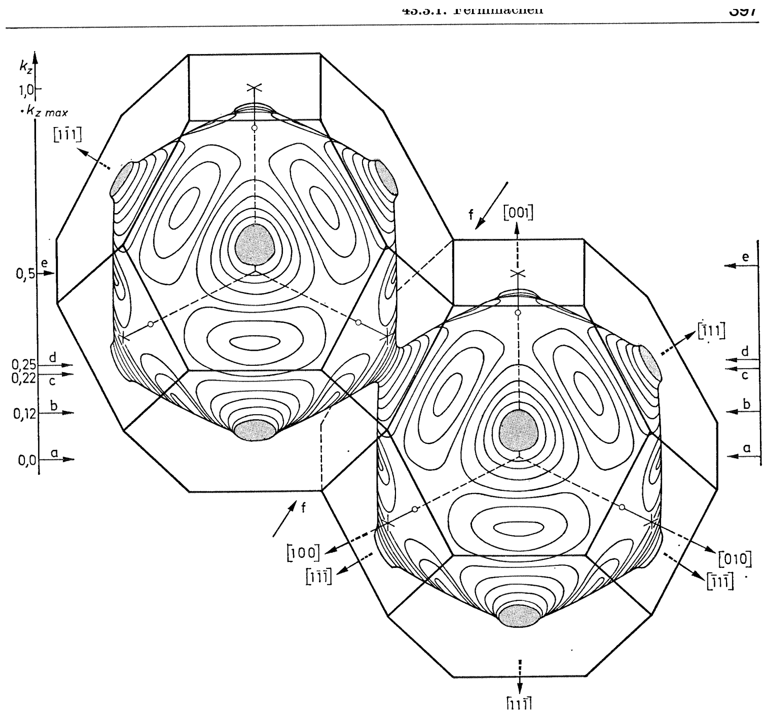
Ein weiterer, sehr wichtiger Physiker der 20. Jahrhunderts, der in Theorie und Experiment gleichermaßen erfolgreiche Italiener Enrico Fermi, hat ein relativ einfaches Modell für die Impulsverteilung der Valenzelektronen erarbeitet: nach der sogenannten Fermi-Statistik sind die Impulse der Metallelektronen in jeder Brillouinzone *kugelförmig* um das Zentrum der Zone verteilt, wobei der Radius dieser Kugel (der *Fermi-Radius* oder *Fermi-Impuls*) charakteristisch für das jeweilige Metall ist.

In einfachen Metallen wie z.B. Natrium sieht das so aus:



Die schraffierten Gebiete enthalten die von den Elektronen besetzten Impulszustände; die Oberflächen dieser "Fermi-Kugeln", die eine scharfe Grenze zwischen besetzten und unbesetzten Bereichen darstellen, heißen Fermi-Flächen.

Theoretisch können Fermiflächen metallischer Festkörper mittels sogenannter "Bandstrukturmethoden" berechnet werden. Die Ergebnisse einer solchen Rechnung für Kupfer für zwei nebeneinanderliegende Brillouinzonen ist im folgenden dargestellt (aus: K.-H. Hellwege, *Einführung in die Festkörperphysik*, Springer-Verlag, Berlin, 1976, S. 397):



Wie Sie sehen, ist nach der theoretischen Vorhersage die Situation im Cu ähnlich der im Na: im Zentrum jeder Brillouinzone befindet sich ein kompakter "Fermikörper", und der Impulsraum zwischen der Oberfläche dieser Körper und der Oberfläche der Brillouinzonen ist von den Elektronen unbesetzt.

Genau besehen, erkennt man aber einige signifikante Unterschiede zwischen Natrium und Kupfer: beim Cu gibt es einige Stellen, wo die Fermikörper Grenzflächen der Brillouinzonen berühren (in der Literatur werden solche Stellen *necks* genannt). Man hat es also nicht - wie im Fall des Na - mit isolierten Fermikugeln zu tun, sondern mit einem System verbundener Fermikörper.

Außerdem zeigen die Fermikörper im Cu auch sonst eine deutliche Abweichung von der für Alkalimetalle (Li, Na, K, ...) typischen Kugelform.

Soweit die Theorie: die zweidimensionale CT-Rekonstruktion von gemessenen eindimensionalen Comptonprofilen gibt nun die spannende Möglichkeit, die theoretischen Ergebnisse experimentell zu überprüfen.

Damit Sie selbst solche Rekonstruktionen einer metallischen *electron momentum density* (EMD) durchführen können, finden Sie auf der Internet-Seite dieser Lehrveranstaltung die Files

`proj_cu_64_96.dat` und `proj_cu_128_96.dat`

Diese Files enthalten 96 Comptonprofile mit jeweils 64 bzw. 128 Werten. Der *gescannte* ξ-Bereich geht von -1.5 bis $+1.5$, und die Ergebnismatrix soll 101 Zeilen bzw. Spalten haben; der bildlich dargestellte (x/y)-Bereich soll ein Quadrat von -1.0 bis $+1.0$ umfassen soll.

- Berechnen Sie mit Ihrem Rekonstruktionsprogramm die Impulsverteilung der Elektronen in Kupfer, und stellen Sie das Ergebnis als *contour plot* (20 Konturlinien) dar. Sorgen Sie dafür, daß in der grafischen Darstellung der x- und der y-Bereich dieselbe Achsenlänge haben. Mit Matlab können Sie das mit den Befehlen

  ```
  mat=load('fimage');
  contour(mat',20);
  axis('equal');
  ```

  erreichen.

- Die Comptonprofile wurden für dieses Beispiel so erstellt, daß x- und y-Achse in Ihrem Rekonstruktions-Diagramm im Impulsraum die Richtungen [100] bzw. [011] repräsentieren. Für eine korrekte Interpretation der Ergebnisse ist es nun wichtig, auch die Umrißlinie der Brillouinzone in der [100]-[011]-Ebene einzuzeichnen. Diese Linie besteht aus 6 Punkten mit den folgenden (x,y)-Koordinaten:

  ```
  a=sqrt(2)/4        b=3*sqrt(2)/4

  (1) = (1/a)      (2) = (0/b)     (3) = (-1/a)
  (4) = (-1/-a)    (5) = (0/-b)    (6) = (1/-a)
  ```

Das Zentrum der Brillouinzone, stets mit Γ bezeichnet, liegt im Mittelpunkt Ihrer Zeichnung.

- **Bitte beachten Sie:** Für die Zeichnung der Brillouinzonen-Linie im Cu-Contourplot müssen die auf Seite 32 unten angegebenen x/y-Koordinaten unter Verwendung der Gleichungen (32) und (33) in die entsprechenden Matrix-Indizes umgerechnet werden:

$$i = \frac{x - x_{min}}{\Delta x} + 1$$

bzw.

$$j = \frac{y - y_{min}}{\Delta y} + 1$$

- Interpretieren Sie Ihre Ergebnisse: passen die theoretischen Voraussagen dazu?

8. Einige weitere Anmerkungen

- Falls Sie Lust haben, sich über weitere Anwendungen der Computertomographie zu informieren, besuchen Sie z. B. die folgenden Internet-Adressen:

****
`www.kfa-juelich.de/isr/3/tomo_e.html`

Stichwort: Kontrolle von radioaktivem Abfall mittels CT (Forschungszentrum Jülich).

****
`www.ifi.unizh.ch/staff/zolli/CAP/Main.htm`

Es geht dabei um eine Anwendung der CT in der Paläoanthropologie, Stichwort: Neandertaler ....

****

Wenn Sie sich dafür interessieren, wie man mittels CT die Geheimnisse einer altägyptischen Mumie enthüllen kann, laden Sie von der Webpage meiner Lehrveranstaltung den folgenden PDF-File herunter:

`3D_reconst_Egyptian_mummy.pdf`

Es handelt sich dabei um das paper

C. Baldock, S.W. Hughes, D.K. Whittaker, J. Taylor, R. Davis, A.J. Spencer, K. Tonge, and A. Sofat, *3-D Reconstruction of an ancient Egyptian mummy using X-ray computer tomography*, J. Roy. Soc. Med. **87**, 806 (1994).

Zur Erinnerung: **die Internet-Adresse dieser Lehrveranstaltung lautet**

www.itp.tu-graz.ac.at