Fourier methods

Lukas Palatinus

Institute of Physics
AS CR, Prague, Czechia

Outline

- Fourier transform:
  - mathematician‘s viewpoint
  - crystallographer‘s viewpoint
  - programmer‘s viewpoint
- Practical aspects
  - libraries, binding
  - sample code

Fourier transform – mathematician‘s viewpoint

Any (reasonably well behaved) function in one, two or more dimensions can be expressed as a sum (or integral) of basis functions:

$$f(x_1, x_2, \dots x_n) = \sum g_i(x_1, x_2, \dots x_n)$$

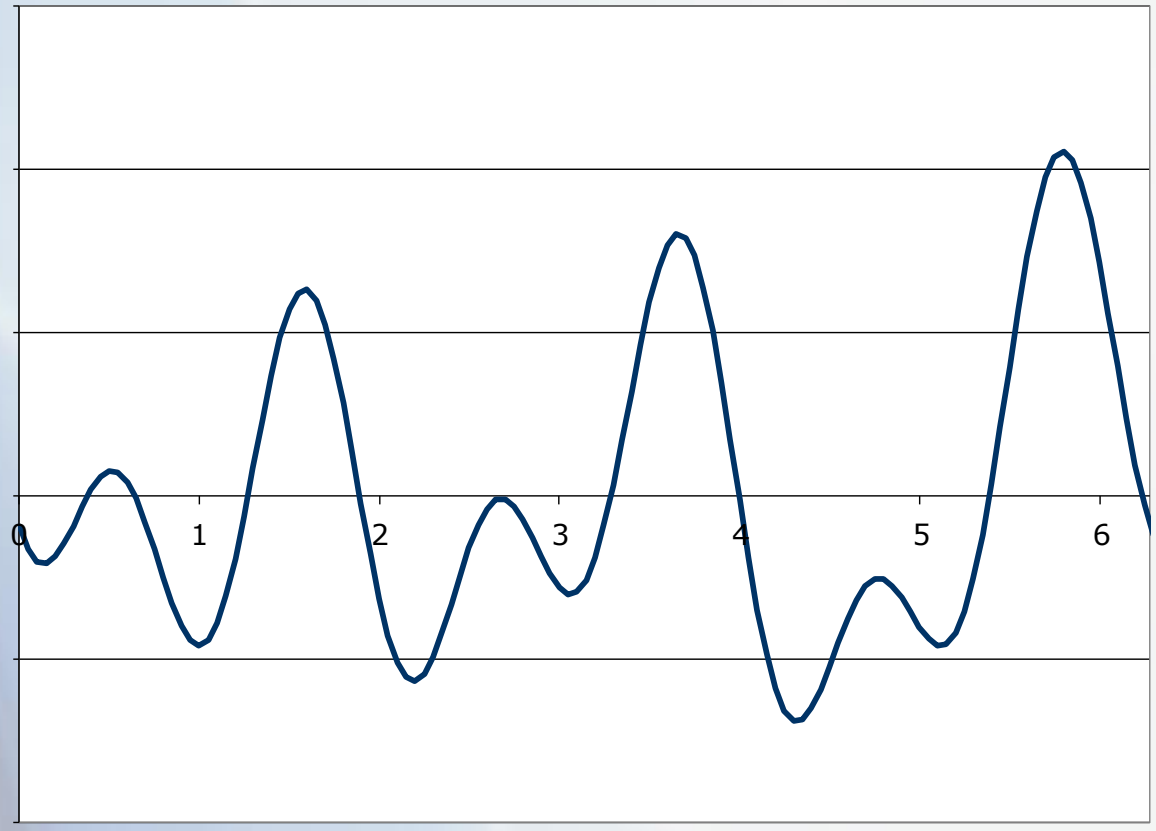
Not all bases are suitable for all functions. Smooth, bound functions varying relatively slowly in space (or time) may be very well approximated by a decomposition to constituting *frequencies and their amplitudes.* This decomposition is called *Fourier transform*. If the function is periodic, its decomposition has countably many components and is called a Fourier sum. Otherwise the number of components may be infinite and the function is expressed by a Fourier integral.

Fourier transform – mathematician‘s viewpoint

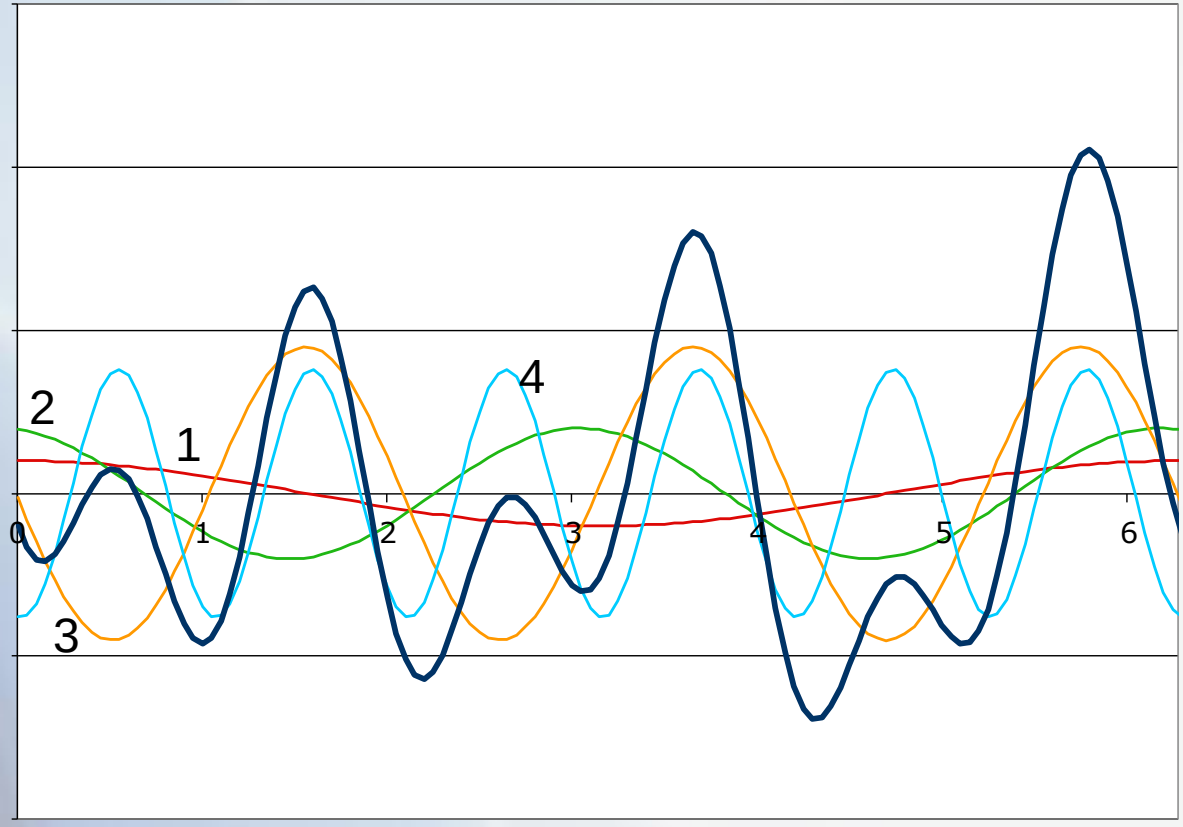
Fourier sum in 1D:

$$f(x) = \sum_{n=0}^{\infty} d_n \cos(nx + \varphi_n)$$

Fourier transform – mathematician‘s viewpoint



Fourier transform – mathematician‘s viewpoint



wave 1: cos(x)
wave 2: 2cos(2x+0.2)
wave 3: 4.5cos(3x+1.6)
wave 4: 3.8cos(6x+3.5)

Fourier transform – mathematician‘s viewpoint

Fourier sum in 1D:

$$f(x) = \sum_{n=0}^{\infty} d_n \cos(nx + \varphi_n)$$

Using $\cos(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$ we can get

$$f(x) = \sum_{n=0}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$$

where $a_n = d_n \cos(\varphi_n)\,, b_n = -d_n \sin(\varphi_n)$

Fourier transform – mathematician‘s viewpoint

Fourier sum in 1D:

$$f(x) = \sum_{n=0}^{\infty} [a_n \cos(nx) + b_n \sin(nx)]$$

Euler‘s formula:

$$\cos(nx) + i\sin(nx) = e^{inx}$$

Then:

$$f(x) = \sum_{n=0}^{\infty} \frac{a_n - ib_n}{2} [\cos(nx) + i\sin(nx)] +$$

$$+ \sum_{n=-\infty}^{0} \frac{a_{-n} + ib_{-n}}{2} [\cos(nx) + i\sin(nx)] =$$

$$= \sum_{n=-\infty}^{\infty} F_n e^{inx}$$

where $F_n = d_n e^{i\varphi_n}$ and is obtained from $F_n = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-inx}\mathrm{d}x$

Fourier transform – mathematician‘s viewpoint

Fourier sum in *n*D:

$$f(\mathbf{r}) = \sum_{\mathbf{h}=0,0,\ldots}^{\infty,\infty,\ldots} [a_{\boldsymbol{h}} \cos(\mathbf{h}.\mathbf{r}) + b_n \sin(\mathbf{h}.\mathbf{r})] = \sum_{\mathbf{h}=-\infty,-\infty\ldots}^{\infty,\infty,\ldots} F_{\mathbf{h}} e^{i\mathbf{h}.\mathbf{r}}$$

If the function is periodic in integer intervals, the factor 2π must be written expicitly:

$$f(\mathbf{r}) = \sum_{\mathbf{h}=0}^{\infty} [a_{\mathbf{h}} \cos(2\pi\mathbf{h}.\mathbf{r}) + b_{\mathbf{h}} \sin(2\pi\mathbf{h}.\mathbf{r})] = \sum_{\mathbf{h}=-\infty}^{\infty} F_{\mathbf{h}} e^{2\pi i\mathbf{h}.\mathbf{r}}$$

Fourier transform – crystallographer‘s viewpoint

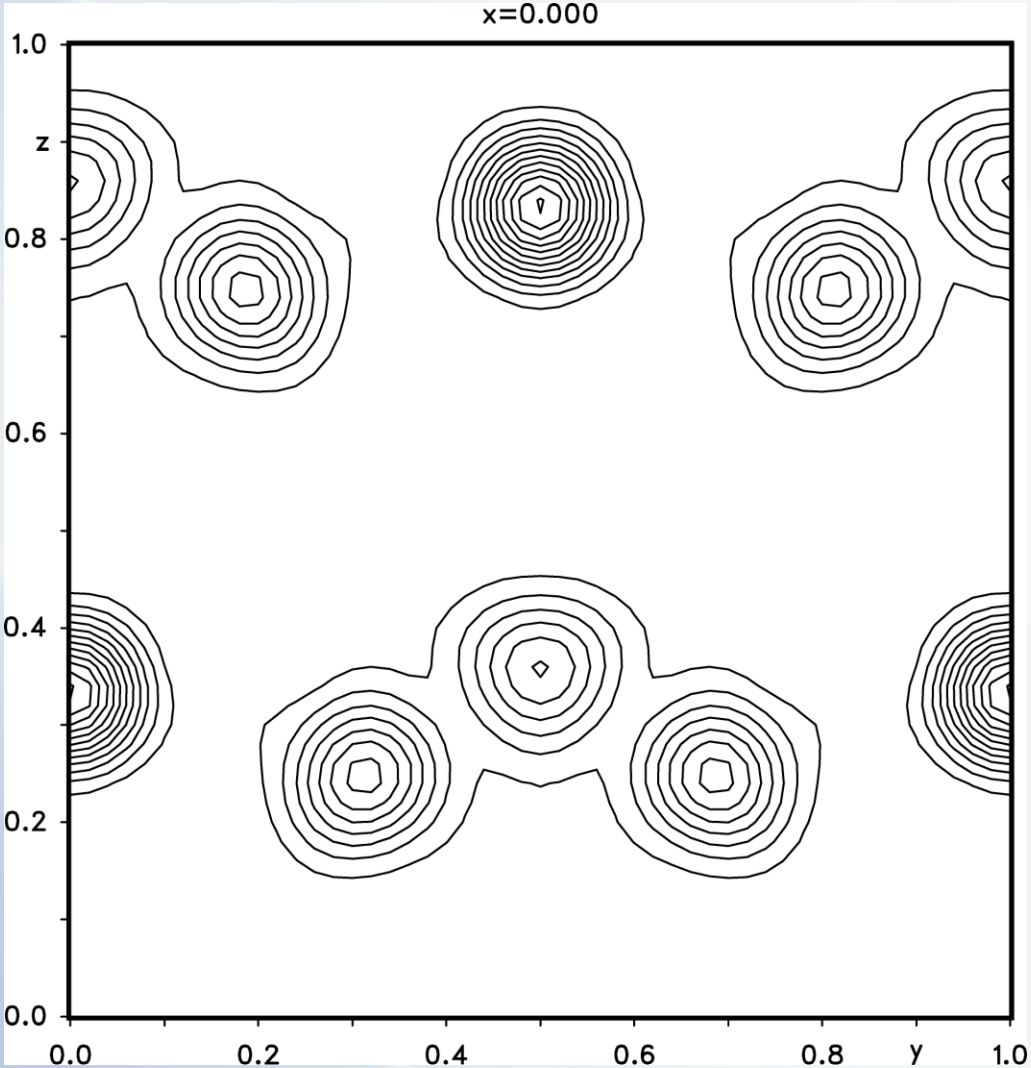
The physics of the diffraction is such that the amplitude and phase of the diffracted beams is (approximately) equal to the amplitude and phase of the Fourier coefficients of the scattering density:

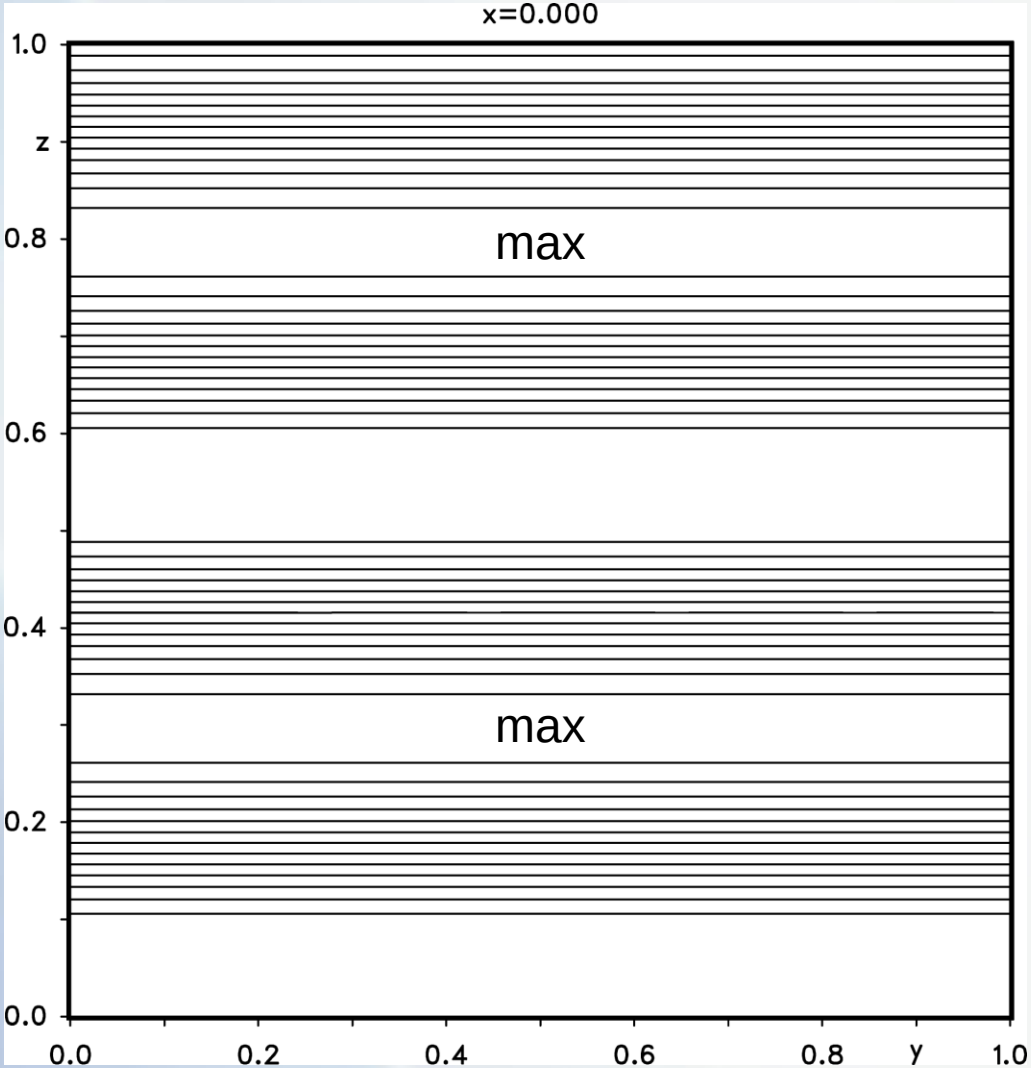
$$F_{\mathbf{h}} = \int \rho(\mathbf{r}) e^{2\pi i \mathbf{h}.\mathbf{r}} \mathrm{d}V$$

Coefficients $F_{\mathbf{h}}$ are called structure factors.

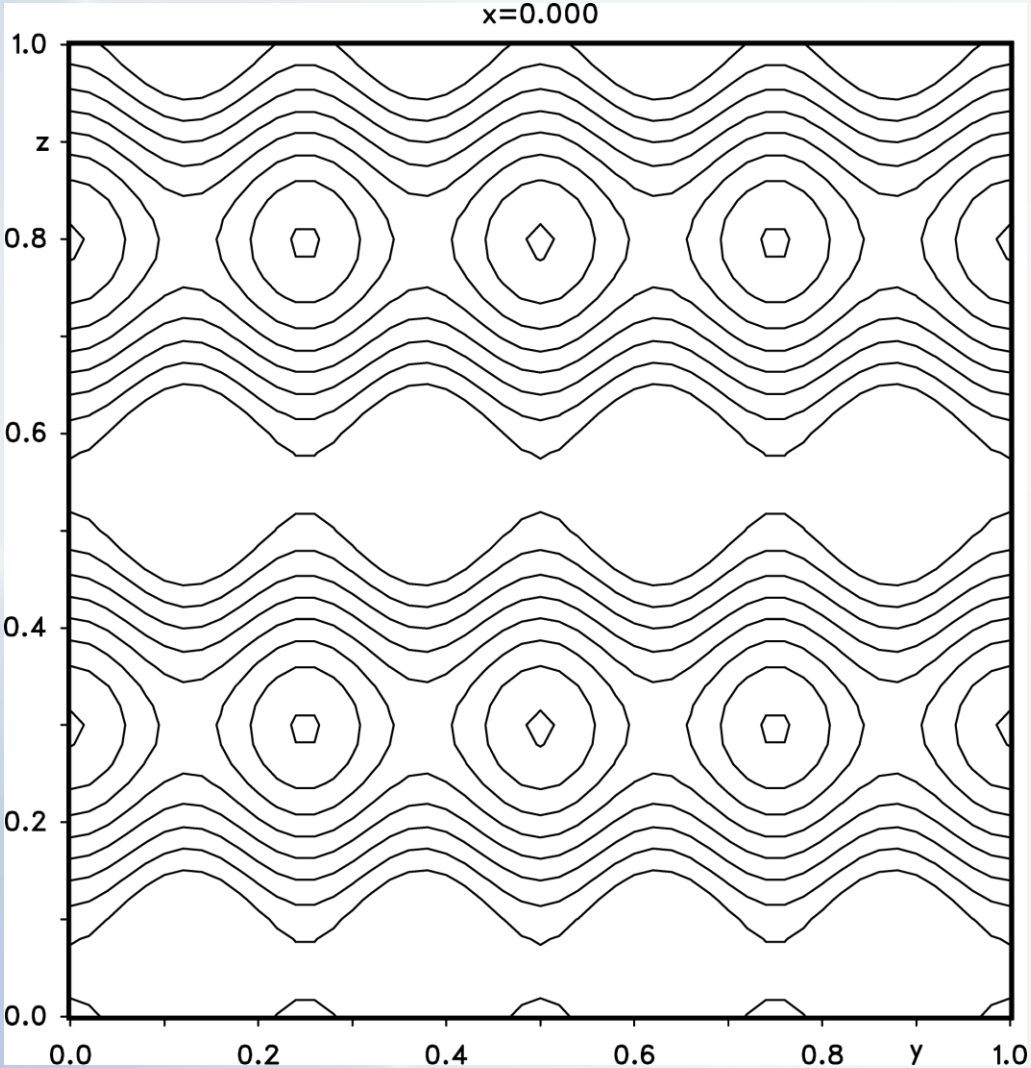
Conversely, the electron density can be calculated as a Fourier summation of the structure factors.

$$\rho(\mathbf{r}) = \frac{1}{V} \sum_{\mathbf{h}=-\infty}^{\infty} F_{\mathbf{h}} e^{-2\pi i \mathbf{h}.\mathbf{r}}$$

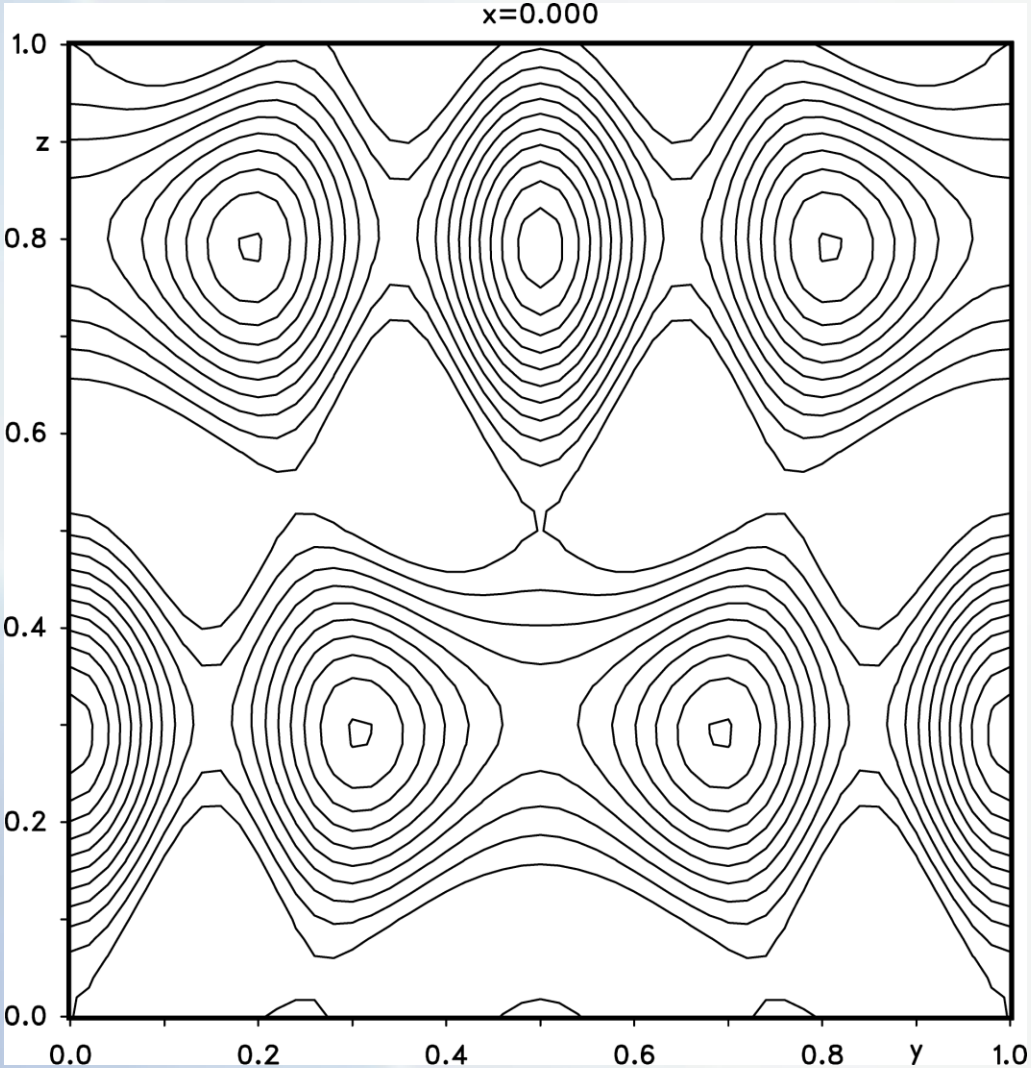

x=0.000
1.0
z
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
y
1.0



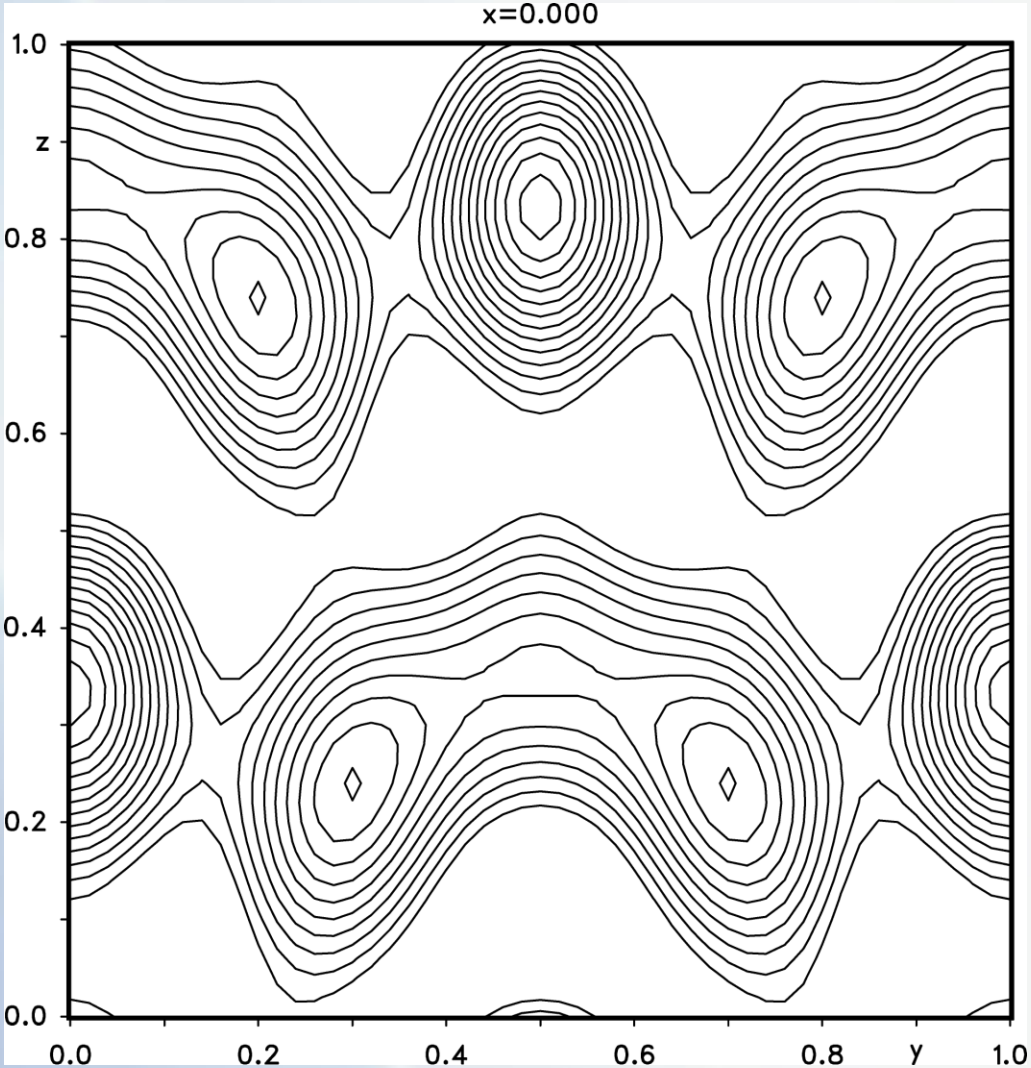
F_{002}

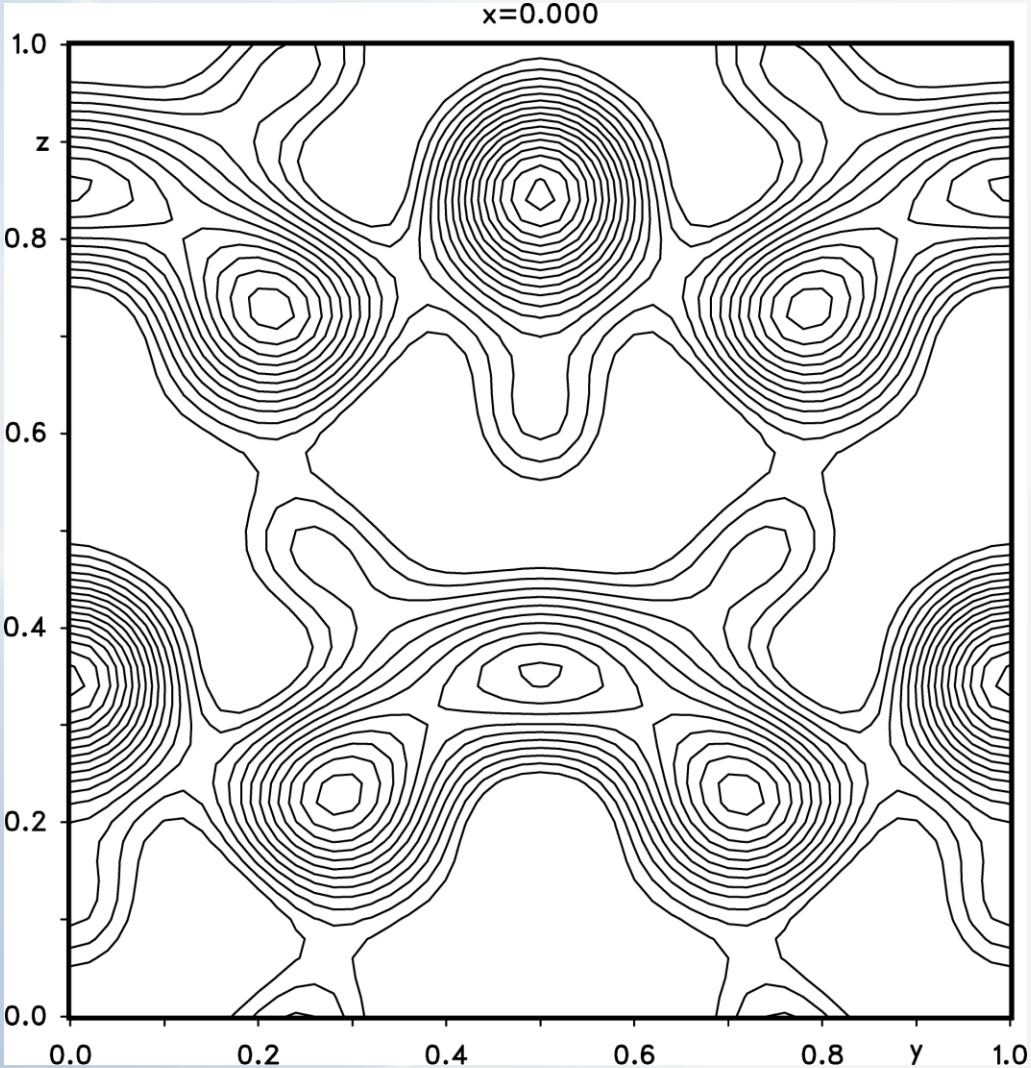


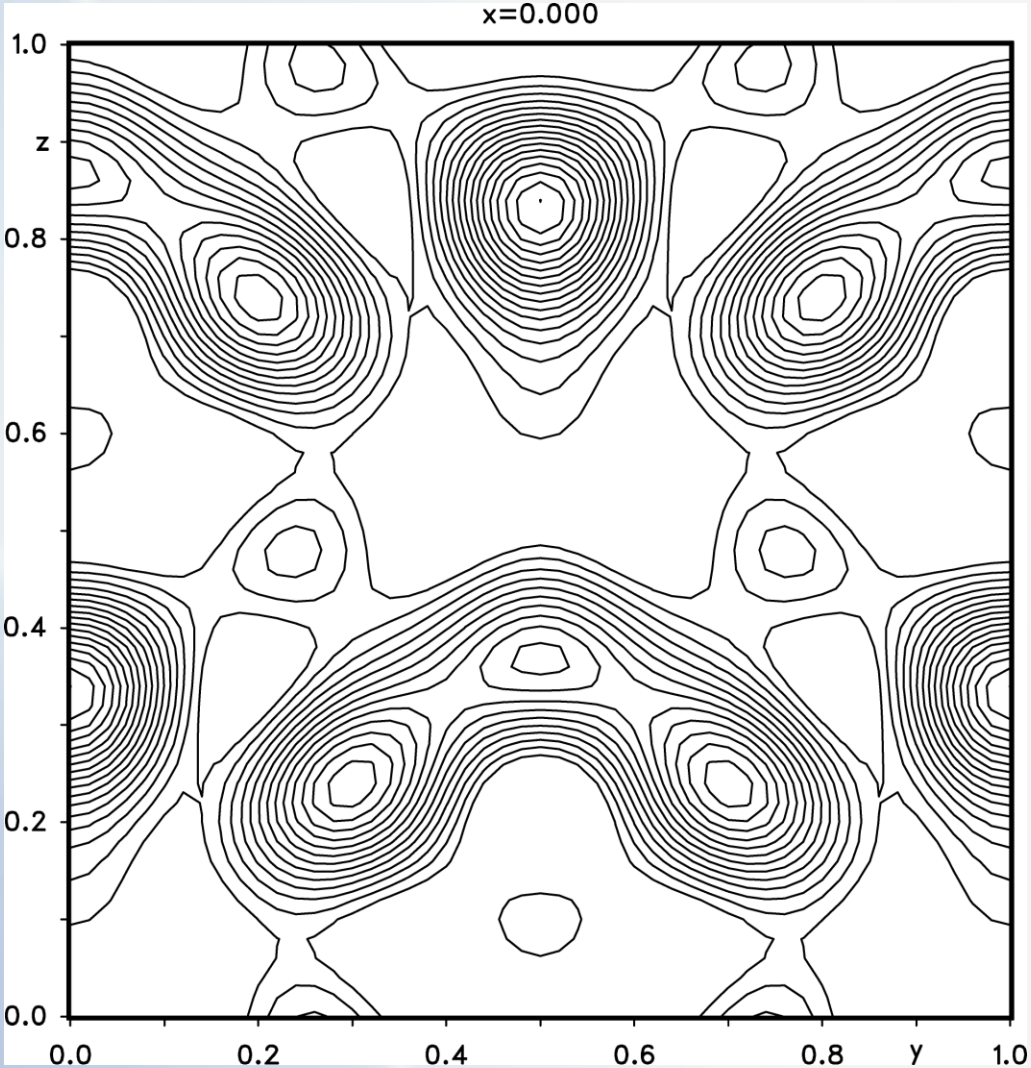
F_{002} + F_{040}

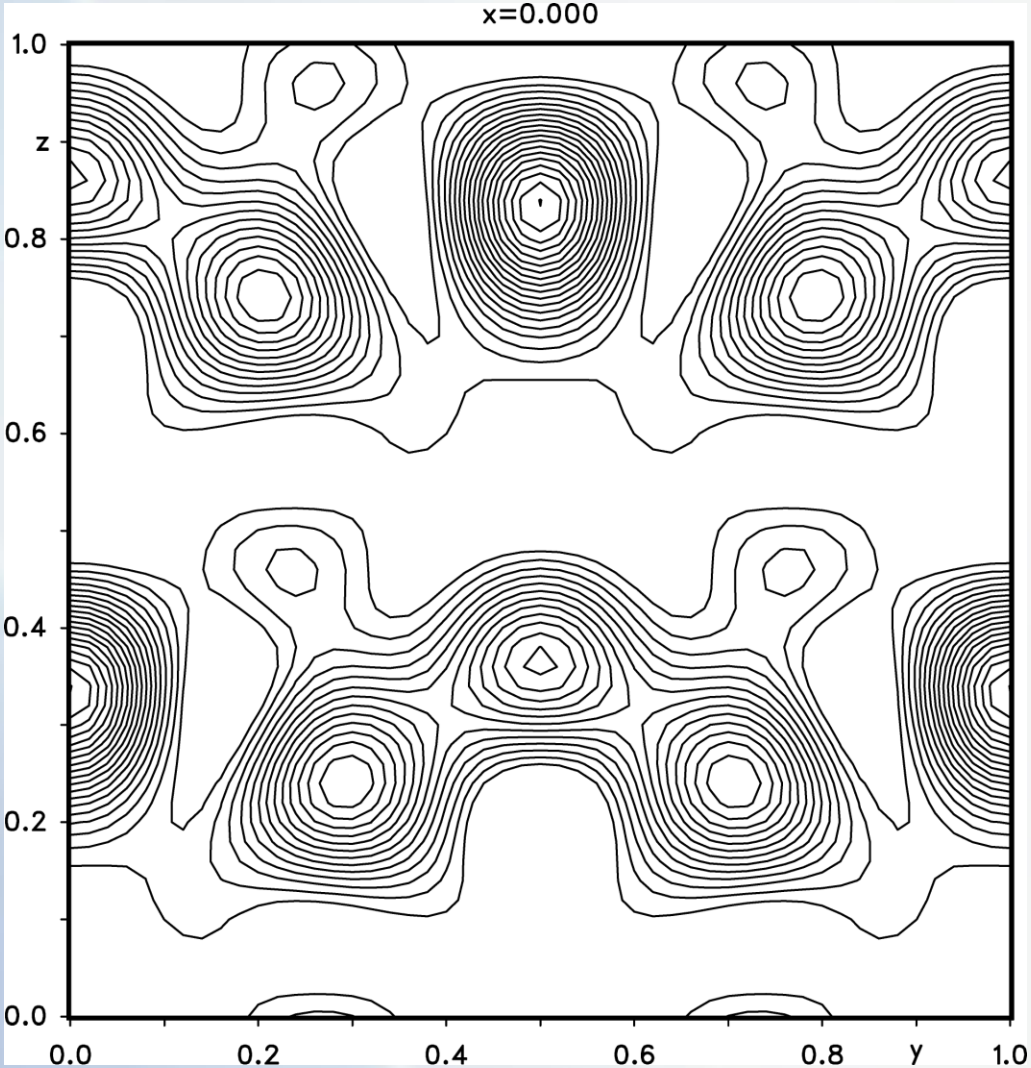


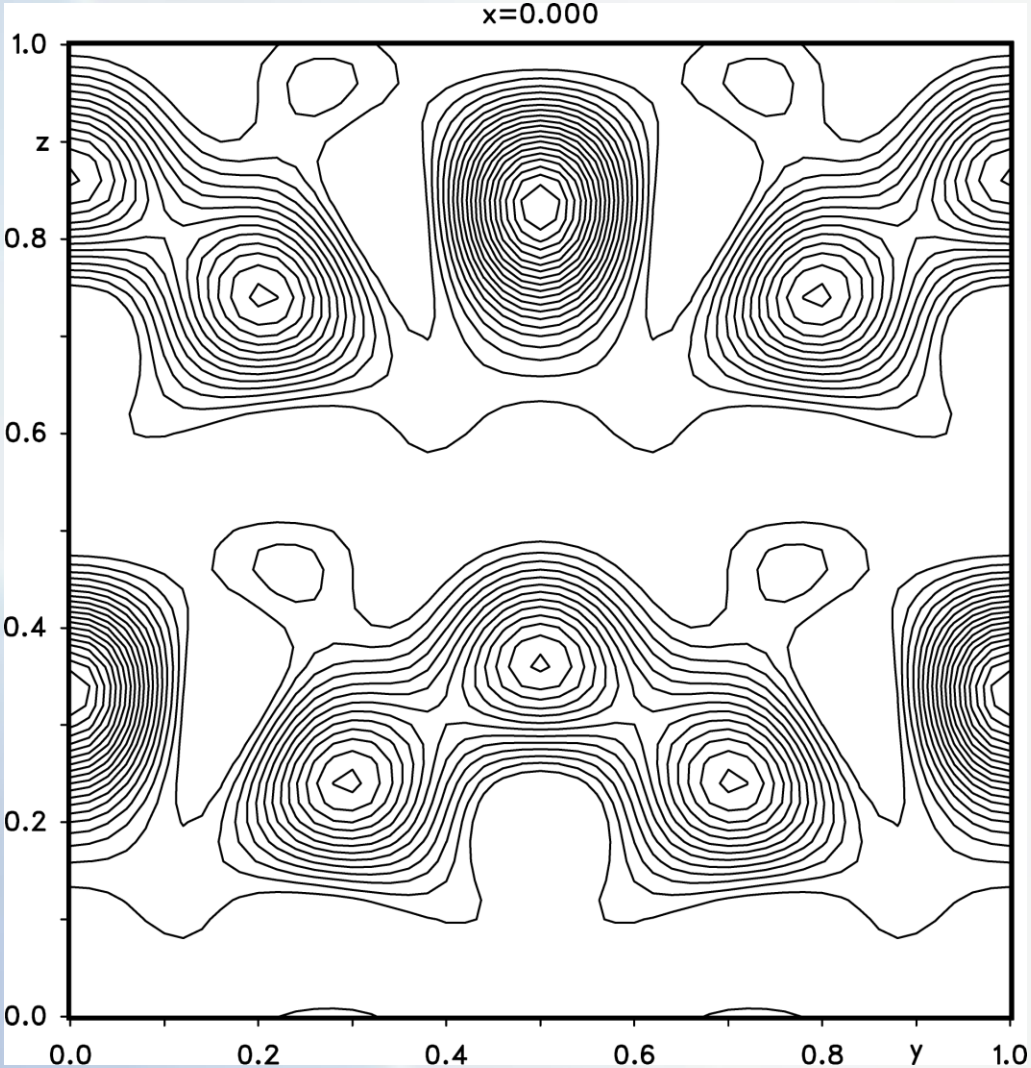
F_{002} + F_{040} + F_{031} + $F_{0\text{-}31}$

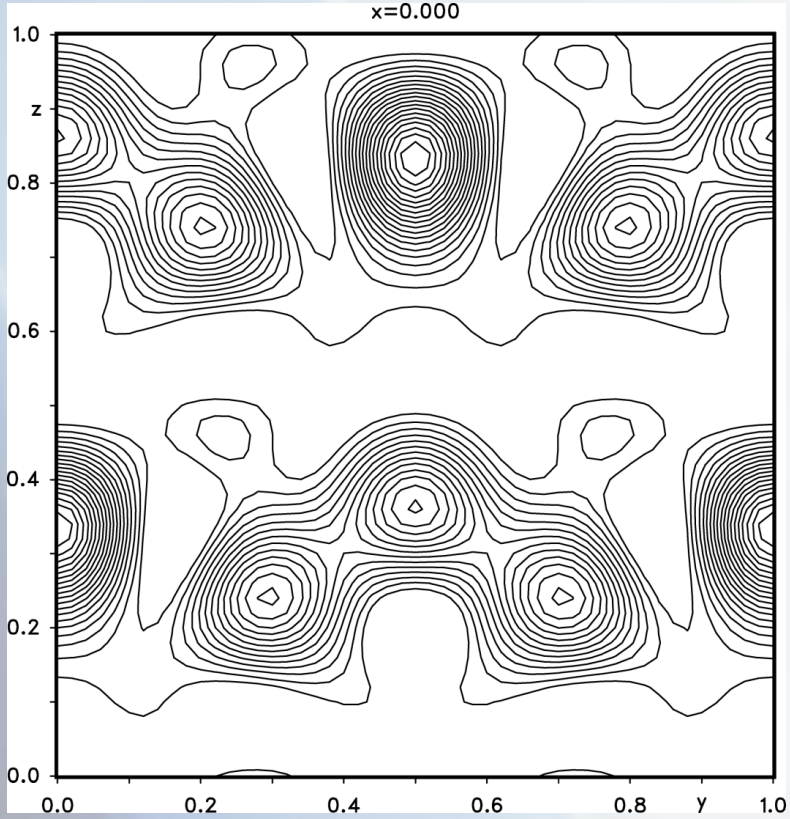

x=0.000
1.0
z
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
y
1.0

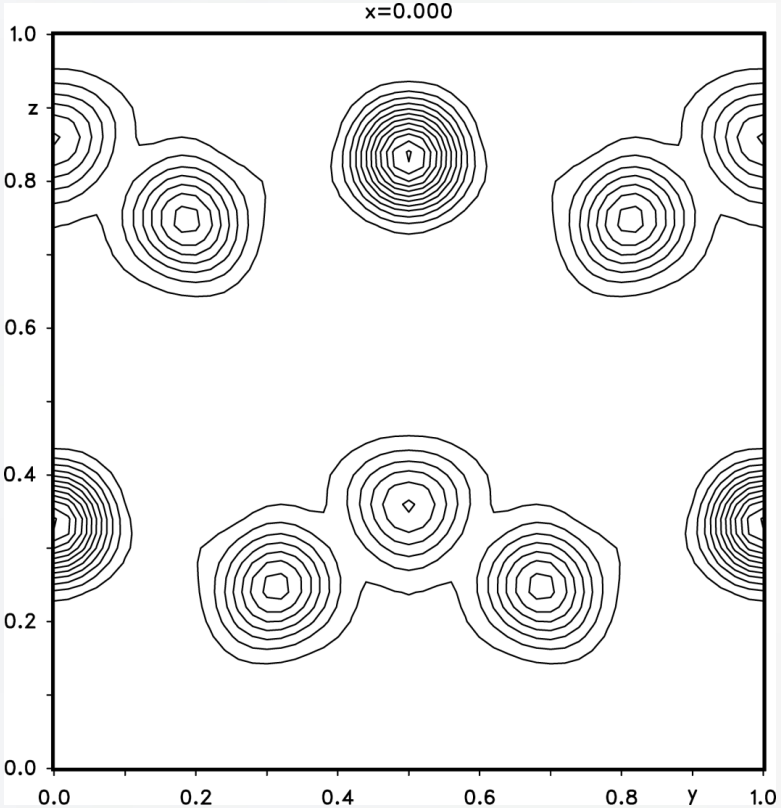

x=0.000
1.0
z
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
y
1.0


x=0.000
1.0
z
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
y
1.0


x=0.000
1.0
z
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
y
1.0


x=0.000
1.0
z
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
y
1.0





Eight reflections (+ their symmetry equivalents) are enough to reproduce the main features of the density map

Fourier transform – crystallographer‘s viewpoint

In diffraction we can routinely measure the *amplitude* of the structure factors, but very rarely the phases.

Hence, the inverse transform

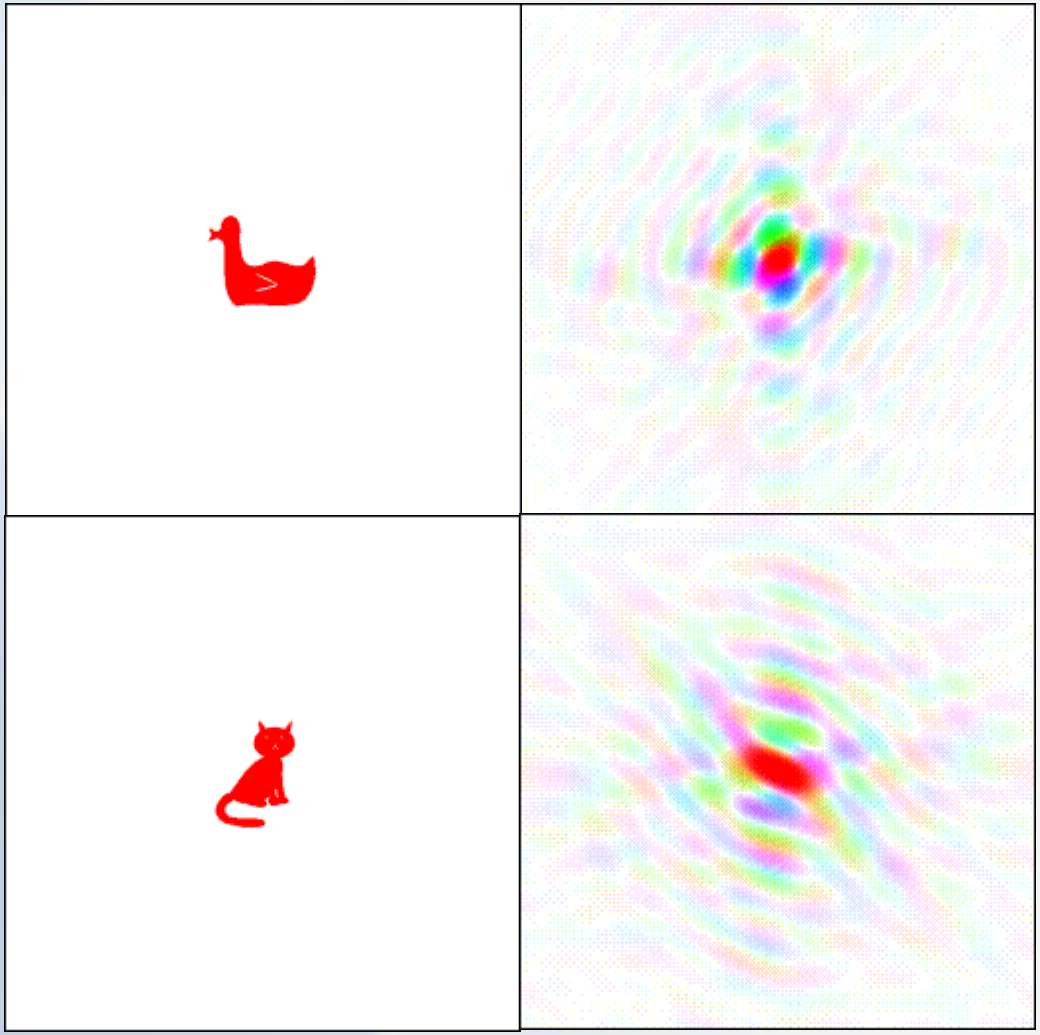
$$\rho(\mathbf{r}) = \sum_{\mathbf{h}=-\infty}^{\infty} F_{\mathbf{h}} e^{-2\pi i \mathbf{h}.\mathbf{r}}$$

cannot be directly performed

crystallopraphic phase problem

Phases and amplitudes

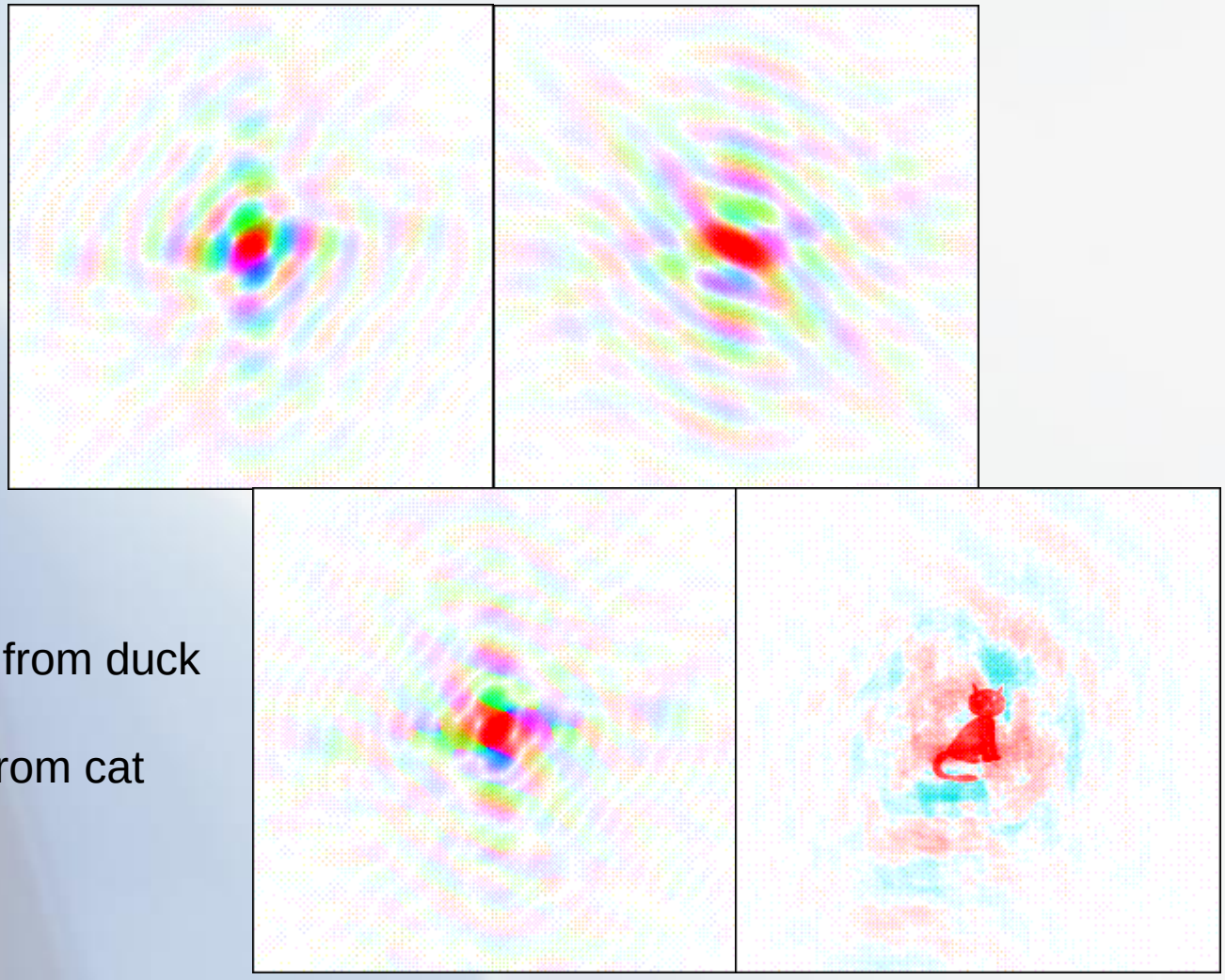
What part of information is carried by amplitudes and what by phases?



http://www.ysbl.york.ac.uk/~cowtan/fourier/fourier.html

Phases and amplitudes

What part of information is carried by amplitudes and what by phases?

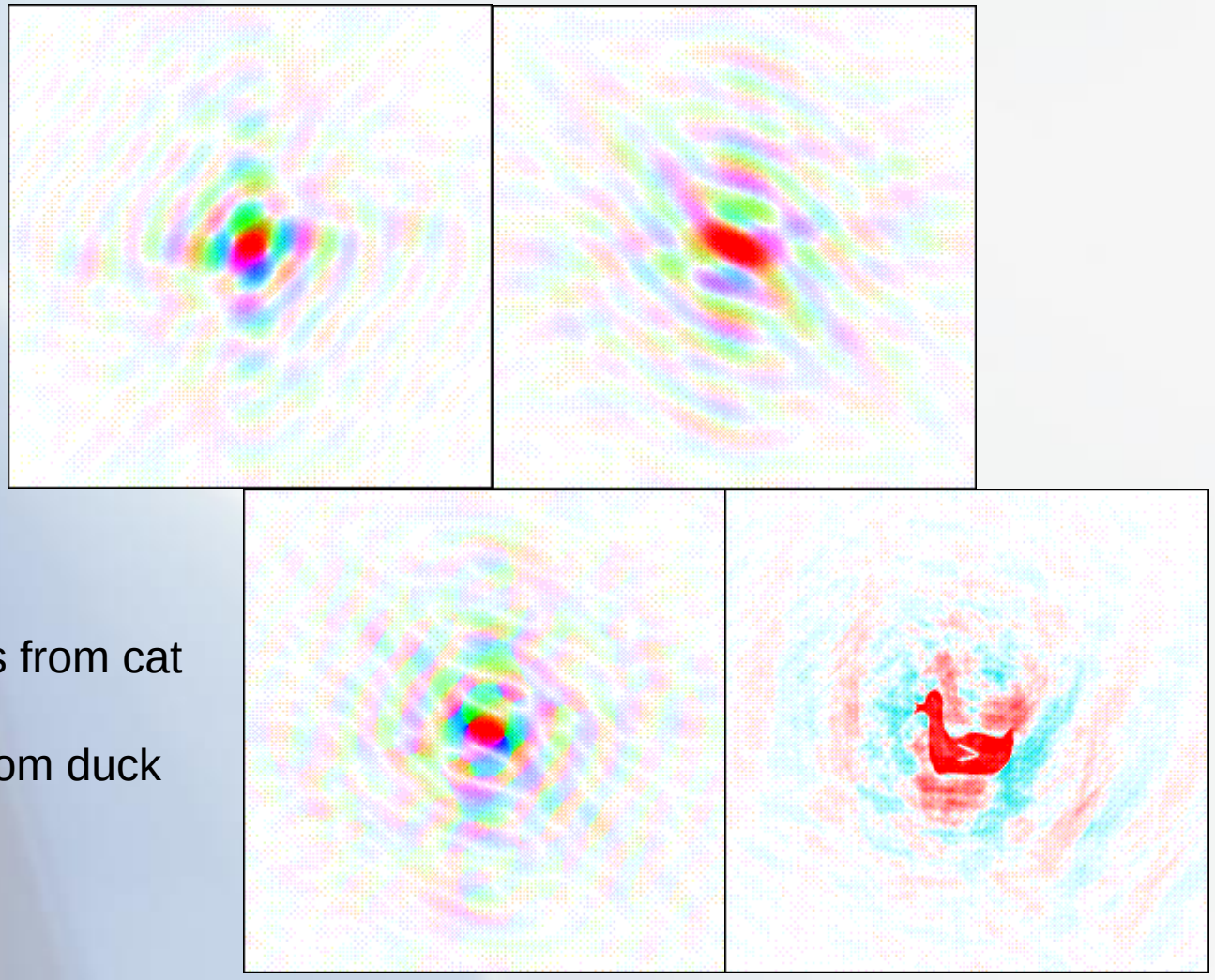


Amplitudes from duck
+
phases from cat

http://www.ysbl.york.ac.uk/~cowtan/fourier/fourier.html

Phases and amplitudes

What part of information is carried by amplitudes and what by phases?



Amplitudes from cat
+
phases from duck

http://www.ysbl.york.ac.uk/~cowtan/fourier/fourier.html

Fourier transform – programmer‘s viewpoint

$$F_{\mathbf{h}} = \int \rho(\mathbf{r}) e^{2\pi i \mathbf{h}.\mathbf{r}} \mathrm{d}V$$

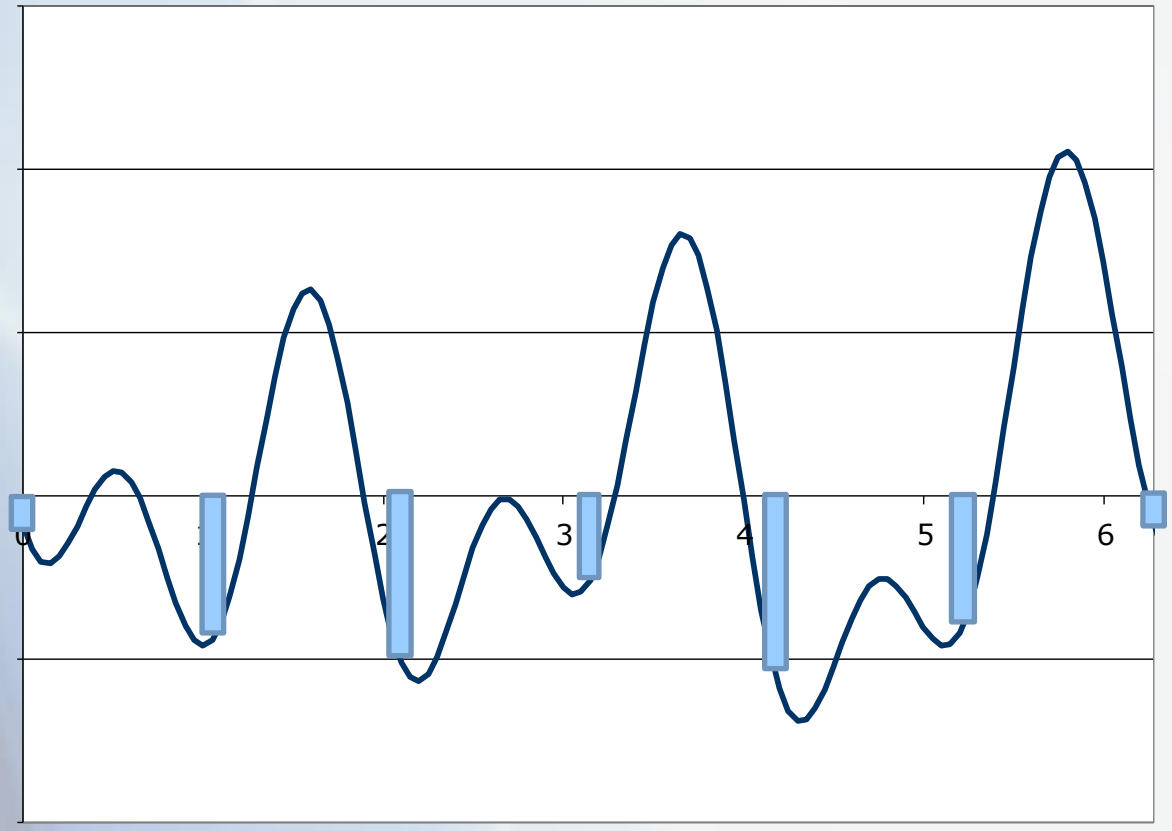
cannot be directly calculated, unless the scattering density is available analytically.

$$\rho(\mathbf{r}) = \frac{1}{V} \sum_{\mathbf{h}=-\infty}^{\infty} F_{\mathbf{h}} e^{-2\pi i \mathbf{h}.\mathbf{r}}$$

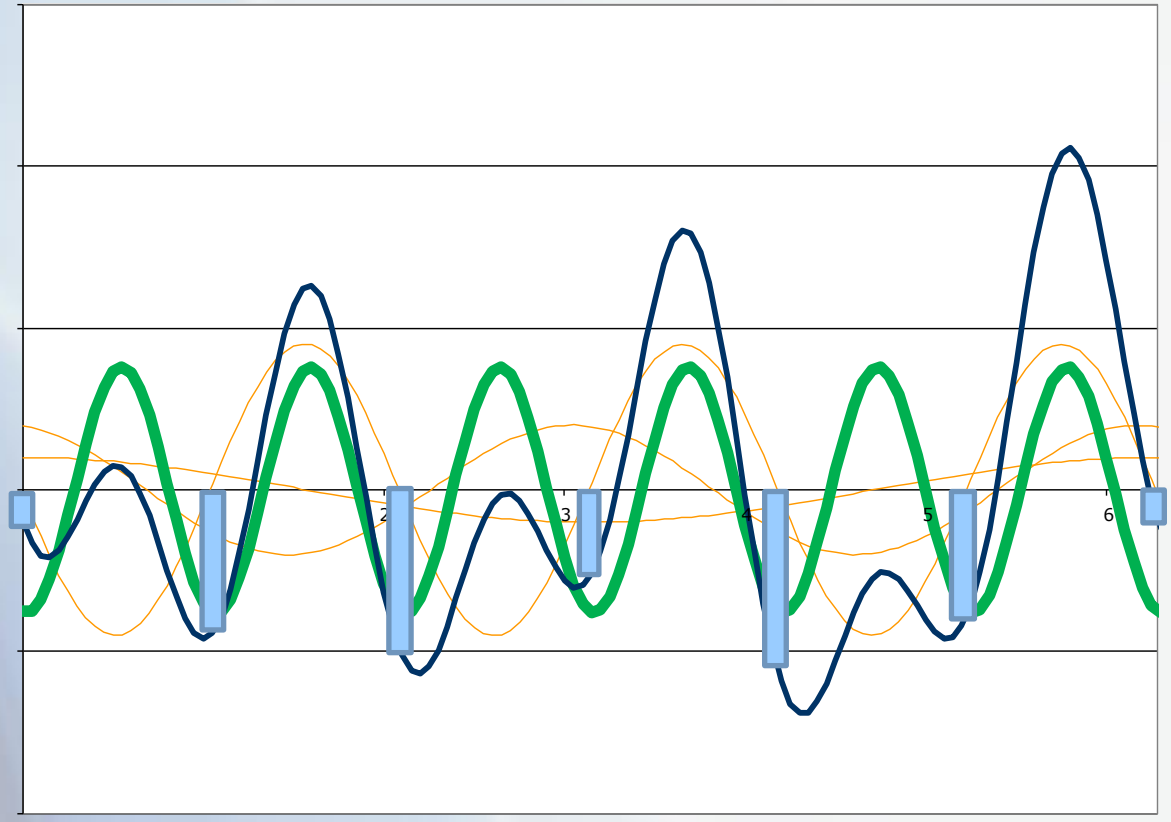
Cannot be calculated, because of the infinite sum.

Solution: sample the scattering density on a regular grid, creating an approximation of the real density with finite number of „parameters“.

Fourier transform – programmer‘s viewpoint

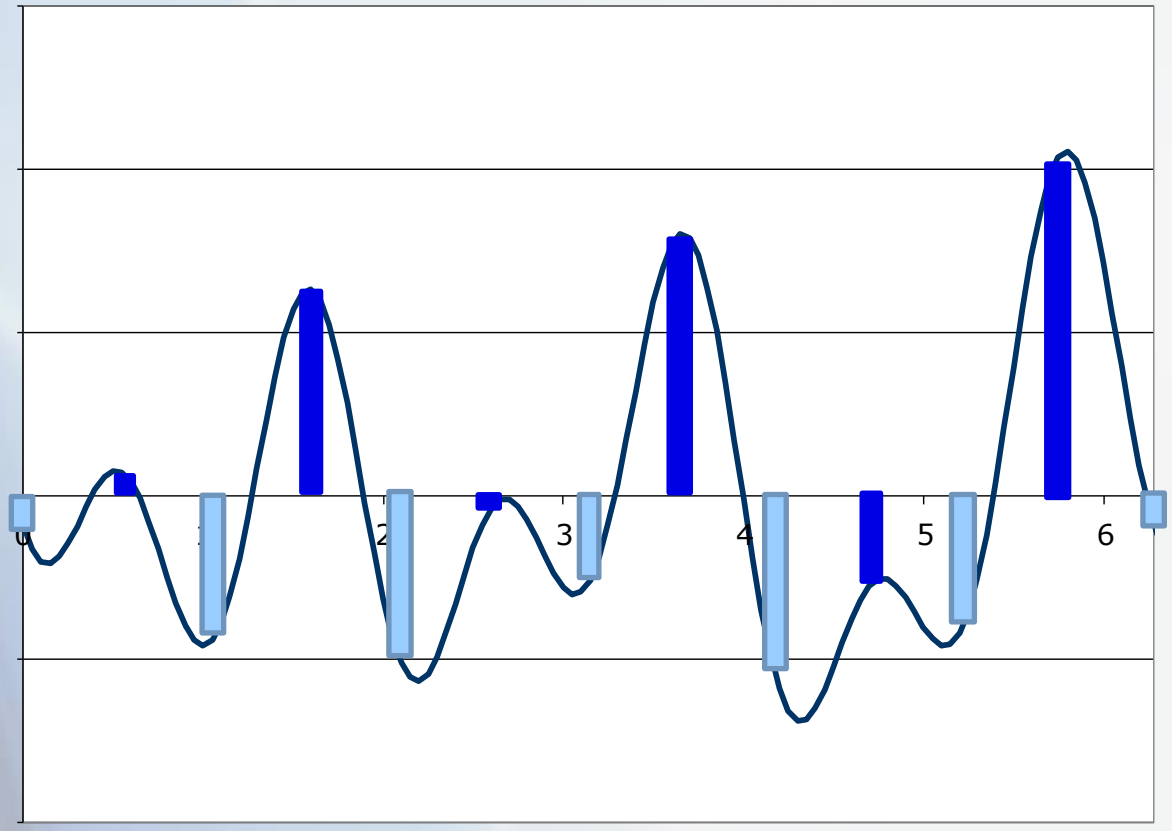


Fourier transform – programmer‘s viewpoint

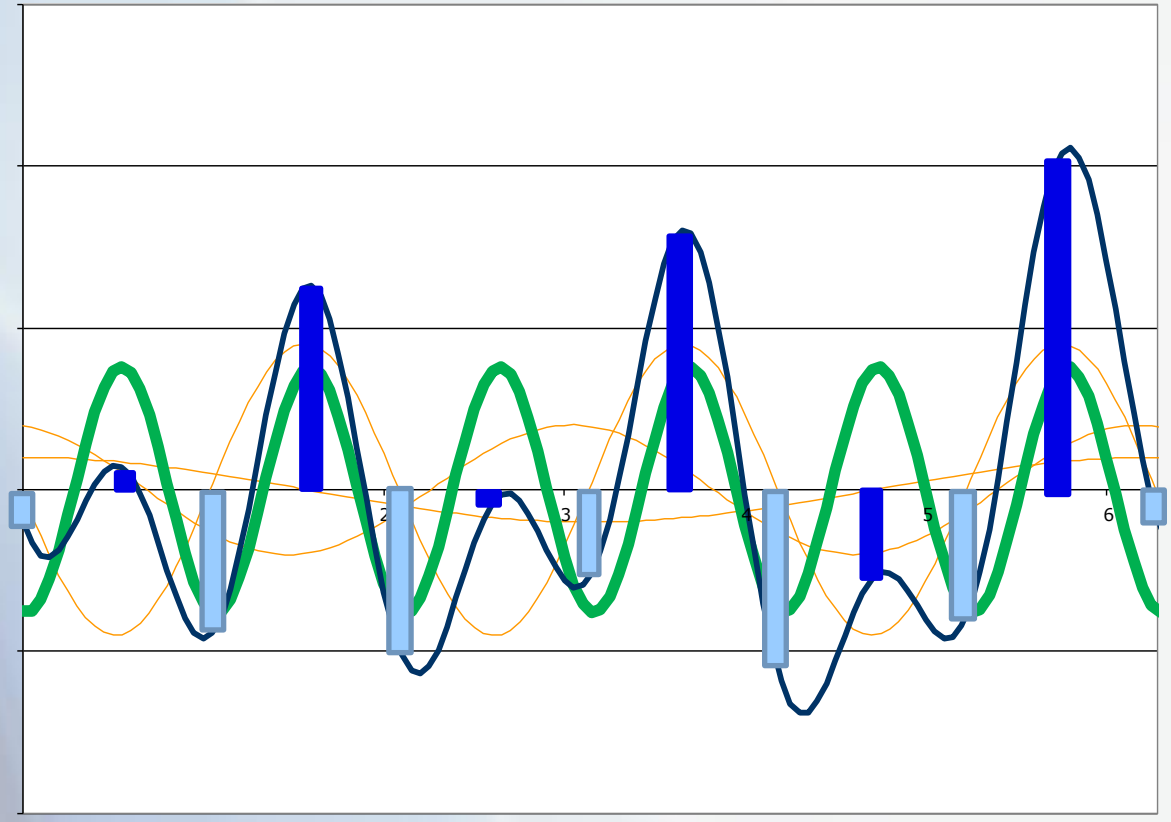


Aliasing: frequencies n and n+kN make equivalent contribution to the discretized function

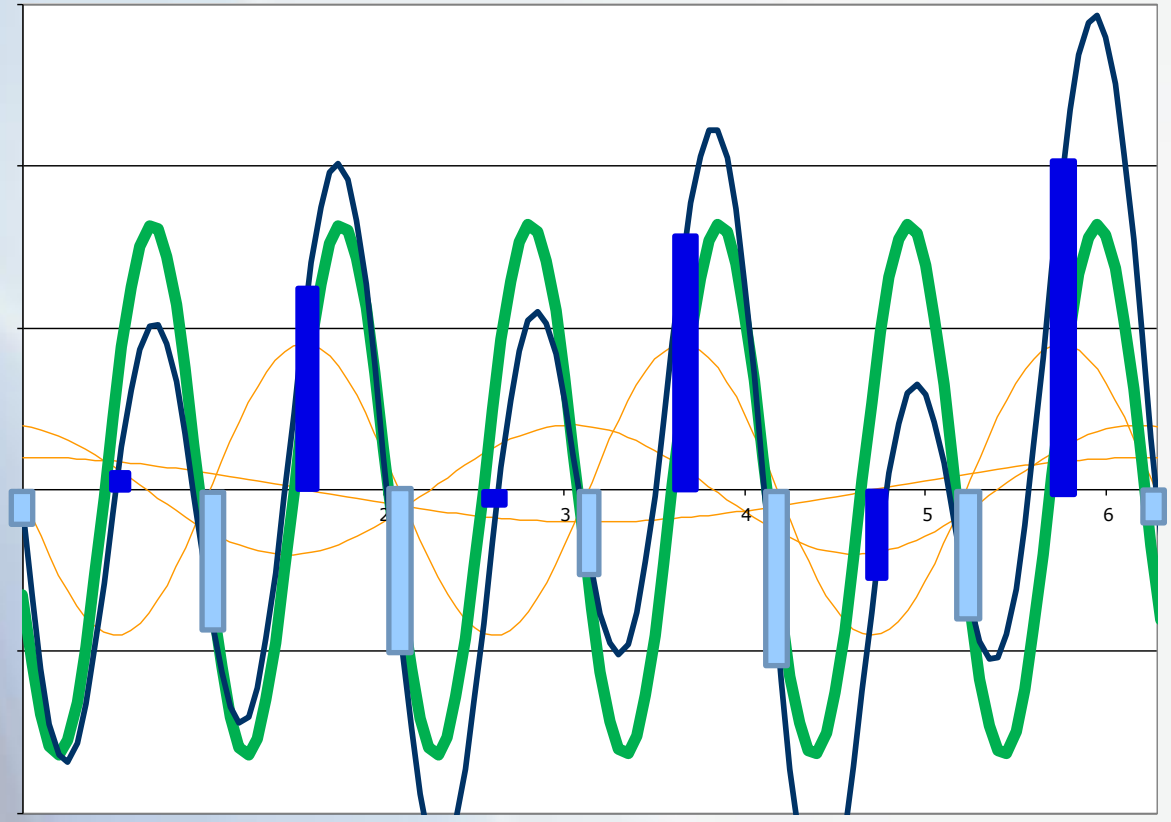
Fourier transform – programmer‘s viewpoint



Fourier transform – programmer‘s viewpoint



Fourier transform – programmer‘s viewpoint



Aliasing: frequency N/2 has only one parameter (phase can be set to 0).

Fourier transform – programmer‘s viewpoint

Fourier transform is in its general („naïve“) implementation a $O(N^2)$ operation.

In 1965, Cooley & Tukey published an algorithm for discrete Fourier transform with O(N.logN) complexity – FFT

First FFT algorithms operated best on grid sizes of the form 2^n. Modern algorithms do also prime-factor FFT.

It is, however, still useful to have grid sizes that can be factored to small prime factors.

Practical aspects
conventions

The sign of the phase factor and the normalization constant in the Fourier transform is a mere convention:

$$F_{\mathbf{h}} = \int \rho(\mathbf{r}) e^{2\pi i \mathbf{h}.\mathbf{r}} \mathrm{d}V \qquad F_{\mathbf{h}} = \frac{1}{V} \int f(x) e^{-2\pi i \mathbf{h}.\mathbf{r}} \mathrm{d}V$$

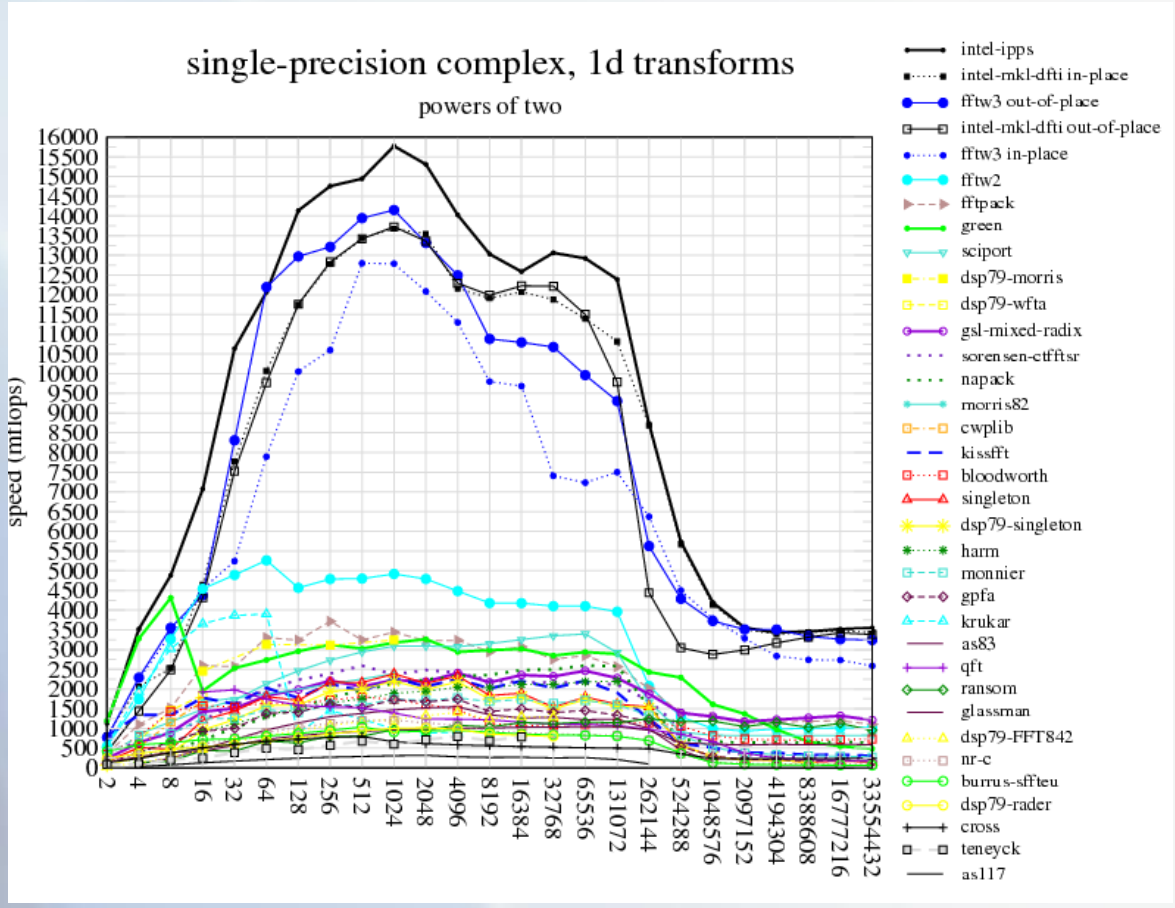
$$\rho(\mathbf{r}) = \frac{1}{V} \sum_{\mathbf{h}=-\infty}^{\infty} F_{\mathbf{h}} e^{-2\pi i \mathbf{h}.\mathbf{r}} \qquad f(\mathbf{r}) = \sum_{\mathbf{h}=-\infty}^{\infty} F_{\mathbf{h}} e^{2\pi i \mathbf{h}.\mathbf{r}}$$

Unfortunately, these conventions are *opposite* in crystallography and in most mathematical literature.

Be careful when using libraries or when you copy others‘ implementations!

Practical aspects
libraries, binding

There are many libraries that contain FFT functionality

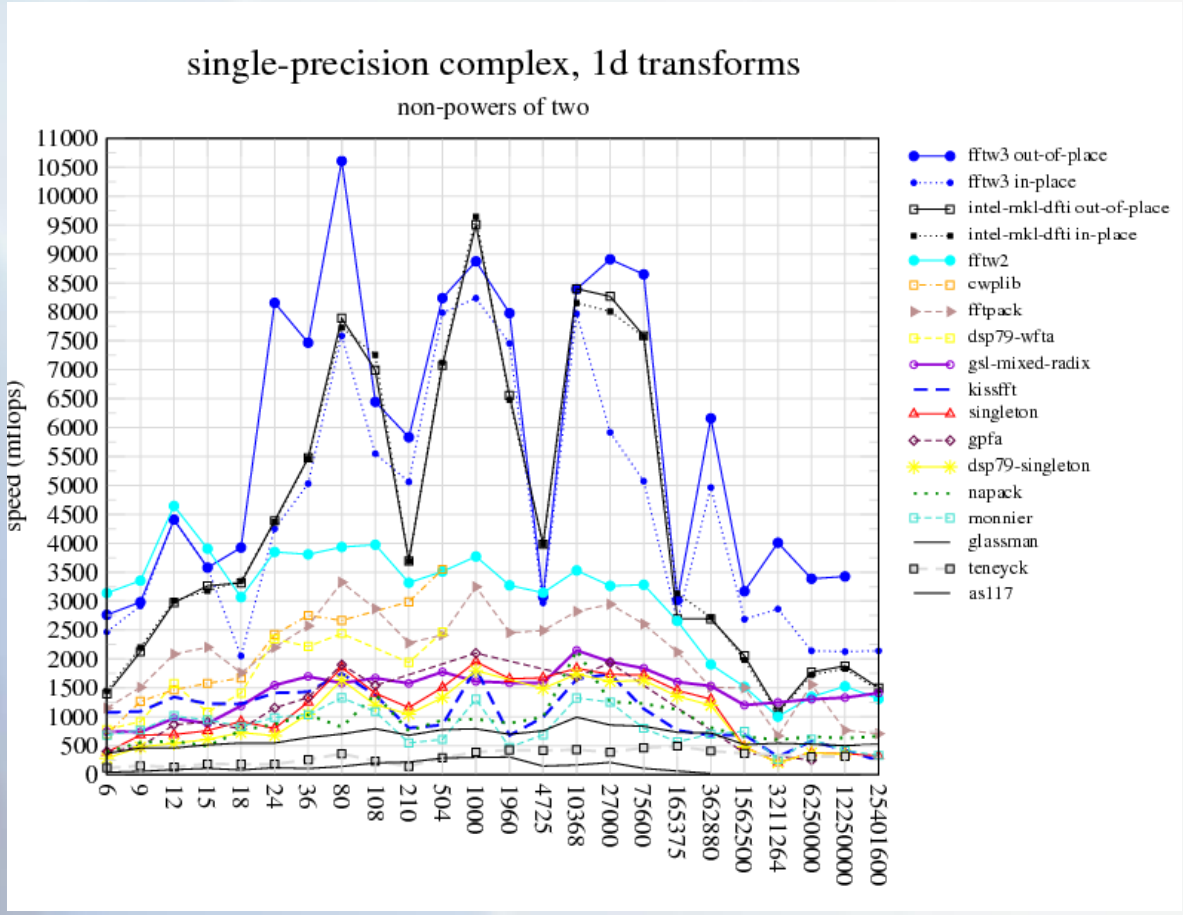


http://www.fftw.org/speed/

3.0 GHz Intel Core Duo, Intel compilers, 64-bit mode

Practical aspects
libraries, binding

There are many libraries that contain FFT functionality



http://www.fftw.org/speed/

3.0 GHz Intel Core Duo, Intel compilers, 64-bit mode

Practical aspects
libraries, binding

FFTW3 – very fast, versatile, free FFT library.
Some features:

- Single and double precision routines
- Real-to-real, real-to-complex, complex-to-complex transforms
- Dedicated 1D, 2D, 3D and nD transforms
- Many different FFT algorithms available
- Options for automatic finding of the best algorithm
- In-place and out-of-place transforms
- Natively called from C, includes wrapper for Fortran, wrappers available for Python, Java, Perl, Ruby, Delphi and more...
- Support for parallel programming
- Binaries available for Windows
- Source code available for compilation on exotic platforms

Practical aspects
sample code

```
      TYPE FFTInstr
         INTEGER(KIND=8) :: plan
         INTEGER :: dir
         REAL :: Norm
       END TYPE FFTInstr


!     Interface to the FFTW subroutine
!*****************************************************************
      SUBROUTINE FFT(instr, rho)
!*****************************************************************
!
      USE SF_Module
      IMPLICIT NONE
      TYPE(FFTInstr) :: instr
      REAL, DIMENSION(:) :: rho

      call sfftw_execute(instr%plan)
      IF (instr%dir==-1) rho=rho/instr%norm

      END SUBROUTINE FFT
```

Practical aspects
sample code

```
!*****************************************************************
    SUBROUTINE MakeFFTPlan(NVox,rho,sf,dir,infastfft,instr)
!*****************************************************************
      USE SF_Module
      IMPLICIT NONE
      INTEGER, PARAMETER :: FFTW_FORWARD=+1, FFTW_BACKWARD=-1
      INTEGER, PARAMETER :: INTEGER, PARAMETER :: FFTW_MEASURE=0, FFTW_ESTIMATE=64
      INTEGER, PARAMETER :: INTEGER, DIMENSION(:) :: NVox
      REAL, DIMENSION(:) :: rho, sf
      INTEGER :: Method, dir
      LOGICAL :: InFastFFT
      TYPE(FFTInstr) :: instr

      IF (InFastFFT) THEN
        Method=FFTW_MEASURE
      ELSE
        Method=FFTW_ESTIMATE
      ENDIF
      instr%dir=dir
      IF (dir==r2cFT) THEN
        call sfftw_plan_dft_r2c(instr%plan,size(NVox),NVox,rho,sf,method)
        instr%norm=1.
      ELSEIF (dir==c2rFT) THEN
        call sfftw_plan_dft_c2r(instr%plan,size(InNVox),InNVox,insf,inrho,method)
        instr%norm=product(InNVox)
      ENDIF
      IF (instr%plan==0) CALL StopProgram('Error, cannot initiate the FFT routine.')

    END SUBROUTINE MakeFFTPlan
```