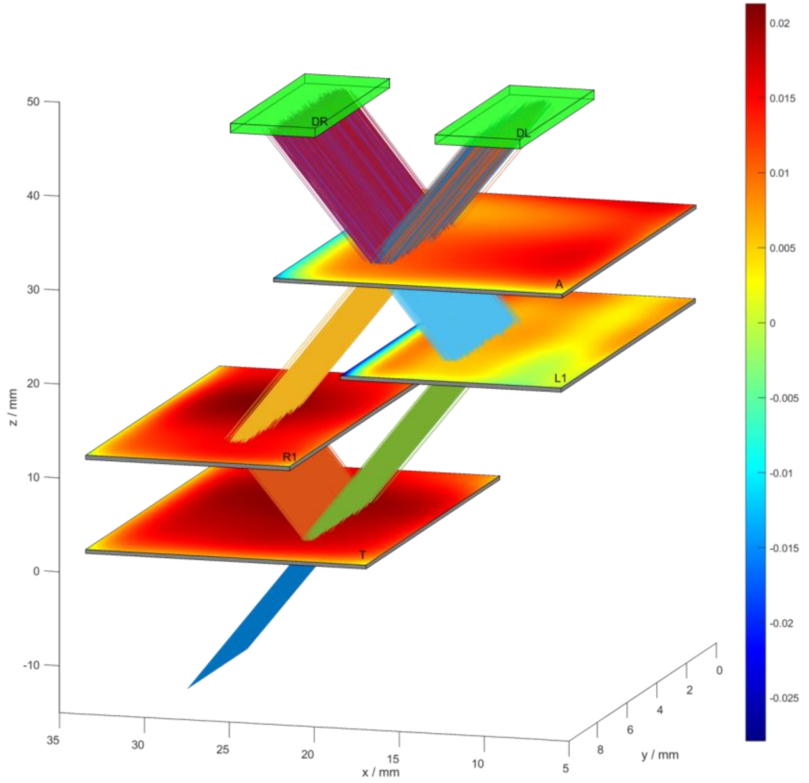
Virtual X-Ray Interferometer for Estimation of Systematic Effects in The Determination of The Lattice Parameter of Si-28

Birk Andreas (4.25)

Physikalisch-Technische Bundesanstalt (PTB)



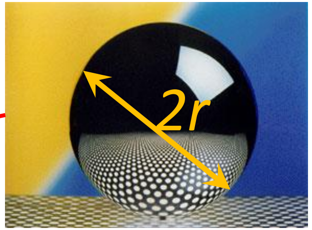
Outline:

1. Motivation
2. Physical background
3. Implementation
4. Results
5. Summary

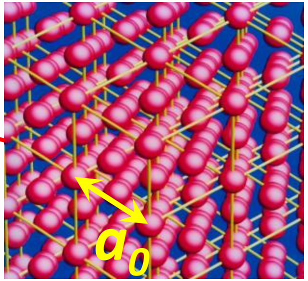
Motivation: X-Ray Crystal Density (XRCD) Method

^{28}Si ingot

$$N_A = \frac{V_{\text{sphere}}}{v_{\text{atom}}} \cdot \frac{M_{\text{Mol}}}{m_{\text{sphere}}} = \frac{\frac{4\pi r^3}{3} \cdot M_{\text{Mol}}}{\frac{a_0^3}{8} \cdot m_{\text{sphere}}}$$



Sphere interferometer

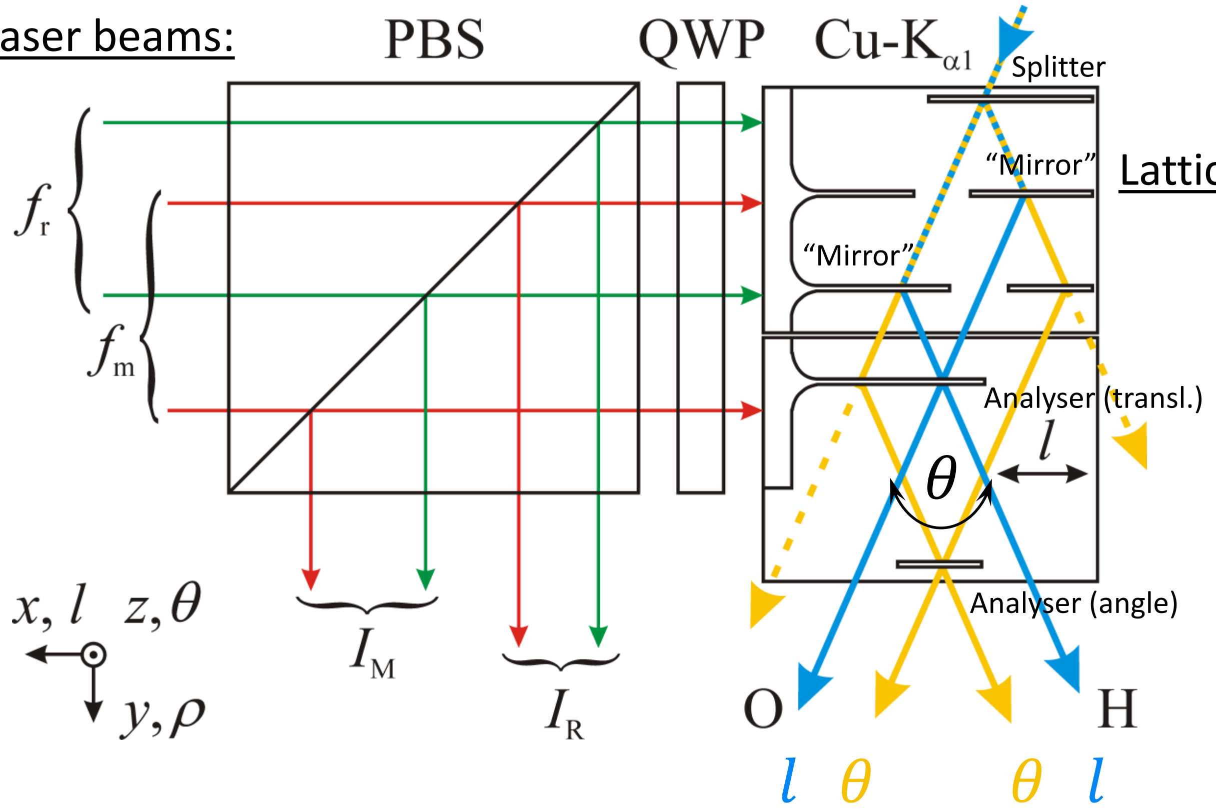


Combined optical and x-ray interferometer (COXI)

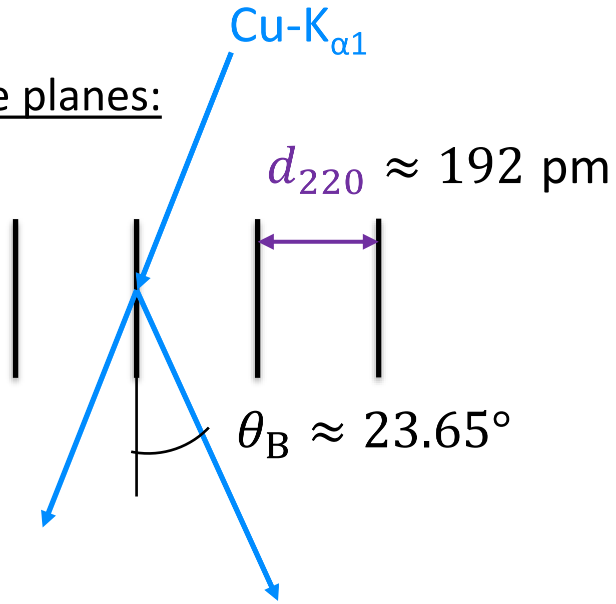
- International **A**vogadro **C**oordination contributed to **SI-Revision 2019**
- **Today:** one of two realisation methods for the kilogram
- **But:** only one lattice parameter measurement by INRiM
- **Therefore:** at least one additional independent measurement at PTB

Physical Background: Working Principle of The COXI

Laser beams:

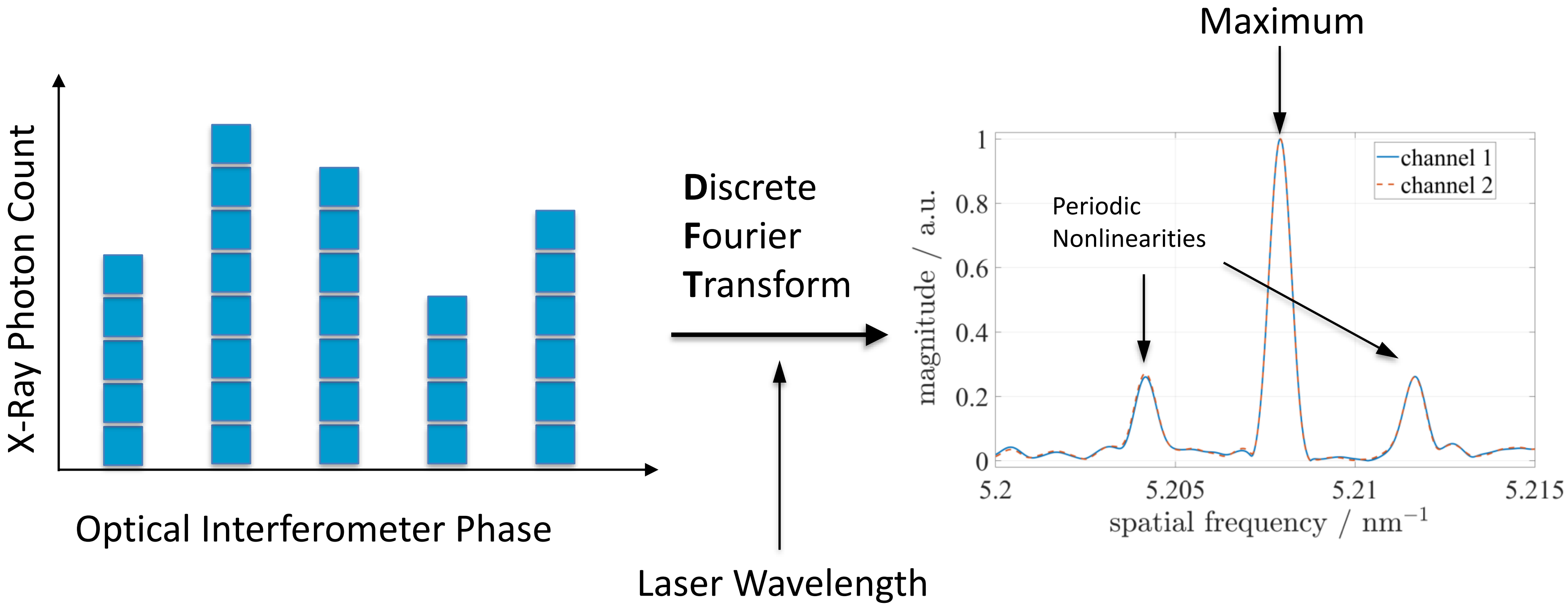


Lattice planes:



$$a_0 = d_{220}\sqrt{8}$$

Evaluation of Measurement Data



B. Andreas and U. Kuetgens, "A continuously scanning separate-crystal single-photon x-ray interferometer," MST **31,** 115005 (2020)

Systematic Effects by Thickness Variation And Lateral Drift

Meas. Sci. Technol. **31** (2020) 115005 B Andreas and U Kuetgens

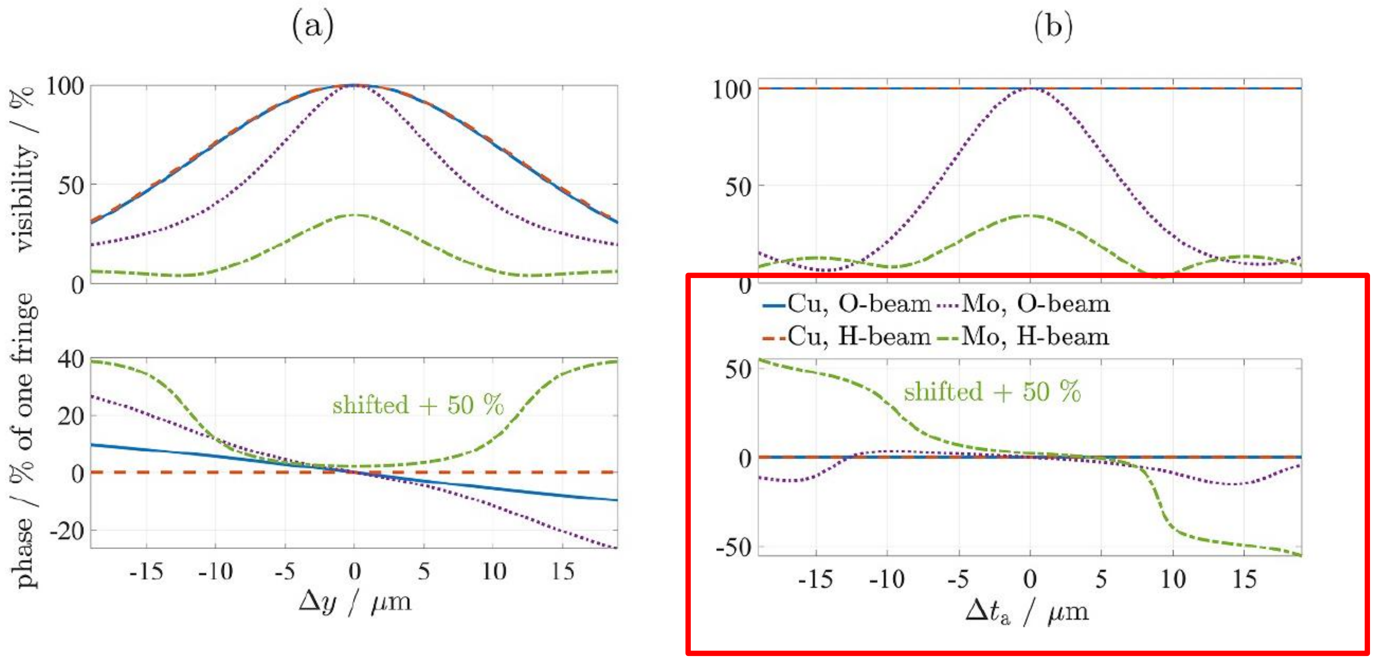
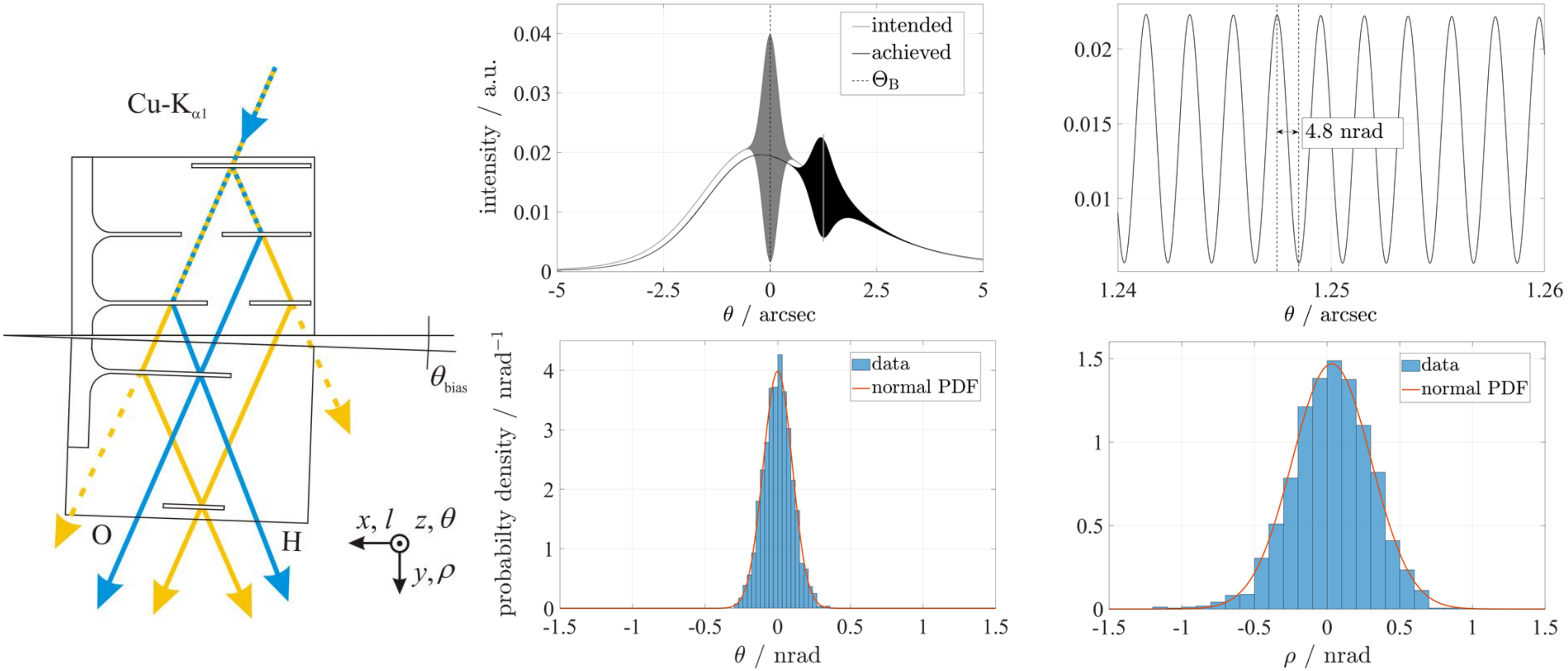


Figure 6. Dependence of fringe visibility and phase of an LLL XRI with 0.4 mm lamellae on defocus (a), i.e. deviation Δy from the optimal mirror-analyzer distance, and the thickness variation of the analyzer lamella Δt_a (b) for O- and H-beam (cf. figure 3(a)) of Cu- and Mo-K_α radiation, respectively, calculated by dynamical x-ray diffraction theory [8, 38, 39]. The phase plots of the Mo-K_α H-beams have been shifted by the denoted amount to improve the readability.

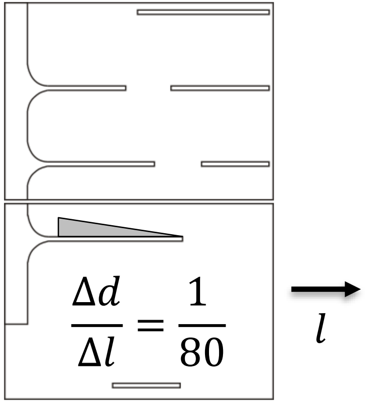
Sub-Nanoradian Angular Control and Offset Angle θ_{bias}



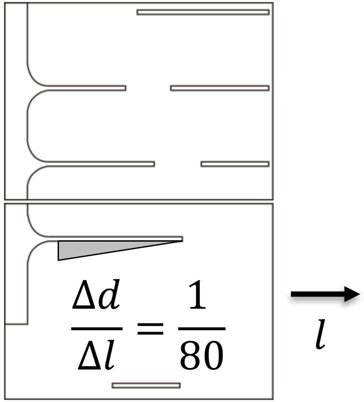
B. Andreas and U. Kuetgens, "A continuously scanning separate-crystal single-photon x-ray interferometer," MST **31,** 115005 (2020)

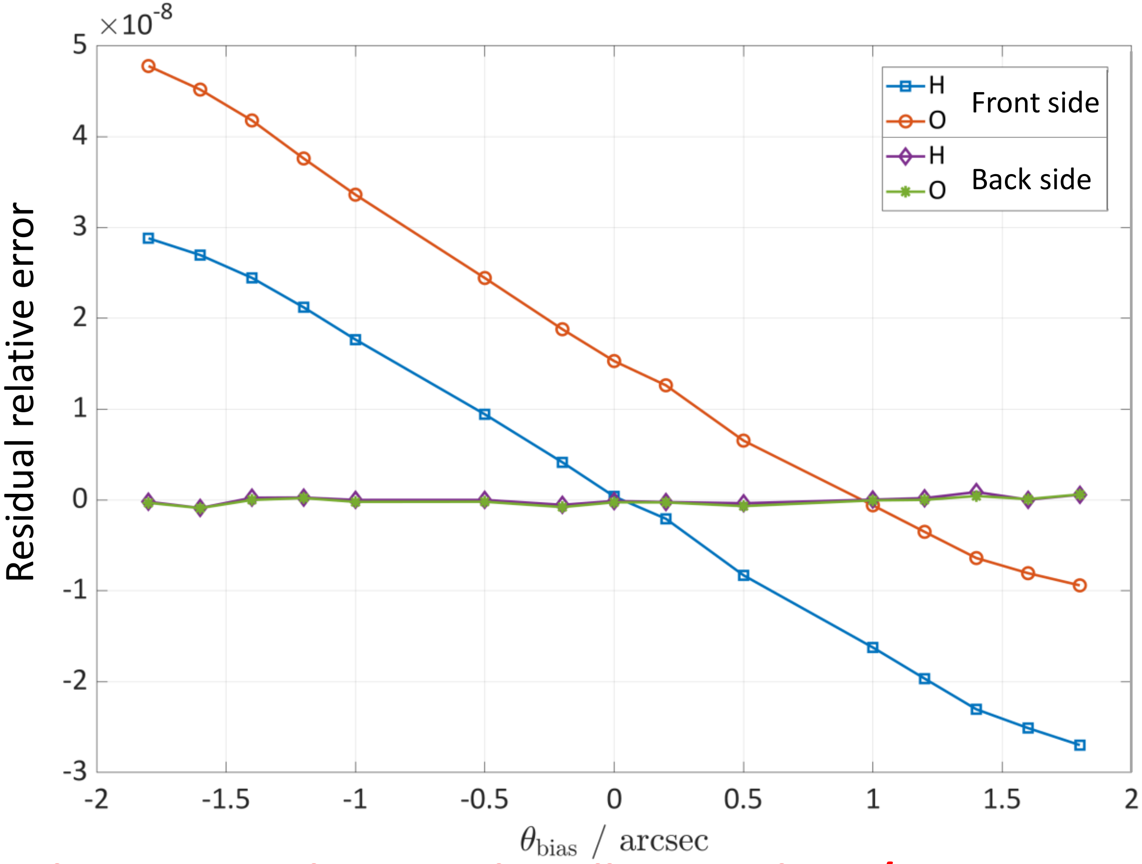
Impact of Thickness Variation (Dynamical Diffraction Simulation)

Front side:



Back side:

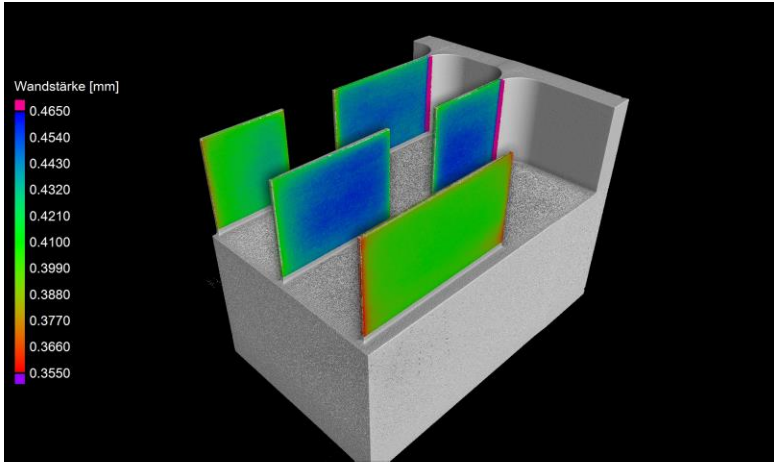


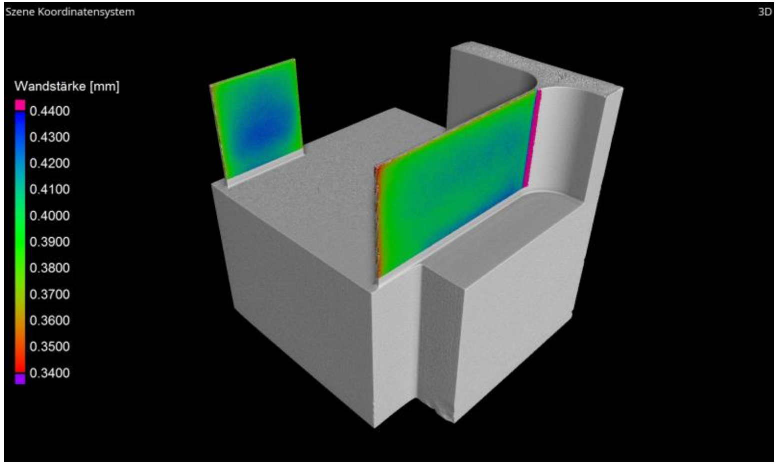


Solution: re-etch certain lamellas on splitter/mirror crystal!

Iterative Etch Robot by Electroless Cu Plating And $FeCl_3$-Etching

X-Ray Computer Tomography Measurements (René Laquai, 5.34)

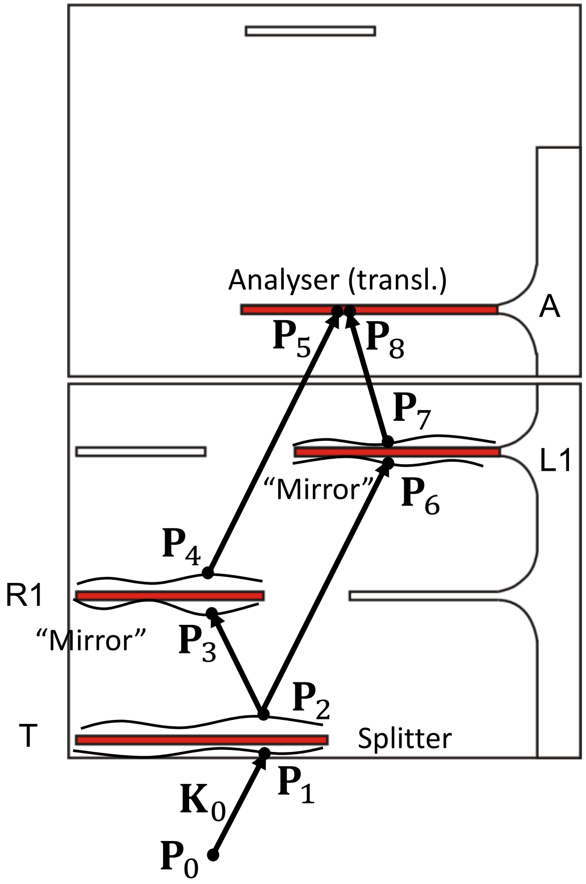




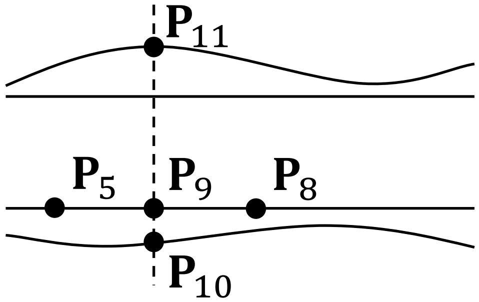
Extraction of all topographies (René Laquai, Josef Frese)

Main Motivation for Simulation

Implementation: Main Simulation Concept



1. $\mathbf{P}_0$ and $\mathbf{K}_0$ are given.
2. A path agent finds the points $\mathbf{P}_1$ to $\mathbf{P}_8$.
3. From $\mathbf{P}_5$ and $\mathbf{P}_8$ the mean position $\mathbf{P}_9$ is calculated.
4. From $\mathbf{P}_9$ and the local topography $\mathbf{P}_{10}$ and $\mathbf{P}_{11}$ are calculated.



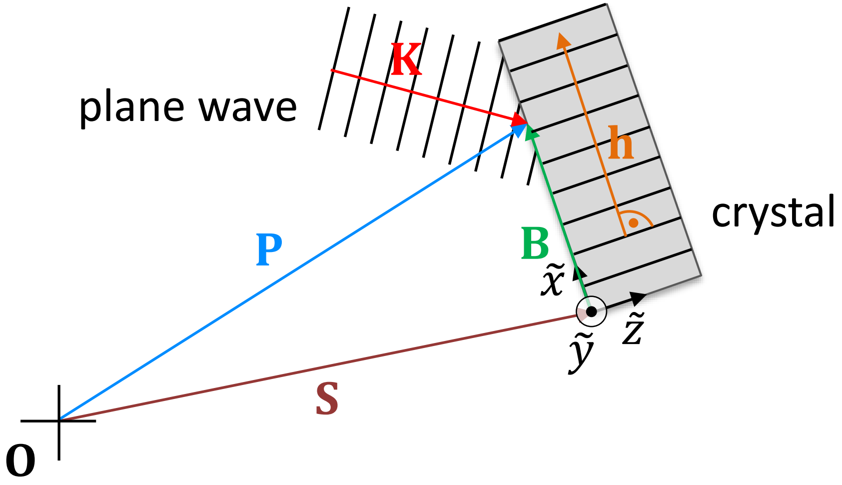
5. The points $\mathbf{P}_1$ to $\mathbf{P}_7$ as well as $\mathbf{P}_{10}$ and $\mathbf{P}_{11}$ are used for calculating the phases of the plane waves used for dynamical diffraction theory computations.

Phase Factors

Plane-wave phase-factor: $PF_{\mathrm{PW}} = \exp(-\mathrm{i}\mathbf{K}\cdot\mathbf{P})$

Local susceptibility: $\chi_{\mathrm{h}}(\mathbf{B}) = \chi_{\mathrm{h}}(0)PF_{\mathrm{h}} = \chi_{\mathrm{h}}(0)\exp(-\mathrm{i}\mathbf{h}\cdot\mathbf{B})$

Cause of translational phase shift!



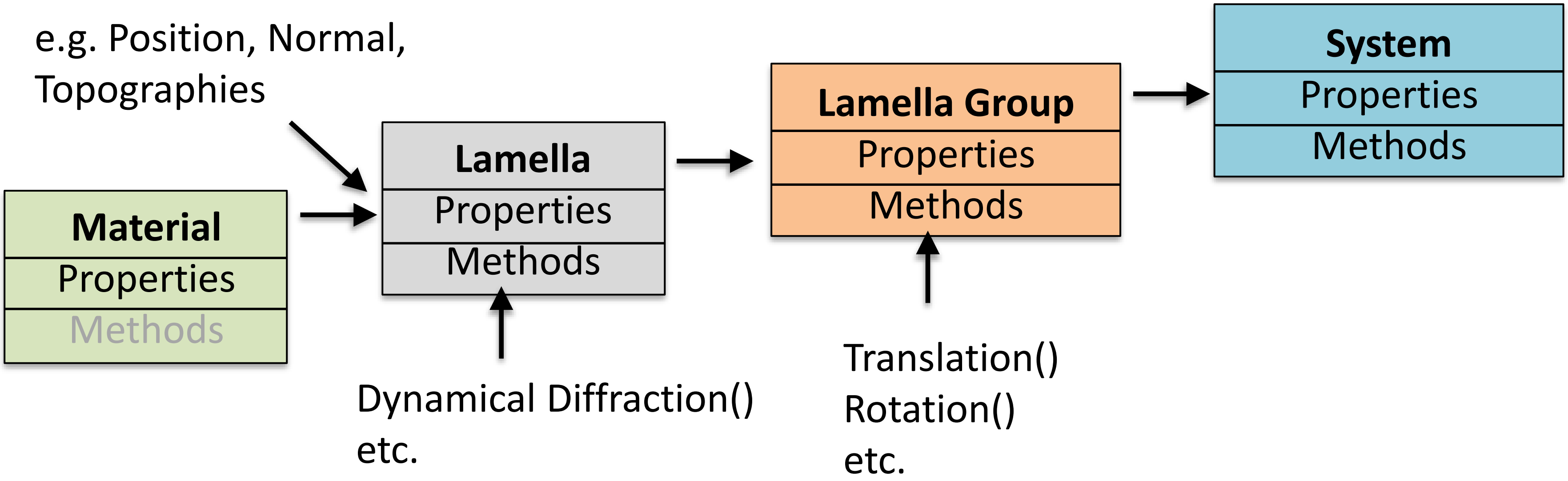
Object-Oriented Programming: Basic Concept

System of (real) entities

↓

Classes of **objects** with inherent **properties** and **methods**

Object-Oriented Programming: Example



⇒ Entity interrelations are mapped to class definitions!

UML of a pre-Alpha Implementation

Lamella

- Handle to Parent
- Index in Parent.List
- Name
- Dimensions
- Topography structs Front and Back:
  - polynomial coeff.
  - local pivot
  - dimensions
- Broadening at exit face (true)

- DynDiffPropagation (each ray as plane wave)

LamellaGroup

- Local Coordinate Vectors:
  - PivotVector
  - Nx, Ny, Nz
- Material parameters
- Reciprocal lattice vector h
- LamellaList

- Add/remove Lamella
- MoveTo (rel. to PivotVector)
- NewNormal (of a Lamella)

System

- LamellaGroupList
- DetectorList

- Add/remove LamellaGroup
- Add/remove Detector
- Trans/rot LamellaGroup

Si (220) Laue Cu

- Specific material parameters
- Wavelength of Cu K alpha1
- Specific reciprocal lattice vector h

- No methods (just storage)

Detector

- Name
- Position
- Normal
- Aperture
- RayArrayList

- Store RayArray
- Clear detector
- Interfere stored RayArrays
- Calculate contrast
- Calculate phase

Camera

- Name
- Position
- Normal
- Aperture
- RayArrayList

- Store RayArray
- Clear detector
- Interfere stored RayArrays

RayArray

- Array of ray positions, first ray has mean position
- Array of ray directions, first ray has mean direction
- Array of optical path lengths, first ray has mean optical path length
- Array of global electrical field components (mixed polarization with 1/sqrt(2) normalization)
- Wavelength

- FreeSpacePropagation (Named Target)
- FreeSpacePropagation (Pivot,Normal,ref. index)

Main

- Define system
- Define initial RayArray parameters
- For index = 1 to N
  - Make local system copy
  - Manipulate local system copy (index)
  - Generate initial RayArray
    - Problem-specific propagation code
  - Store Results into observables arrays
- End
- Generate Reports

Utility Functions:

- Figure=**PlotSystem**(System)
- b=**DupRayArray**(a)
- dim=**GetDim**(Front,Back)
- FoundObj=**FindEntity**(Sytem,Name)
- [RMS,PTV]=**RetraceRayArray**(RayArray,BiPolyFitCoeffs,Pivot,Normal,PrecisionTarget (1e-16))

Implementation in MATLAB

Approx. 3000 lines of code in 30 m-files (8 class definitions, 20 functions and 2 scripts):

XRT_Camera.m	XRT_DupLamellaGroup.m	XRT_GetDim.m	XRT_PathAgent.m
XRT_RetraceRayArray.m	XRT_CrysTrans.m	XRT_DupRayArray.m	XRT_GetResults.m
XRT_PathExtractor.m	XRT_Si220LaueCu.m	XRT_Detector.m	XRT_DupSystem.m
XRT_Lamella.m	XRT_PlotSystem.m	XRT_Si220LaueCu2.m	XRT_DupDetector.m
XRT_EvalSystem.m	XRT_LamellaGroup.m	XRT_MainXRIpar.m	XRT_RayArray.m
XRT_System.m	XRT_DupLamella.m	XRT_FindEntity.m	XRT_Rayplot.m

BiPolyDer.m BiPolyFit.m BiPolyMat.m BiPolyVal.m

CTD_Load.m CTD_LoadBatch.m

2D-polynomials up to 6^{th} order incl. all mixed terms

CT-Data import

The evaluation is done with the same functions that are used for the real measurements (not shown here).

Object-Oriented Programming in MATLAB: An Example

```
classdef dict < handle
%  Alternative implementation of a dictionary (e.g. for older MATLAB
%  versions up to R2022a) based on containers.Map.
%
%  Syntax and arguments:
%
%   D=dict(KeyList,ValueList);
%
%   Creates the dictionary with the cell arrays KeyList and ValueList.
%
%  Properties:
%
%   Nothing to see here, keep moving!
%
%  Methods:
%
%   value=D.Get(key);
%
%   D.Set(key,value);
%
%   D.Add(key,value);
%
```

Handle classes…

…allow syntax like this

Object-Oriented Programming in MATLAB: An Example

```
%   [key,value]=D.Pop;
%    Gets the last key-value pair and removes it from the dictionary.
%
%   D.Remove(key);
%    Removes the key and its value from the dictionary.
%
%   L=D.Size;
%    Assigns the length of the dictionary to L;
%
%   [Keys,Values]=D.List(printflag(optional));
%    Retrieves the Keys and Values of the dictionary as cell arrays. If no
%    printflag (any value of any type) is given, the contents are
%    displayed.
%

    properties (Access = private)
        D
    end

    methods
        function obj=dict(keys,values)
```

(Access = private) ← Its public without this!

function obj=dict(keys,values) ← Constructor

Object-Oriented Programming in MATLAB: An Example

```
methods
    function obj=dict(keys,values)

        if isstring(keys)
            keys=convertStringsToChars(keys);
              .
              .
              .
    end
              .
              .
              .
    function R=Get(obj,key)
        try
            R=obj.D(key);
        catch
            error('Key not in dictionary!');
        end
    end
              .
              .
              .
```

Constructor

A further method

Object-Oriented Programming in MATLAB: An Example

```
                        .
                        .
                        .
    end
end
```

Here is (optional) room for helper functions in the same file.

Object-Oriented Programming in MATLAB: An Example

Command Window

```
>> D=dict({1,2,3,4,5},{'Horst',42,true,"Whatever",[1 2;3 4]});
>> D.List
    {[1]}    {[2]}    {[3]}    {[4]}    {[5]}

    {'Horst'}    {[42]}    {[1]}    {["Whatever"]}    {2×2 double}


ans =

  1×5 cell array

    {[1]}    {[2]}    {[3]}    {[4]}    {[5]}

>> D.Size

ans =

     5

>> [k,v]=D.Pop

k =

     5
```

← Create an instance

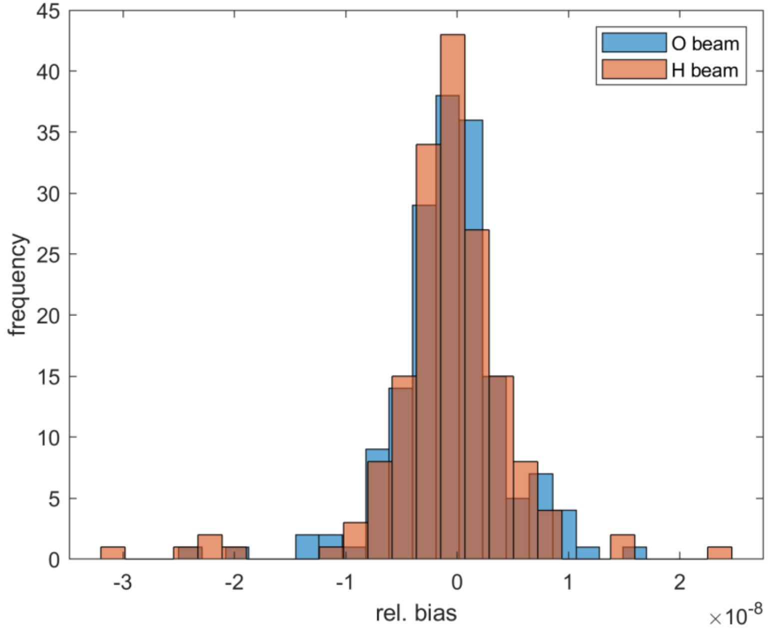
Object-Oriented Programming in MATLAB: An Example

```
v =

     1     2
     3     4

>> D.Size

ans =

     4

>> D.Get(4)

ans =

    "Whatever"

>> D.D
No public property 'D' for class 'dict'.

>>
```

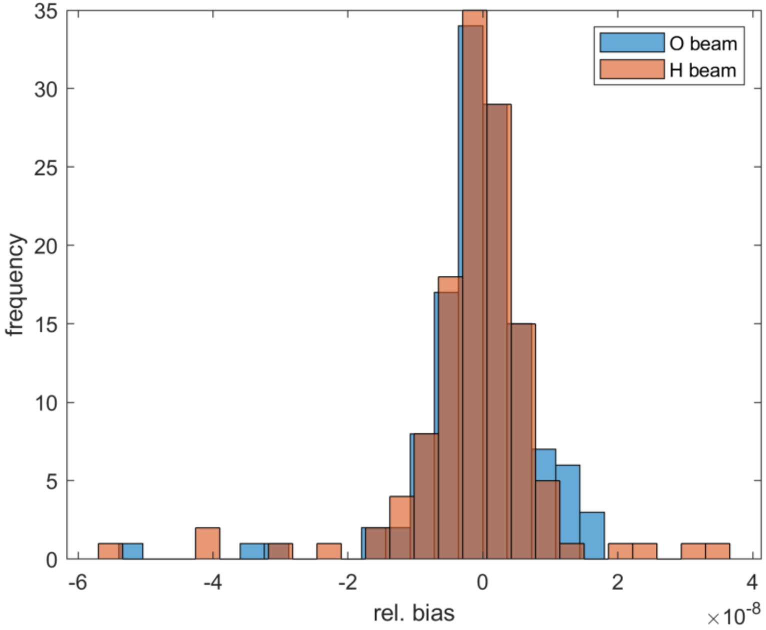
…because its private!

Monte-Carlo Results (1.9 h per run)

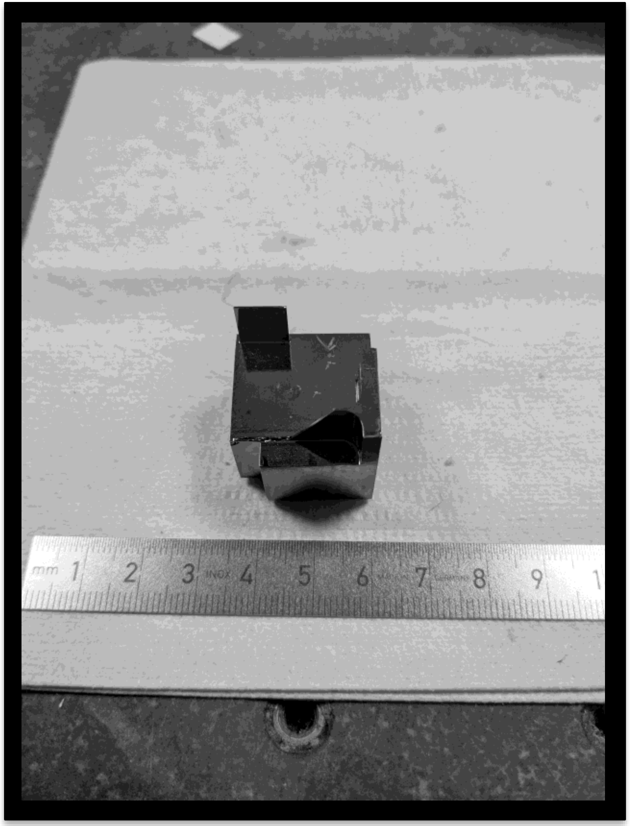
Without offset angle (166 Runs):
O beam: -6.4E-10 ± 3.9E-10
H beam: -9.0E-10 ± 4.6E-10



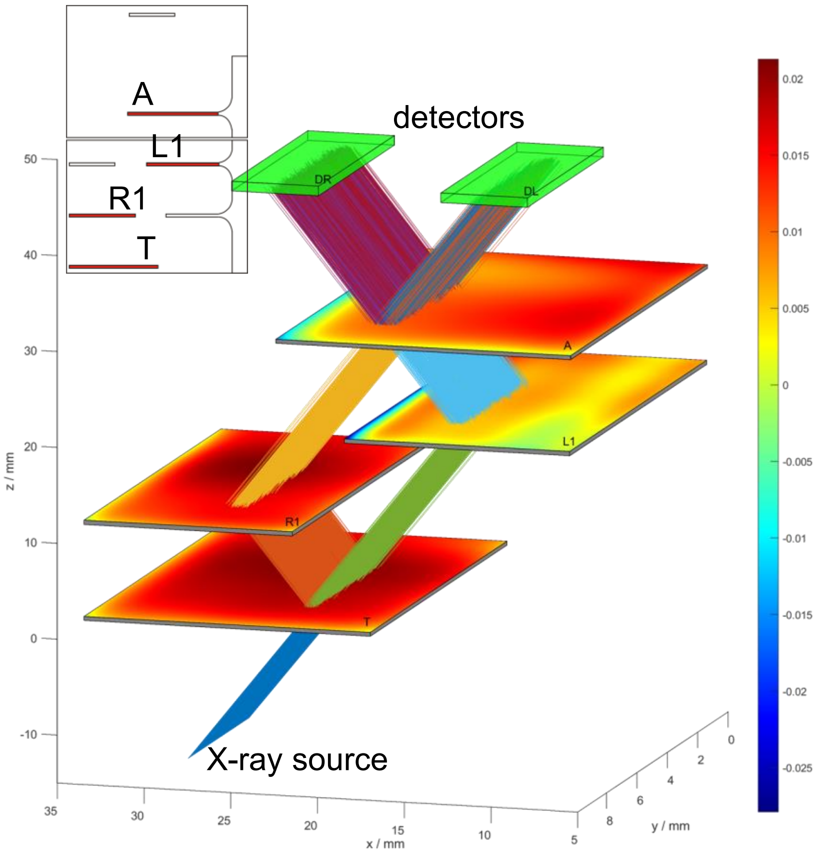
With 1.2” offset angle (126 Runs):
O beam: -5.5E-10 ± 7.8E-10
H beam: -1.23E-9 ± 9.5E-10



Murphy's Law



Summary



- Implementation in MATLAB using handle classes
- Representation of measured topographies by bivariate 6th grade polynomials (incl. mixed terms)
- Local consideration of lamella thicknesses
- Path determination via ray-tracing
- Bookkeeping of phase factors
- Dynamic diffraction theory with plane waves inside lamellas
- Evaluation analogous to experiment data
- Monte Carlo (ca. 1.9 h per Run)