X-ray phase contrast imaging simulation program

USER GUIDE

# Overview:

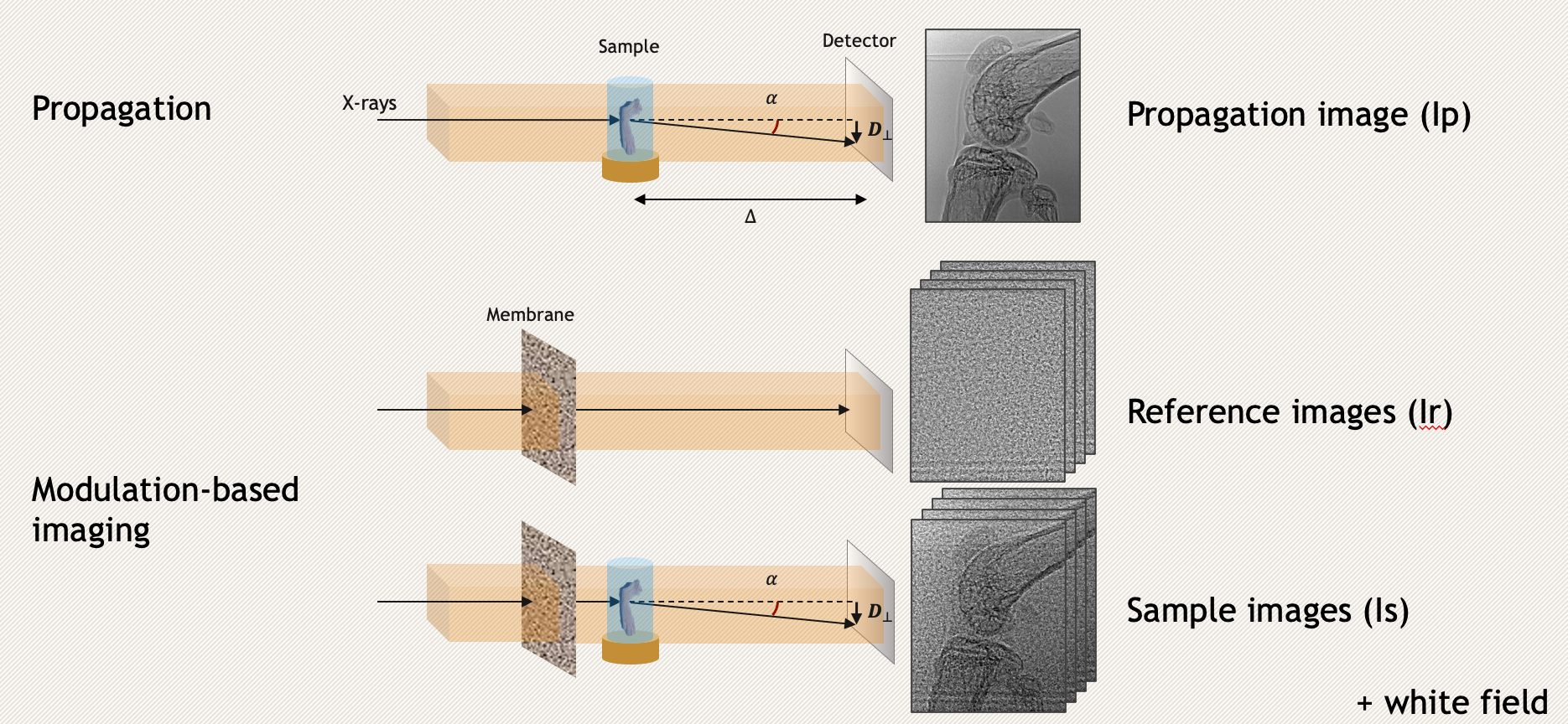
Objective: compute the intensity images obtained during modulation based x-ray phase contrast imaging.

From thickness maps of the geometries of the samples and membranes it computes sample images (with membrane and sample) and reference images (with only the membrane) along with simple propagation image (with only the sample) and a white (nothing other than air, filters and detector in the path of the beam.

There are 2 calculation methods: one based on ray-tracing and one based on wave optics and Fresnel propagator.

All the experiment parameters are set in the xml files (in folder “xmlFiles”).

Some algorithmic parameters are set directly in the code in the ‘main’ part of the main.py file.



If you need a new option that is not available in the code as it is or if you notice any bug, you can create a trouble ticket on GitHub! 😉

Table of content

[Overview: 1](#_Toc95385546)

[User instructions: 2](#_Toc95385547)

[1) Determine the sampling requirements for your experiment (“oversampling” parameter) 2](#_Toc95385548)

[2) Set the Parameters in the main (main.py) 2](#_Toc95385549)

[3) Set the Experiment global parameters in Experiment.xml 3](#_Toc95385550)

[4) Set the detector parameters in Detector.xml 3](#_Toc95385551)

[5) Set the source parameters in Source.xml 4](#_Toc95385552)

[6) Set all the objects parameters in Sample.xml 5](#_Toc95385553)

[*Additional remarks:* 8](#_Toc95385554)

[Required Python libraries: 9](#_Toc95385555)

# User instructions:

## Determine the sampling requirements for your experiment (“oversampling” parameter)

In this simulation, samples geometries are defined as thickness maps. For precise results, the description of the geometries requires to be over sampled compared to the detector pixel size.

If using ray-tracing you simply need a sampling >= 2 (Shannon).

If using Fresnel: Open the script: usefullScripts/getSamplingFactor.py. Enter the correct parameters for your experiment. Note down the minimal oversampling. (Some parameters are explained in the following instructions) (﻿Häggmark, I., Shaker, K., & Hertz, H. M. (2020). In Silico Phase-Contrast X-Ray Imaging of Anthropomorphic Voxel-Based Phantoms. IEEE Transactions on Medical Imaging, 40(2), 539-548.)

/!\ for low resolution experiments, the oversampling factor for the Fresnel method may be very important and induce a longer computation time.

## Set the Parameters in the main (main.py)

- experimentName : [string] corresponding to the one in the csv file

- filepath: [string] is the path of the folder where you want to store the simulated images

- overSampling: [int] is the oversampling factor for the sample and membrane geometries. See previous section on how to choose it.

- nbExpPoints: [int] is the number of pair Is, Ir with different position of the membrane you want to simulate.

- ﻿simulation\_type: [string] “Fresnel” or “RayT” too chose the type of calculation method (For polychromatic low coherence simulations ray-tracing is better).

## Set the Experiment global parameters in Experiment.xml

**<experiment>**

**<name>**Clinic\_ContrastPhantom**</name>**

**<distSourceToMembrane** unit**=**"m"**>**1**</distSourceToMembrane>** # [m]

**<distMembraneToObject** unit**=**"m"**>**0**</distMembraneToObject>** # [m]

**<distObjectToDetector** unit**=**"m"**>**1**</distObjectToDetector>** # [m]

**<membraneName>**Mask\_CuSn\_From\_edf**</membraneName>**

**<sampleName>**ContrastPhantomB**</sampleName>**

**<sampleType>**AnalyticalSample**</sampleType>**

**<detectorName>** flatPanelSimap **</detectorName>**

**<sourceName>**clinic**</sourceName>**

**<meanShotCount>**27600**</meanShotCount>**

**<plateName>**C\_plate**</plateName>** # non mandatory parameter, default = no plate

**<inVacuum>**True**</inVacuum>** # non mandatory parameter, default = False

**</experiment>**

# non mandatory parameter = # nmp

For now, **sampleType** must always be “AnalyticalSample” because “voxelized sample” is not implemented yet.

All element name must correspond to the elements names in their own xml file (sampleName, detectorName…).

**plateName** is an optional parameter. If you introduce a plateName, you must add a sample corresponding to its name in the Sample.xml file. In the simulation it corresponds to the detector protective plate (usually a thin carbon layer).

**meanShotCount** is the average number of photons per pixel leaving the source (=the number of photon that would hit 1 pixel of the detector if there was only vacuum in between).

**inVacuum** if ‘False’, attenuation in air between the source and the detector is accounted for. (default is False if not defined in the xml file)

## Set the detector parameters in Detector.xml

**<detector>**

**<name>**flatPanelSimap**</name>**

**<myDimensions>**

**<dimX>**500**</dimX>** # [pixels]

**<dimY>**1500**</dimY>**

**</myDimensions>**

**<myPixelSize** unit**=**"um"**>**127**</myPixelSize>** [µm]

**<myPSF** unit**=**"pixel"**>**1**</myPSF>** # std deviation of the PSF in pixels

**<myBinsThersholds>**20,30,40**</myBinsThersholds>** # nmp, default no bins separations [keV]

**<myScintillatorMaterial>**CsI**</myScintillatorMaterial>** # nmp, default no scintillator

**<myScintillatorThickness** unit**="**um**">**600**</myScintillatorThickness>** # required if myScintillatorMaterial defined [um]

**</detector>**



**dimX** and **dimY** will correspond to the dimensions of your final image in pixels.

**myBinsThersholds** are the thresholds of a spectral detector. They must be in the range of the source spectrum. They must be floats separated by only “,”. If not specified, the detector is not a spectral one. (It is an optional parameter). If given **<myBinsThersholds>**20,40**</myBinsThersholds>** the images will be separated in bin going from 0 to 20keV then 20 to 40 keV and then 40keV to the end of the spectrum.

The detector spectrum efficiency can now be calculated from its scintillator parameters. It is calculated as the attenuation in the thickness of the scintillator plaque using its material absorption coefficient. This coefficient is the beta coefficient that must be present in the TablesDeltaBeta.xls file of the Sample/DeltaBeta folder. (See the part about objects definition below for more details about this file). For the program to take the scintillator into account you simply add the **myScintillatorMaterial** and **myScintillatorThickness** parameters in the xml file. (They are optional parameters).

The PSF value corresponds to the standard deviation of the point spread function of the detector approximated by a gaussian filter. For a photon counting detector with no PSF set it to 0.

The pixel size assumes square pixels.

## Set the source parameters in Source.xml

Sources.xml: the source can be monochromatic or polychromatic **/!\ they do not have the same parameters**

**There are now 2 types of polychromatic sources. Simple x-ray tubes in reflexion which spectrum already exist in SpekPy or your own source which spectrum exit in an xls file.**

**<source>**

**<name>**simap**</name>**

**<myType>**Polychromatic**</myType>**

**<mySize** unit**=**"um"**>**8**</mySize>** #size of the emitting spot [µm]

**<sourceVoltage** unit**=**"kVp"**>**40**</sourceVoltage>** # [kVp]

**<myTargetMaterial>**Mo**</myTargetMaterial>** # nmp default is W

**<filterMaterial>**Be**</filterMaterial>** # nmp default None

**<filterThickness** unit**=**"mm"**>**0.2**</filterThickness>** # mandatory only if filterMaterial is defined

**<myEnergySampling** unit**=**"keV"**>**5**</myEnergySampling>**

**</source>**

The source size (**mySize**) corresponds to the FWHM of the spot on the anode. It will be projected on the detector plane and approximated by a gaussian shape.

The source voltage allows to get its spectrum from the SpekPy library.

The source target material can be Tungsten: W (value by default if not defined), Molybdenum: Mo or rhodium Rh.

The **filterMaterial** is optional too. It corresponds to a filter placed right after the source with thickness **filterThickness**. If those parameters are not specified there is no filter.

**myEnergySampling** is for under-sampling the spectrum for the calculation (if we calculate for every energy the computation time gets quite long). It must be >=0.5keV.

Now with your own spectrum:

**<source>**

**<name>**YourSource**</name>**

**<myType>**Polychromatic**</myType>**

**<spectrumFromXls>**True**</spectrumFromXls>**

**<pathXlsSpectrum>**Sources/W\_50kVp.xls**</pathXlsSpectrum>**

**<energyUnit>**eV**</energyUnit>** <!—eV, keV or MeV-->

**<energyColumnKey>**Energy**</energyColumnKey>**

**<fluenceColumnKey>**Flux**</fluenceColumnKey>**

**<myEnergySampling** unit**=**"keV"**>**4**</myEnergySampling>**

**<mySize** unit**=**"um"**>**50**</mySize>**

**</source>**

To use this kind of source, use this template with: **<spectrumFromXls>**True**</spectrumFromXls>**

In this case, you need to give the source spectrum file path (**pathXlsSpectrum)**. /!\ it must be .xls not xlsx because the python library cannot read those.

You give it the energy column key, fluence column key. The values must start in the row following those keys. You must also give it the energy unit. (The code works with keV so if you put eV or MeV in energyUnit it converts it. You can still resample it but only with energy bins larger than the one in your excel file (I haven’t implemented interpolation to down sample it).

Monochromatic case:

**<source>**

**<name>**id17**</name>**

**<myType>**Monochromatic**</myType>**

**<mySize** unit**=**"um"**>**10**</mySize>**

**<myEnergy** unit**=**"kev"**>**27**</myEnergy>**

**</source>**

## Set all the objects parameters in Sample.xml

This is the trickiest part.

There are 2 main types of sample: the sample of interest and the membrane. The air volume is there by default and must not be modified a thin protective plate can also be added in fron of the detector as a Sample of type “thin\_film”.

**<sample>**

**<name>**filNylon**</name>**

**<myType>**sample\_of\_interest**</myType>**

**<myMaterials>**Nylon**</myMaterials>**

**<myGeometryFunction>**CreateSampleNylonWire**</myGeometryFunction>**

**<myRadius** unit**=**"um"**>**70**</myRadius>**

**<myOrientation** unit**=**"degree"**>**81.56**</myOrientation>**

**</sample>**

**<sample>**

**<name>**Mask\_CuSn\_From\_txt**</name>**

**<myType>**membrane**</myType>**

**<myGeometryFunction>**getMembraneSegmentedFromFile**</myGeometryFunction>**

**<myPMMAThickness** unit**=**"um"**>**4000**</myPMMAThickness>**

**<myMaterials>**CuSn,PMMA**</myMaterials>**

**<myMeanSphereRadius>**6**</myMeanSphereRadius>**

**<myNbOfLayers>**2**</myNbOfLayers>**

**</sample>**

**<sample>**

**<name>**C\_plate**</name>**

**<myType>**thin\_film**</myType>** <!--element that will be placed in front of the detector-->

**<myGeometryFunction>**get\_my\_thickness**</myGeometryFunction>**

**<myMaterials>**CarbonFiber**</myMaterials>**

**<myThickness** unit**=**"um"**>**2500**</myThickness>**

**</sample>**

**<sample>**

**<name>**air\_volume**</name>**

**<myType>**volume**</myType>**

**<myGeometryFunction>**get\_my\_thickness**</myGeometryFunction>**

**<myMaterials>**Air**</myMaterials>**

**</sample>**

Every sample has:

- **name**

- **myType** which is either “sample\_of\_interest”, “membrane”, “thin\_film” (this one corresponding to the detector protective plate) or “volume” (for the volume of air between the source and the detector).

- **myMaterials** is either 1 material name or a list of materials separated by comas “,” (NO SPACES). This material will be used in the getDeltaBeta function. Important precisions about those materials bellow.

- **myGeometryFunction** is the function that you use to get the thickness map of your sample (or thickness mapS if there are several materials) more details bellow.

Any other parameters are related to the way you generate the samples geometries.

##### Delta Beta parameters:

/!\ For each material of the considered samples, a delta-beta table must exist in the TablesDeltaBeta.xls file (in Samples/deltabeta/) in the same way as the other already entered data.

A list of the already existing materials is in the first column of the sheet.

To generate a new table of data use the website: <http://ts-imaging.science.unimelb.edu.au/Services/Simple/ICUtilAbsorb.aspx>

Then copy and paste the obtained values in the ArangeValue sheet of the TablesDeltaBeta.xls doc. To separate the columns, use Data>Convert (the energies must be in the B column). Then, remove all the additional spaces with the replace tool (replace “ “ with “”). Change the value in K1 to give it your material name (careful not to add any spaces before or after the word). Then copy columns K to N and use “paste Values” to add them after every other on the “Data” sheet. Add the name of your new material in the first column.

**[EDIT 14/06/2022]**

The xraylib python library can now be used. This library requires either the chemical formulas or elements percentages of your material to be written in the Samples/Materials.csv file. Per default, the code will now search first if your material is in this file before searching in the xls file.

Some tests were done comparing the index values obtained from TSImaging with the ones given by xraylib. They are not quite identical but the difference is usually smaller than 10 percent.

##### Membrane geometry:

The membrane geometry is based on a CuSn membrane microscopic image segmented. The average radius of those spheres was 4.57 µm but the distribution of those radia is not uniform. (See Appendix 1). The average with respect to the surface they occupy is 12 µm.

As a parameter, you can choose to change the mean sphere radius (it is advised to start by simulating 1 point and verify that the grains are visible on the reference image). We observe best results when they appear to be between 5 and 10 pixels wide.

You can choose the grains material (first material of your list) and the material of the support (usually PMMA).

You can choose the number of layers of spheres you want to generate: **myNbOfLayers**.

You can choose the thickness (**myPMMAThickness**) and material of the support on which the spheres are placed. We use PMMA plates hence its name but its material given in second place in “myMaterials” can be changed.

/!\ the grains geometry is independent from the PMMA thickness on which they are fixated. The PMMA thickness is the total thickness of all considered layers: if we use 3 layers of 2mm thick plates, the PMMA thickness should be 6000um.

To create the membrane there are 2 options:

1. Generate it directly in the simulation from the segmented coordinates of the spheres using the getMembraneSegmentedFromFile geometry function. This is quite time consuming. If you are going to use several time the same geometric configuarion, you can use the second option.
2. Start by saving thickness maps of the spheres using the Samples/﻿getMembraneFromFile.py script (launching the first option directly will also save the thickness maps of your membrane that you can re-use here). You must enter all the parameters in the main corresponding to your experiment. Launch it and copy the path.

Then in the simulation you will use the getMembraneFromFile geometry function. Paste the path of the folder where they are saved in the xml file corresponding to your membrane for the simulation under **myMembraneFile**.

In that case, you must remember the number of layers to adapt the thickness of PMMA to add.

Remark: the segmented surface is limited so depending on the size of the simulated surface you may see the stitching edge appear on the geometry. It does not disturb the phase retrieval.

##### Samples geometries:

Some geometry functions are already implemented: a cylinder, a sphere, two spheres in a cylinder, 2 spheres in a parallelepiped and a contrast phantom used for the test of micro-tomography tools. You can choose them by setting their geometry functions. You can also load thickness maps if those have been previously created.

In order to select a geometry, you must define it in the item **<myGeometryFunction>** in the Sample.xml file.

**<myGeometryFunction>**CreateSampleCylindre**</myGeometryFunction>**

For that function you must precise <myRadius> in um, <myOrientation> in degree and <myMaterial>. Be sure that the diameter of your wire fits your final image size (defined by the size of your detector) when multiplied by the magnification (defined by your distances).

**<myGeometryFunction>**CreateSampleSphere**</myGeometryFunction>**

Set <myRadius> in um and <myMaterials>. Check also the size of your sample.

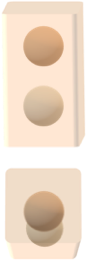
**<myGeometryFunction>**CreateSampleSpheresInCylinder**</myGeometryFunction>**

This sample has a fixed size. Spheres radius is 0.5mm, cylinder radius is twice the one of the sphere. The tube height is the final image height.

You only need to set <myMaterials> with 3 materials for this sample: Sphere1Material,Sphere2Material,TubeMaterial. Once again, please check that the sample fits your geometry.

If you wanted to change the sphere parameters, you need to go directly in the code in Samples/createSampGeom.py at the beginning of the function. (ctrl+f CreateSampleSpheresInCylinder to find it faster).

**<myGeometryFunction>**CreateSampleSpheresInParallelepiped**</myGeometryFunction>**

Is identical to CreateSampleSpheresInCylinder except that the spheres are in a smoothed edges parallelepiped instead of a cylinder.

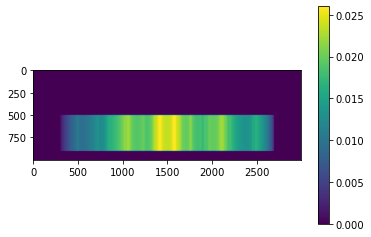
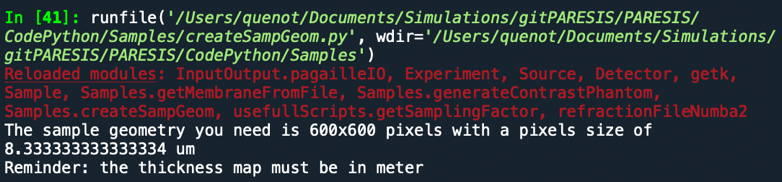
**<myGeometryFunction>**generateContrastPhantom**</myGeometryFunction>**

This phantom geometry is fixed and really cannot be changed. It is a phantom of materials equivalent to human tissues which materials are already set in the xml files but can be changed at your convenience. The final geometry will be calculated for the phantom seen from the side which will be more interesting for tomography.

*Figure 1: Contrast phantom geometry.*

The function is stored in the generateContrastPhantom.py file. This script can be launched directly with its main to create your geometry beforehand and save the thickness maps as .edf files. Then you will have to use the **<myGeometryFunction>** ﻿loadSampleGeometryFromImages **</myGeometryFunction>** to load the geometry (faster than generating the phantom fitting your geometry everytime you launch the code).

**<myGeometryFunction>**﻿loadSampleGeometryFromImages**</myGeometryFunction>**

This function requires a folder path **<myGeometryFolder>.** This folder must be containing all the thickness images corresponding to the different materials of your sample (as .tif or .edf). The thickness values must be in m and the image dimensions must match your experiment geometry. In order to know the dimensions needed for your thickness map, you can use the main in the creatSampGeom.py. In the main part (at the end of the file) enter the required experiment information and run this file. It will give you the pixel size you need and the dimensions in pixels.

*Figure 2: Example of a thickness map image.*

When loading several materials geometry, the thickness maps will be ordered in the alphabetic order of their file names. This order must be the same as the one you give in the **<myMaterials>** item in the xml file.

**Create a new geometry function**

A template is already implemented in the file createSampGeom.py.

﻿CreateYourSampleGeometry(myName, dimX0, dimY0, pixelSize)

You can modify this function at your convenience to create any sample geometry you want. It takes as parameters, the dimensions of the thickness maps you need to create, and the pixel size. (the name is not really useful). You must create a 3D numpy array with as many thickness maps as the number of materials comprised in your sample. The order of the material you give in **<myMaterials>** in the xml file must be the same as the order of the thickness maps in your geometry array.

## Additional remarks:

1. For now, the code does not calculates the number of photons per pixel as a function of voltage, current intensity and exposition time, you have to chose the value by yourself. If you know a simple and efficient way to calculate that, please let me know.
2. Both calculation methods give similar results for synchrotron simulations but they diverge for less coherent systems. It is then preferable to use the ray-tracing one.
3. The refraction calculation is greatly accelerated by numba.jit

# Required Python libraries:

* OpenCV, <https://pypi.org/project/opencv-python/>
* Numba, <http://numba.pydata.org/>
* Xml.dom, <https://docs.python.org/3/library/xml.dom.minidom.html>
* Xlrd, <https://pypi.org/project/xlrd/>
* spekpy

<https://bitbucket.org/spekpy/spekpy_release/wiki/Further%20information#markdown-header-specify-a-target-material-other-than-tungsten>

R Bujila, A Omar and G Poludniowski, *A validation of SpekPy: a software toolkit for modelling x-ray tube spectra*. Phys Med. 2020; 75:44-54.

A Omar, P Andreo and G Poludniowski, *A model for the energy and angular distribution of x rays emitted from an x-ray tube. Part I. Bremsstrahlung production*. Med Phys. 2020; 47(10):4763-4774

A Omar, P Andreo and G Poludniowski, *A model for the energy and angular distribution of x rays emitted from an x-ray tube. Part II. Validation of x-ray spectra from 20 to 300 kV*. Med Phys. 2020; 47(9):4005-4019

A Omar, P Andreo and G Poludniowski, *A model for the emission of K and L x rays from an x-ray tube*. NIM B 2018; 437:36-47.

G Poludniowski, *Calculation of x-ray spectra emerging from an x-ray tube. Part II. X-ray production and filtration in x-ray targets*. Med Phys 2007; 34(6):2175-86.

G Poludniowski and PM Evans, *Calculation of x-ray spectra emerging from an x-ray tube. Part I. electron penetration characteristics in x-ray targets*. Med Phys 2007; 34(6):2164-74.

G Poludniowski, et al., *SpekCalc: a program to calculate photon spectra from tungsten anode x-ray tubes*. Phys Med Biol 2009; 54(19):N433-8.

* Scipy, <https://pypi.org/project/scipy/>

**Simulation physical models:**

from the validation article: Quenot, L., Brun, E., Létang, J. M., & Langer, M. (2021). Evaluation of simulators for x-ray speckle-based phase contrast imaging. Physics in Medicine & Biology, 66(17), 175027.

RAY TRACING

We start by considering the projection approximation which is most of the time valid when the object is small in comparison to the propagation distance and states that the description of the wave through an object can be described as phase shift and attenuation following straight lines through the object. It means that we can calculate the phase shift and the absorption B through the entire object at once with the following relations:

where x and y are the spatial coordinates in the plane perpendicular to the propagation direction z. and are the indexes for refraction and absorption introduced, is the wavelength and is the wave number.

The intensity after the object can then be calculated as:

From the phase shift, the refraction angle of each ray can be computed as:

And using that angle map, the redistribution of intensities on the detector plane after propagation over a distance z2 can be calculated as:

The intensity received in the detector plane is then convoluted with the source projected size and sampled at the detector pixel size. Then a convolution with the source blurring and point spread function of the detector was applied. Finally, shot noise is modeled using a Poisson distribution for a more realistic result.

FRESNEL PROPAGATOR

The second simulation tool developed is based on the wave optics model for x-rays, describing propagation as a linear operator usually called the Fresnel propagator.

Wave models can be derived starting from Maxwell's equation. From that physical model, wave descriptions for coherent x-rays have been developed and are detailed in \parencite{Paganin2006}.

The starting point of the model is to write the wave equation of the beam:

where is the wavelength, z is the coordinate along the propagation axis, x and y are the coordinate on the perpendicular plane, kx, ky and kz are the vectorial components of the wave vector. Before the wave reaches any object, it can be considered that the wave vector is along the propagation direction therefore, the previous equation becomes:

Then, when a wave goes through an object, it will be partly absorbed and phase shifted. A simple way to calculate those changes is if we use the projection approximation as in the ray tracing model.

Under that approximation, the wave after the object will be modified by a transmittance function that can be described as follows:

However this approximation has limits as it was shown in \parencite{morgan2010projection} and for larger objects or coherent sources, it may be biased.

Once the wave carries the object information, it will undergo other modification during propagation to the detector plane. The experiments of interest here being mainly limited to the Fresnel domain under the paraxial approximation, that propagation can be described by the ``Fresnel Propagator'' \parencite{Paganin2006}.

There are two formulations of that propagator, one in the real space and the second one in the Fourier space. The first one including a convolution product, it has a big computation cost while the second one is a simple multiplication in the Fourier space as shown in eq. \ref{eq:FresnelPropagator}

where is the Fresnel propagator in Fourier space and z is the distance of propagation.

After propagation to the detector plane, the wave function is converted to intensity information with the following relationship:

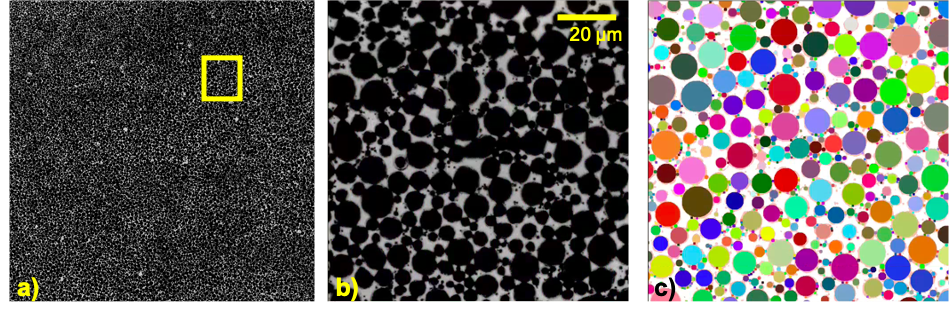
Then, re-sampling and convolution with the source blurring and the detector's PSF were applied as well as the introduction of the shot noise the same way as for the ray-tracing code.

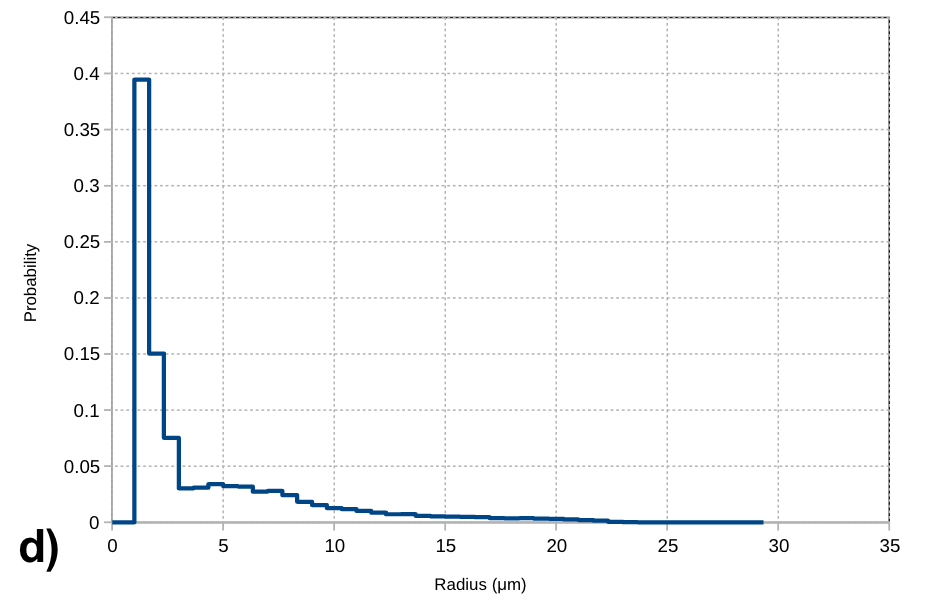
In the case of indirect detectors, the scintillator spectrum efficiency can be taken into account. It is then calculated as the proportion of absorbed photons in the plaque for each energy considered with the law: with k the wavenumber, the absorption coefficient of the scintillator material at the considered energy and t the thickness of the scintillator plaque.

Appendix 1:

Sphere radius distribution:

Quenot, L., Brun, E., Létang, J. M., & Langer, M. (2021). Evaluation of simulators for x-ray speckle-based phase contrast imaging. *Physics in Medicine & Biology*, *66*(17), 175027.





Appendix 2:

Class Diagram of the code architecture NOT UP TO DATE

