Exploring the simulation of GISAXS patterns from real space 3D models with Blender and HipGISAXS-2.0.

Paul Vautravers

University of Manchester

October 31, 2022

Abstract

3D real space models were made for silver nanowire (AgNW) and cellulose nanofibril (CNF) meshes using Blender. The distorted wave born approximation (DWBA) was applied to the AgNW mesh models through the grazing incidence small angle x-ray scattering (GISAXS) simulation software, HipGISAXS-2.0. The simulation process for producing GISAXS patterns was automated, while the initial stage in blender was semi-automated. Scattering patterns produced for meshes of 2360 AgNWs used cylindrical form factors and were averaged together, displaying patterns that encouraged further use of this simulation pipeline.

Contents

1. Introduction

1.1. Nanoparticles:

The study of nanoparticles is developing at a prodigious rate, owing to the fascinating and immensely useful properties that systems composed of nanoparticles exhibit. These properties differ so much from those of bulk materials due to confinement effects. Confinement effects and their appearance as interesting, novel properties, depend on the size and shape of individual nanoparticles, as well as the arrangement of systems of nanoparticles. In order to develop systems of nanoparticles with desirable properties, characterisation tools are required to understand their morphology.

1.2. GISAXS:

Many methods exist which characterise nanoscale structures. Conventionally, real space tools such as Atomic Force Microscopy (AFM) and Transmission Electron Microscopy (TEM) have been extremely popular for probing local surface structures. However, while convenient and well established, these tools have notable disadvantages for probing more sophisticated materials and material processes: they produce limited sample statistics, tip convolution artefacts, they are difficult to apply in situ during kinetic processes, such as sample growth and they completely fail to gather information for embedded, sub-surface structures [1].

Grazing Incidence Small Angle X-ray Scattering (GISAXS) is a characterisation technique that manages to remedy these issues, albeit with its own shortcomings. GISAXS can be compared with regular Small Angle X-ray Scattering (SAXS), but featuring a reflection geometry as opposed to a transmission geometry. GISAXS' scattering geometry differs in several ways from that of SAXS, on account of the substrate-film interface. Due to the shallow angles involved, close to the critical angle of the material, multiple scattering effects must be accounted for. The Born Approximation (BA) for a single scattering event is no longer valid for GISAXS. The Distorted-Wave Born Approximation (DWBA) accounts for the additional scattering events which occur in this regime, considering (in decreasing likeliness): scattering (BA), reflection then scattering, scattering then reflection and lastly, reflection, scattering and then a further reflection. The terms that constitute the DWBA are shown in figure 1. Having accounted for multiple scattering events, the form factor is now modified such that:

$$F(q_{xy}) = F(q_{xy}, q_z) + R(\alpha_i)F(q_{xy}, p_z) + R(\alpha_f)F(q_{xy}, -p_z) + R(\alpha_i)R(\alpha_f)F(q_{xy}, -q_z), \quad (1)$$

where q_{xy} is $({q_x}^2 + {q_y}^2)^{1/2}$ and p_z is $(\boldsymbol{k_i} + \boldsymbol{k_f})_z$. For incident and final angles much greater than the material's critical angle, one can neglect the higher order terms and the first order BA is retrieved. Notable features of resulting GISAXS patterns are the specular peak, where $\alpha_i = \alpha_f$, and the diffuse scattering intensity peak, $\alpha_i \neq \alpha_f$, also known

as the Yoneda peak. Due to reflection and refraction, the Yoneda peak occurs at the material dependent critical angle for total external reflection [1]. In conjunction with other features, the specular and Yoneda peaks may then be used to characterise films of nanoparticles. Furthermore, due to the beam travelling a significant distance through the film, a large sample area is probed and the resulting patterns provide statistically significant information on nm scale morphologies averaged over mm to cm scales. Depth dependent information can be retrieved by varying incident angle, with information on embedded structures also being accessible. Lastly, assuming the film can bear exposure to x-rays, the technique is non-destructive and can be applied in a range of different dynamic processes, enabling *in situ* studies [2]. Disadvantages of GISAXS compared to other techniques are: its reliance on synchotron radiation; collected signals are very small without a high brilliance source, morphological information is presented in reciprocal space; must be converted back into real space, and difficulty characterising single nanoparticles; beam size and limitations of x-ray optics convene to make it difficult to probe very local parts of a sample.

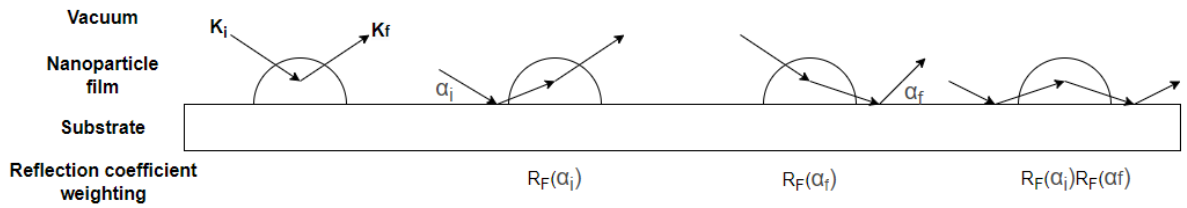


Figure 1: Illustration of the scattering and reflection events which the distorted wave born approximation (DWBA) accounts for. R_f denotes Fresnel reflection coefficient; α_i and α_f the incident and final angles; K_i and K_f the incident and final wave vectors. Schematic adapted from [2] and [3]

.

1.3. HipGISAXS:

To complement real experimental GISAXS results, multiple in-house GISAXS simulation and analysis tools have been developed. While many open-access, dedicated GISAXS tools now exist, many are orientated towards specific types of material structures: they limit the user to specific models and are not fit for all purpose GISAXS simulations. For example, the tool IsGISAXS [4] focuses on nano particles supported on a substrate and struggles with the number of layers present in the sample. In every case, analysis has been limited to systems of objects with analytical form factors. HipGISAXS [5], however, goes beyond other existing tools and allows for arbitrary extended objects as well as standard analytical objects to be treated. The ability to quickly simulate GISAXS patterns for discretised graphical objects, e.g. triangulated meshes contained in stl files, is enabled through HipGISAXS' judicious use of parallel computing.

1.4. Overview of report:

The focus of this project was firstly to explore using the animation software Blender to produce authentic digital samples of silver nanowire (AgNW) and cellulose nanofibril (CNF) films and secondly to simulate GISAXS patterns from these models with HipGISAXS-2.0., using either analytical form factors where possible or triangulated meshes.

Without further detailed discussion of the scattering theory relevant for GISAXS or the parallel computing theory required for HipGISAXS, this report is structured in the following manner. The methods section details: first; the process of producing AgNW and CNF samples in Blender, second; retrieval of relevant data from Blender models, third; the use of the HipGISAXS-2.0 code and fourth; the development of automation routines. The results section gives examples of blender samples generated for both materials as well as present the limited GISAXS patterns produced at the time of writing. Thereafter, suggested improvements to the implementation of blender and HipGISAXS-2.0 are discussed.

2. Methods

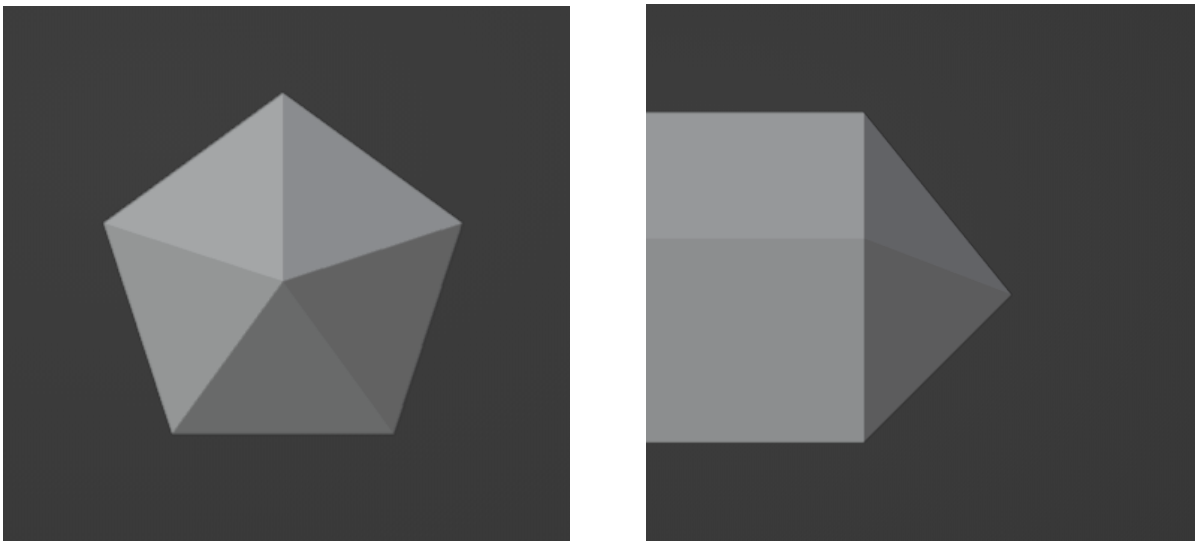
2.1. Development of 3D models with Blender

2.1.1. Silver Nanowires

The blender user interface was used to add a cylinder object. Immediately upon adding the cylinder, the number of vertices that define the circular cross section was lowered to 5. This produced the desired pentagonal prism shape. Additionally, the edges around the pentagonal face were extruded outwards and snapped together. This created a pentagonal pyramid end to the model wire ends, as seen in figure 4. Additionally, a plane object was originally chosen to act as a substrate.

After constructing a model for a single wire, it was necessary to specify the physics of the object. In the case of AgNW, rigid-body physics was suitable. Rigid body parameters were tweaked many times in order to achieve stable and convincing behaviour for many AgNWs. The final parameters may be found in the appendix. Most importantly, the object's physics is rigid body, with a convex hull and set to 'active'. On the other hand, the substrate must also be given rigid body dynamics, but set to 'passive'. This ensures that the wires fall down, and that the plane interacts with them, but does not fall itself.

To develop a model of an AgNW film, a large number of AgNWs was required. This was achieved by applying the array modifier to the single AgNW base model. Applying the modifier once allowed the construction of a row of wires, extending perpendicular to the wire length. Applying this a second time then made a plane of wires. Applied once more, the array modifier then gave a 3D array wires. To separate the resulting array object back down into its constituent wires, 'separate by loose parts' in edit mode



(a) Cross section of AgNW (b) Pentagonal Pyramid end

Figure 2: Illustration of the AgNW unit made using Blender

was selected, and then back in object mode 'origin to geometry'. This produces an array of wires all with the correct physics, based on the original model. The array structure used for the final films featured rows of wires, with gaps, that were orientated differently based on the vertical position in the array, seen in figure 3. This complex array structure occupied almost the whole substrate and improved the homogeneity of the film, compared to simple array structures.

In order to achieve a random arrangement of wires on the surface, it was necessary to randomise the positions and orientations of the wires around their positions in the initial array. This was first achieved by applying the randomise transformation, from the object menu, with all wires selected, and then varying the parameters accordingly. Later, however, this was performed via scripting.

2.1.2. Cellulose Nanofibrils

In addition to developing models for AgNW networks, blender was also used to create a small scale sample of a CNF film. The workflow for the AgNWs was also used for the CNF film. While having a circular cross section, the number of vertices was also reduced for the nano fibrils. In this case, the choice was motivated by practical limitations rather than design. Given that the fibrils required flexibility to collapse over one another, additional vertices around their circumference, split across the length of the object, were required. These additional rings of vertices then provided points where the object could bend. As such, the number of edges defining the cross section were limited to 20. This was to limit additional computation when applying non rigid body dynamics to the fibrils. The cross section and a side view of the fibril with its rings of vertices can be seen in figure 3. These sets of vertices along the fibril length were added from edit

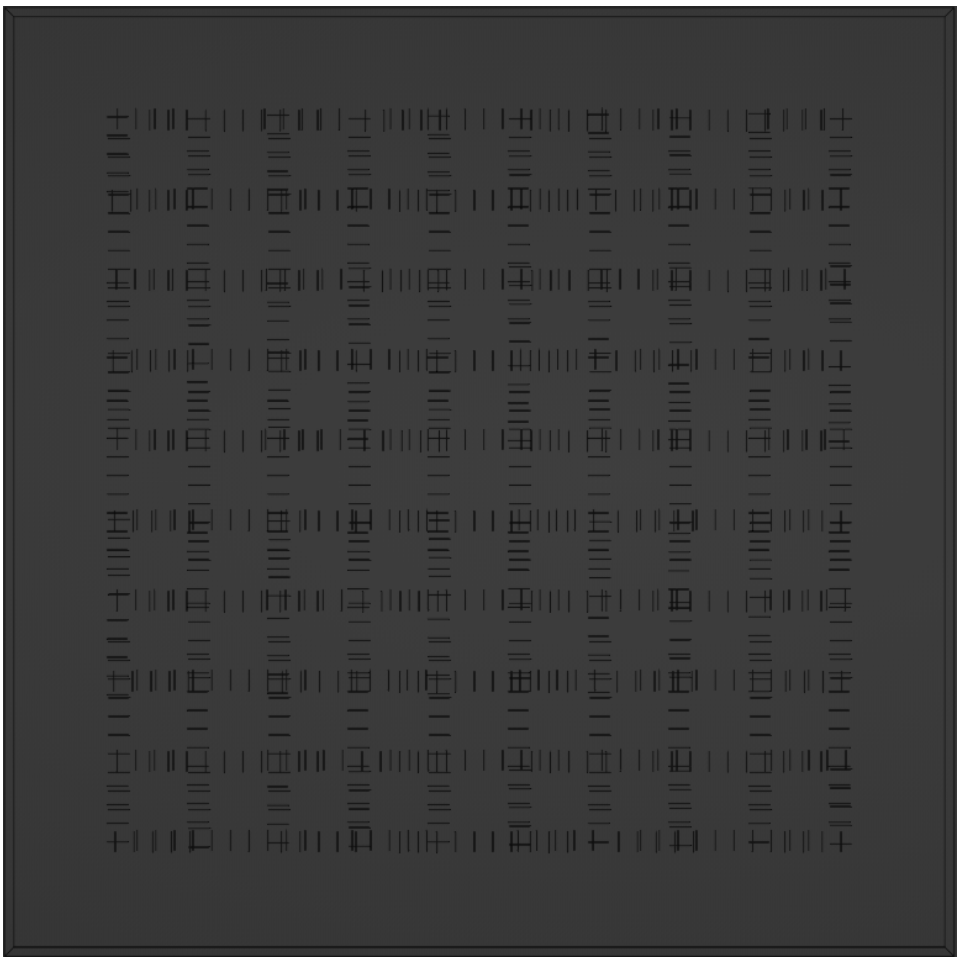
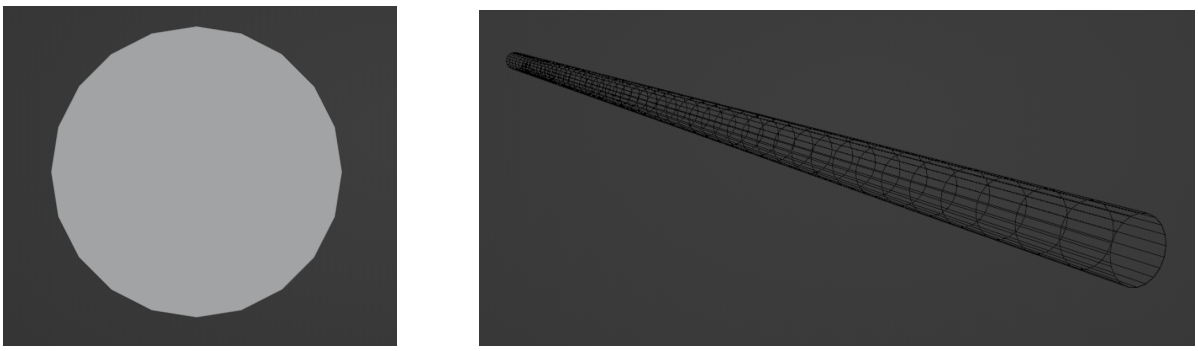


Figure 3: Vertical view of array of 2360 silver nanowire models, arranged to improve homogeneity of the deposited film. Alternating layers had wires in rows along x and y.

mode, using the 'loop cut' tool.

The soft body option provided by blender did not produce the desired effects for the nanofibrils; instead, the 'cloth' option was used, due to its internal pressure parameter. This ensured fibrils would not collapse under one another when interacting. The cloth simulation is, however, far more sophisticated and computationally demanding. This motivated reducing the number of vertices defining the cross section, as every vertex can move in a cloth model. In addition to having cloth physics, the fibrils also had 'collision' physics. The full list of physics parameters may be found in the appendix at the end of the report. An array structure was also used for the fibrils; however, their being so computationally demanding limited the samples to 25 fibrils. As such, arrays of 5x5 (y,z) were used. In this case, animations lasted up to 160 frames, and the fps dropped to around 0.5. Samples of 200 fibrils resulted in the fps dropping to 0.04, illustrating the challenge of using such sophisticated physics.



(a) CNF 20 sided 'circular' cross section.

(b) Additional vertices added to CNF unit to provide flexibility.

Figure 4: Illustration of the CNF unit made using Blender.

2.2. Applying scripting and outputting data from blender

Next it was necessary to develop a script with blender's python API to expedite the process of developing the film and to export the relevant data. This section will only apply for AgNW, as the HipGISAXS-2.0 code for arbitrary object inputs was not produced at the time of the project, naturally required to deal with the far from analytical shapes of the cellulose nanofibrils.

To avoid applying the randomise transform from the UI, python's uniform distribution was used to add a small translation and rotation to each sample, in all dimensions. The radius and lengths of the wires were distributed using python's normal distribution. Code was then written to store the positions and Euler angles of the wires, which would then be used with a cylindrical form factor in HipGISAXS-2.0 to simulate the GISAXS pattern for the 3D sample. This required firstly setting the animation in motion from the script and a function to get the angles and positions only once the film was settled. This was implemented using a frame handler to execute code at a specific frame. The relevant code is provided in the appendix.

2.3. Using HipGISAXS-2.0

With the positions, orientations and scalings outputted from blender, HipGISAXS-2.0 was used to generate GISAXS patterns. The approach used here was to treat the wires as cylinders. In this manner, the analytical cylindrical form factor was applied and effects due to the orientation and translation of individual cylinders dealt with trivially. Throughout this project, the Maxwell computing cluster at DESY was used to execute HipGISAXS-2.0. First, the experimental parameters were specified in the file, 'gen _config.py'. Having inputted the required parameters, the command , "python3 gen_config.py" was entered into the terminal. This updated the config.json file, which was used later by 'main.py'. Secondly, the path to the numpy file was specified in 'main.py' and the command "python3 main.py" was entered. These are the only steps

to generate GISAXS images once the numpy file is available.

2.4. Automation

The above procedures could not be applied to collect a large amount of data, without being extremely tedious. Clearly, an important part of this project was automating the data collection process and applying HipGISAXS-2.0.

The first development on this front was updating the code in 'main.py' to use the python 'sys' library, so that a default file name could be kept in the code, and a sample number could be inputted from the terminal. Following this, a .sh file was written that simply iterated over a range of values and generated the scattering pattern for each sample number. This enabled GISAXS patterns to be produced at a large scale, without further input from the user.

The process of creating samples was also automated. First this was done on the local system using a .bat file, but it was subsequently updated to a .sh file, to run on maxwell. This requires blender to be loaded on Maxwell, with command ,"module load maxwell blender". Blender then runs on Maxwell, opening the base file and applying the python script described above to it. The data is then collected in a numpy file, first as a dummy file, and then renamed according to the sample, to avoid overwriting data. This has been integrated with HipGISAXS-2.0 pattern generation to do everything in a single loop, or they can be done separately. As such, large volumes of sample data and simulations are generated remotely, in a reasonable timescale, without user input.

2.5. Miscellaneous

With sample sizes of only around 2000 nano wires, GISAXS patterns fluctuated between different samples. In order to produce GISAXS patterns that would be valid for a large system of nanowires, it was necessary to average GISAXS patterns over many different samples. Furthermore, it was necessary to investigate the number of samples required to reach an ensemble average that no longer displayed fluctuations. The approach taken was to construct a dictionary object in python, where an initial list of images was split into 2,4,8,16 sub lists etc. In this manner, an average was made from all images, then two averages from the two halves of the original list and then 4 averages from the 4 quarters etc. The dictionary keys correspond to the number of samples, giving an array containing one or more images; individual images can then be called with their index in the array. For example, with an initial input of 64 images, the average over all 64 is accessed with, 'agnw_avg _dict['64'][0]', while the 2nd average over 32 is given by, 'agnw_avg _dict['32'][1]'.

A simple image analysis approach enabled a crude estimate of the area coverage of these films. This was performed by taking a top-down image of the AgNW film, converting the pixel array into a black and white binary, and counting the number of black pixels,

corresponding to pixels where wires were present. The process of taking images, however, was not automated and so remains only a small development in this project, as discussed later. Furthermore, some limited work to characterise the size of pores produced in the film was completed using the image analysis library cv2 and the function 'connectedComponentWithStats'.

3. Results

3.1. Constructing 3D models

3.1.1. AgNW films

Blender's user interface was employed successfully in conjunction with an automation routine to produce a large amount of AgNW sample data. The pentagonal morphology of individual wires was reproduced while the qualitative features of AgNW networks were also replicated through blender. A substrate was produced of dimensions 80x80 μm, where a metre in blender corresponded to 1 μm. Gaussian distributed wires had lengths with mean and standard deviation 10 $\pm$ 0.1 μm and radii with mean and standard devaition of 0.06 $\pm$ 0.001 μm. Figure 5 presents a sample AgNW network produced in Blender. This appears to display good homogeneity, a large amount of connectivity, but also regions where pore-like gaps have formed in the film.

Additionally, side profiles of the networks developed in Blender also appear qualitatively correct, being uniform to a good degree, thinning out only slightly from the centre. Parameters in Blender's user interface such as the substrate-wire friction were employed successfully to promote film spreading and limit the number of standing wires. This is displayed in figure 6.

3.1.2. CNF films

Systems of Cellulose Nanofibrils were also produced with Blender, although on a far reduced scale. On account of the extremely large render times required for the nanofibrils, models were limited only to 25 fibrils. This was a failure of the approach used with blender, due largely to the number of vertices given to the fibrils. Despite that, these tests did show that blender is highly capable of producing chaotic systems of non rigid body objects in a convincing manner. Figure 7 provides two example images from samples of CNF made in Blender.

Side profiles of the selected samples clearly illustrate the arrangement of nanofibrils laying over one another, bending and drooping, shown in figure 8.

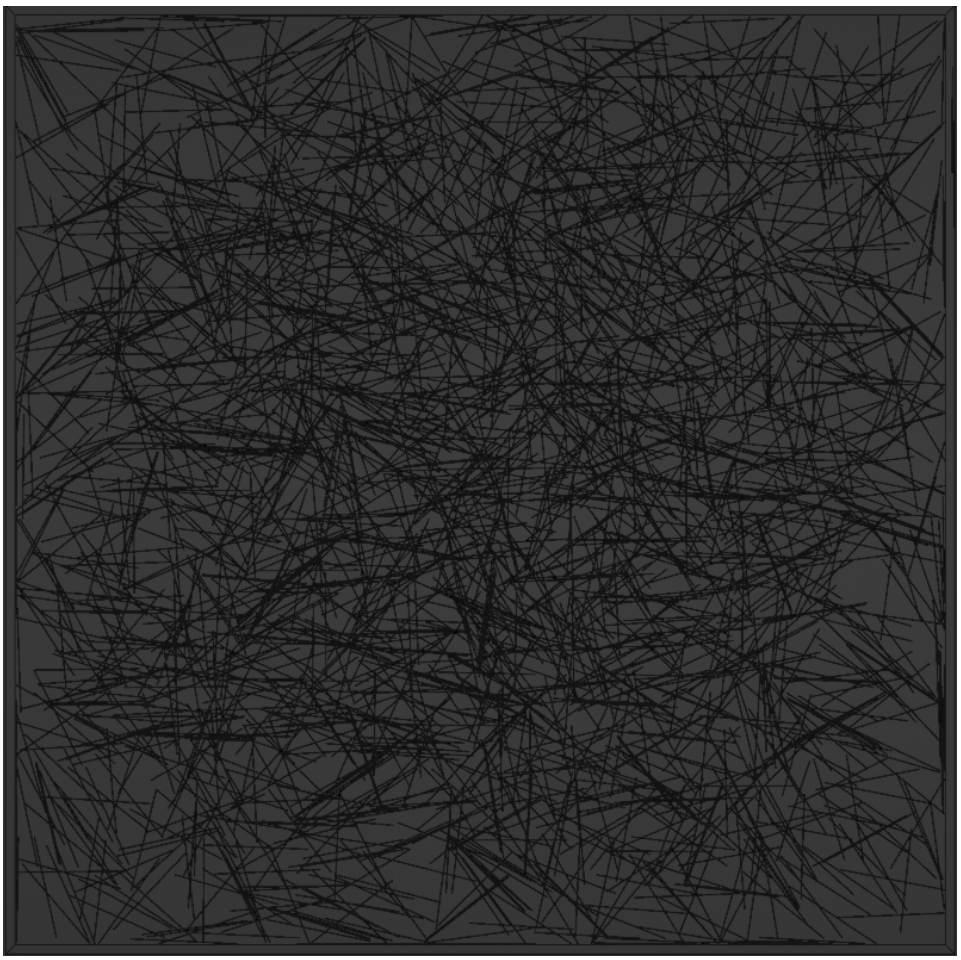


Figure 5: Image captured in blender UI, showing z axis view of 2360 silver nanowires deposited on a substrate.

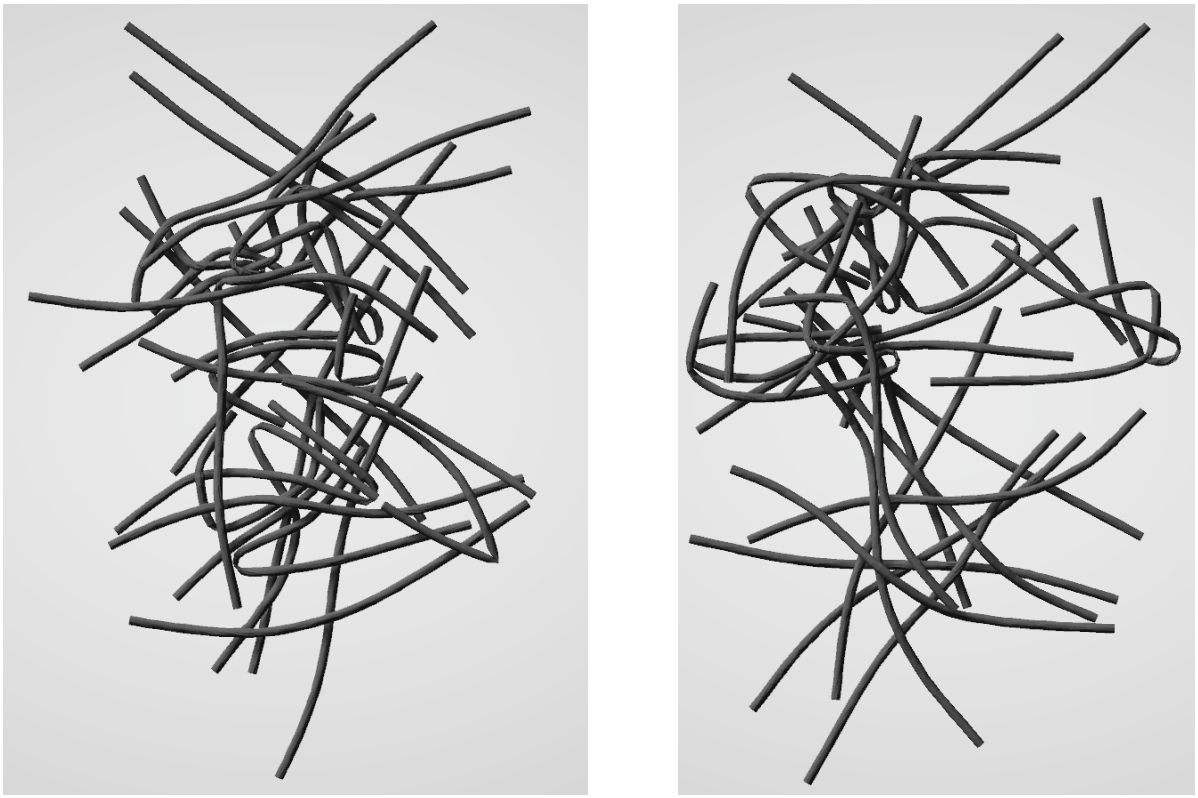
3.2. Simulation of GISAXS patterns

3.2.1. AgNW with Cylindrical Form Factor

Treating the wires as cylinders and applying the DWBA, GISAXS images were produced using HipGISAXS-2.0. As with sample generation, the simulation process was also successfully automated, producing 160 sample images in 4 hours and 8 minutes with CPUs. The patterns produced in this project did not completely satisfy what was expected, but they did encourage further work with the approach, using blender and HipGISAXS-2.0. Figure 9 illustrates two sample scattering patterns. These images do present some of the correct structure, with a diffuse scattering peak (Yoneda) extended outwards in Q_{xy}, at a non-zero Q_z value, as well as a specular peak raised above the Yoneda region. Additionally, there is high intensity at low Q_z and central Q_{xy}, corresponding to the direct beam intensity.

To acquire statistically reliable information on the morphology of a typical AgNW film,

Figure 6: Image captured in microsoft 3D object viewer, showing x axis view of 2360 silver nanowires deposited on a substrate. Film only bulges marginally at the centre.

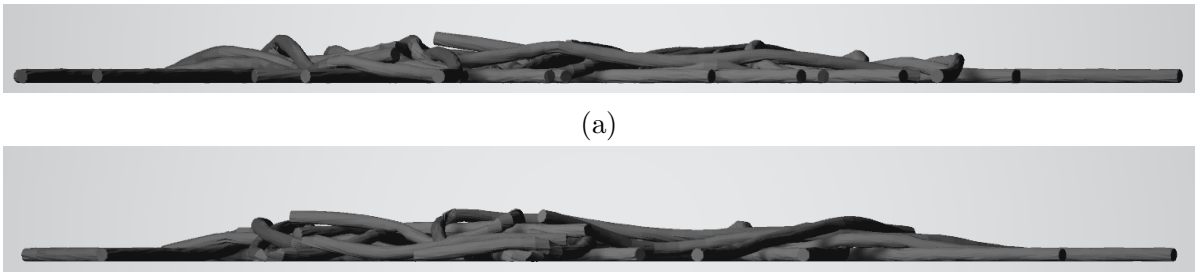


(a) (b)

Figure 7: Top view of system of 25 cellulose nanofibrils made with Blender.

many different sample scattering patterns needed to be averaged. In this manner, the different samples are like Monte Carlo iterations to provide an ensemble average for the film in question. The possibility to average over different subsets of the total image set illustrated the required number of samples to approach the ensemble average. The average GISAXS pattern for 256 samples is shown in figure 10.

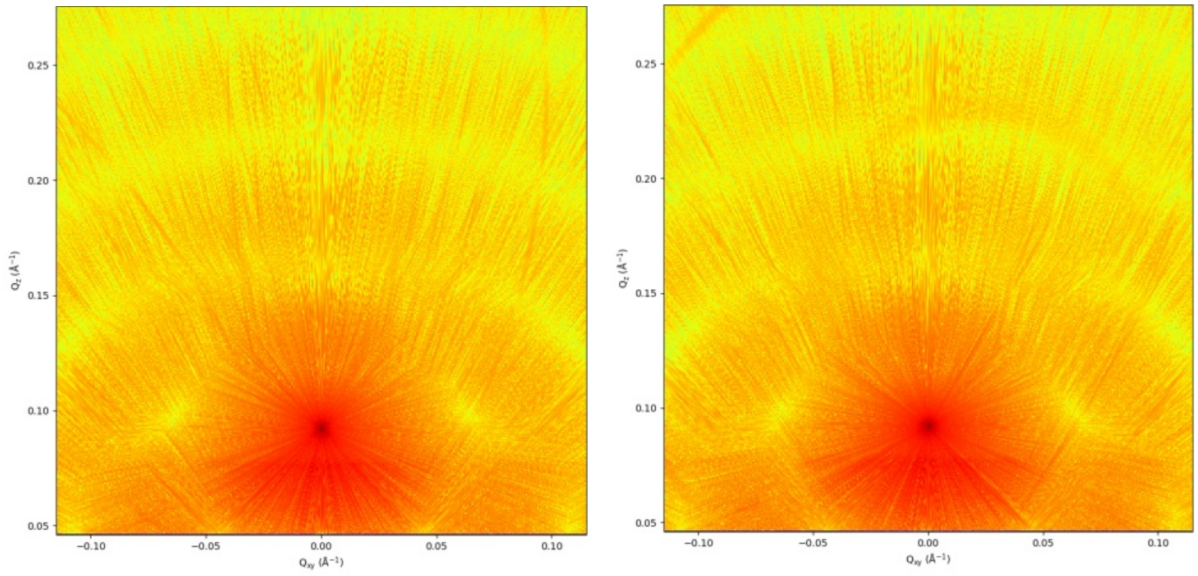
This ensemble scattering pattern retains many of the features of the individual scattering patterns, There is some very limited structure along Q_z above the specular peak.



(a)

(b)

Figure 8: Side view of system of 25 cellulose nanofibrils made with Blender.



(a) (b)

Figure 9: HipGISAXS patterns produced for the first and second AgNW sample generated in blender. The parameters employed in the gen _config.py file to produce these images were $\theta = 0.4°, \delta$ =2.88×10^{-6}, β =2.62×10^{-8} and $\lambda = 0.954$.

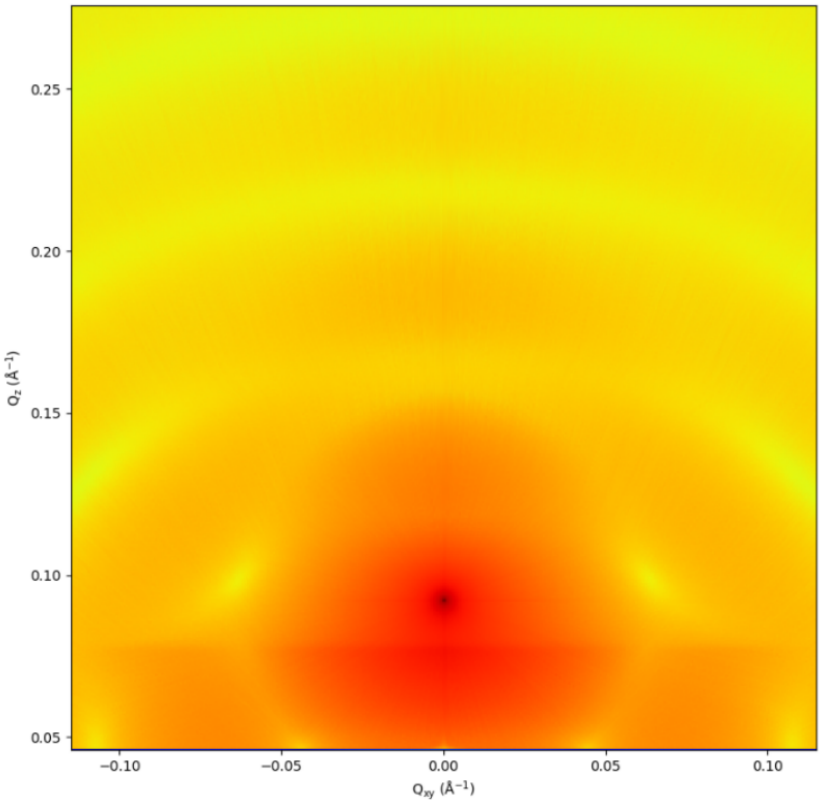


Figure 10: Average GISAXS pattern produced from 256 different AgNW films produced in blender, each with 2360 films.

4. Discussion

While this project demonstrated HipGISAXS-2.0 and Blender's capacity to produce GISAXS patterns of real space models, and a working pipeline for generating sample data was produced, many improvements remain.

Blender was found to be an extremely versatile software, which in its basic to intermediate functionality can be used confidently within a week. There is a wealth of resources online for developing complex models. Furthermore, blender files could be opened remotely on Maxwell, and python scripts then executed on these files - allowing the process to be automated. However, automation of blender sample generation was certainly not streamlined in this project. The current approach involves opening a base blender file containing the array of AgNWs, which are then randomised and allowed to settle into a film. At a chosen frame number, relevant data is extracted and blender then quits, before the original base file is reopened and the process restarted. Opening and quitting blender is necessarily quite demanding; a solution that involves looping from within blender and resetting the base array there would likely be far quicker and more logical. Moreover, while this automation process works diligently in the background on a remote computer, it is still a nuisance for it to open blender's user interface at all. A future

solution should implement the term'-b' in the batch script to allow this process to run completely in the background, without the user interface opening. This would require the animation to be fully rendered in blender, and not just in the user interface. This in itself may be more demanding, and necessitate changes to the frame handling - it remains to be seen if avoiding the user interface would be beneficial. Rendering of physics animations in blender can be improved by 'baking the physics', this should certainly be explored to improve data production rates with blender. Additionally, scaling of the model needs to be improved. Currently the whole array and substrate needs to be scaled in coordination with the dimensions inputted into the python script. For example, the model used for the patterns above are scaled so that 1m in blender corresponds to 1μm. Moving to a set up without an initial base array, and either scaling the substrate and the model together from the script or removing the current substrate entirely would potentially help with this. Eventually a user could be able to input the number of wires and the various model scales without altering the base file.

HipGISAXS-2.0. demonstrated its capacity to generate sample patterns using the data captured from blender. Currently, the user still needs to update a lot from within the various scripts, but this will naturally improve as work is continued to move from HipGISAXS-1.0 to the newer version. Among the features yet to be fully moved from v1.0 to 2.0 is the input and use of triangulated meshes made in blender. This feature was one of the strongest aspects of HipGISAXS-1.0 and will markedly increase the number of models that can be analysed with HipGISAXS-2.0 when transferred. In the context of this project, use of the triangulated meshes would allow full treatment of the pentagonal cross sections - rather than just treating them as cylinders. This, however, would vastly increase the time to simulate GISAXS scattering patterns. To remedy this, the HipGISAXS-2.0 scripts should be implemented with GPUs, rather than CPUs. This should be possible with the existing HipGISAXS-2.0 scripts using Cupy, but various issues prevented this when attempted on Maxwell. Future projects should focus on simulating scattering patterns from triangulated data from blender (with the aid of GPUs), enabling the analysis of a wealth of diverse and complicated 3D models.

Additional work has also been explored using blender and the data it generated. Images taken from the user interface of the film, from above, were crudely analysed in python to find an area coverage value. This could be improved by using a camera in the blender model, and having it take an image at a certain frame. This would require rendering the animation in full, again demanding, but would enable automatised area coverage analysis. Similarly, the numpy data collected from blender contains sufficient information to calculate a rough value of the film volume filling. This simply requires the film surface to be approximated and the enclosed film volume compared with the volume of the individual wires. Comparison of the volume filling and area coverage with simulated GISAXS patterns could be aided by machine learning to infer information about the former from the latter. This avenue should certainly be explored in future work.

5. Conclusion

Convincing models of silver nanowire and cellulose nanofibril meshes were produced using blender. With a base file in blender, an automation routine was developed to generate large volumes of data, either locally or remotely (i.e. on Maxwell). HipGISAXS-2.0 was used to produce GISAXS patterns that had many correct features. Improvements to the model scaling and automation, as well as the analysis of triangulated blender data, were identified as key points for future work. The project has demonstrated a proof of principle for this approach, and identified physical variables such as volume filling that could soon be analysed within the framework of Blender and HipGISAXS-2.0.

6. Acknowledgements

I would like to thank Professor Roth for giving me the opportunity to work on such an interesting project and to be part of the welcoming community at DESY.
Thank you to DESY as a whole and in particular the organisers of the DESY summer student program for organising this incredible experience.
I am extremely grateful to Dinesh Kumar from Berkeley for his steadfast commitment to meeting and discussing the project code every week.
I would also like to thank the Postdocs and postgrads in the Professor Roth's working group who helped so much and were always friendly.
In particular I am thankful for the help and advice from Yusuf Bulut, whose input was invaluable when I was struggling with the project.
Thank you to Andre Rothkirche for frequently helping with all my IT and shell scripting issues.
Lastly, thank you to the amazing cohort of DESY summer students who have contributed to some amazing memories during my time here.

7. Appendix

This appendix contains the most important source code from this project, code for randomising the blender nanowires and collecting the resulting data. The remaining code can be found in the project repository, hosted on github [6]; in particular, code for averaging images and area coverage analysis. Additionally, tables of parameters for the existing blender models are also given in this appendix.

A. Blender

A.1. Data Collection

The code below is executed with the base blend file, 'agnw_2360_cplx.blend'. This code is specifically for automation, hence the call to quit blender after saving the numpy data. Without this call, the code could be pasted into blender's scripting window and edited with the base model directly.

```
1  """
2  Python code written to randomise a basis array of nanowires and let them settle on a
       substrate
3  in Blender. This relies on the initial blender file having a substrate and series of
       wires that
4  are placed in a collection 'AgNW'. Outputs numpy file containing scales, positions
       and euler
5  rotations of nanowires
6  Paul Vautravers 19/09/2022
7  """
8  import bpy
9  import numpy as np
10 import random
11 import mathutils
12
13 #assignment of nanowire collection
14 units = bpy.data.collections['AgNW']
15
16 def get_xyz_angles():
17 #function to collect the xyz coords and euler angles of each agnw
18 #returns xyz coords and euler angles
19
20     for i, obj in enumerate(units.all_objects):
21
22   #positions and angles are collected in temporary array
23         xyz_coords_euler_angles_temp = np.hstack((obj.matrix_world.to_translation()
               [:], obj.matrix_world.to_euler() [:]) )
```

```
24
25          if i == 0:
26              xyz_coords_euler_angles = xyz_coords_euler_angles_temp
27          else:
28
29              xyz_coords_euler_angles = np.vstack((xyz_coords_euler_angles,
                    xyz_coords_euler_angles_temp))
30
31      return xyz_coords_euler_angles
32
33  def randomise_agnw(r_mean,r_sig,l_mean,l_sig, shift ):
34  #function to randomise the scale, rotation and position of wires around initial
        array site
35  #inputs are the mean and standard deviation of radius and length respectively, plus
        a shift
36  #returns the radii and lengths of every wire
37
38  #units are considered in micrometres, e.g 10 for the mean length is a 10 micrometre
        long wire
39  #must be accounted for in hipgisaxs
40
41      radii_arr = np.array([])
42      len_arr = np.array([])
43
44      for obj in units.all_objects:
45
46    #objects must be selected in blender to operate on them
47          obj.select_set(True)
48
49    #radii and lengths of wires are Gaussian distributed
50          rand_radius = np.random.normal(r_mean,r_sig)
51          rand_length = np.random.normal(l_mean,l_sig)
52
53          radii_arr = np.append(radii_arr,rand_radius)
54          len_arr = np.append(len_arr,rand_length)
55
56    #x,y dimensions calculated from radius and applied to wires
57          rand_xy = rand_radius/(np.sqrt(2))
58          obj.dimensions = [rand_xy,rand_xy,rand_length]
59
60    #wires shifted randomly in x,y,z around their original lattice site
61          rand_translation = tuple(np.random.uniform(-shift,shift,size=3))
62          obj.location = obj.location + mathutils.Vector(rand_translation)
63
```

```
64    #angles distributed over 2 pi and object angle updated
65          rand_rotation = tuple(np.random.uniform(0,2,size=3))
66          obj.rotation_euler = mathutils.Euler(rand_rotation)
67
68    #object deselected before next iteration
69          obj.select_set(False)
70
71       radii_arr = np.reshape(radii_arr,(len(radii_arr),1))
72       len_arr = np.reshape(len_arr,(len(len_arr),1))
73
74       scale_arr = np.hstack((radii_arr,len_arr))
75
76       return scale_arr
77
78  def get_data(key_frame,scale_arr):
79  #function to collect agnw data at the specified frame
80  #inputs are the frame number to take data, and scale array to combine with
        coordinates
81  #returns a 'frame handler' which is then applied in a separate function below
82
83       #function defined within larger function
84       def xyz_handler(scene):
85
86            if bpy.context.scene.frame_current == key_frame:
87
88          #coords and angles collected and joined with scale array
89                xyz_angles_arr = get_xyz_angles()
90                xyz_angle_scales_data = np.hstack((xyz_angles_arr,scale_arr))
91
92                #data is deleted for wires that fell through the substrate
93          #occurs when there's a large number of wires
94                del_indices = np.array([],dtype=int)
95                for i,val in enumerate(xyz_angle_scales_data[:,2]):
96                     if val < 0:
97                          del_indices = np.append(del_indices,int(i))
98                xyz_angle_scales_data = np.delete(xyz_angle_scales_data,del_indices,0)
99
100               #data saved under temporary filename in same folder as this script
101               np.save("agnw_temp",xyz_angle_scales_data)
102
103         #blender quits so that automation can open the script again
104               bpy.ops.wm.quit_blender()
105
106      return xyz_handler
```

```
107
108 def register(update):
109 #function to apply the frame handler defined above
110     bpy.app.handlers.frame_change_post.append(update)
111
112 def main():
113 #wires randomised and relevant data collected at specified frame number
114
115     bpy.app.handlers.frame_change_post.clear()
116     scale_array = randomise_agnw(0.06,0.001,10,0.1,5) #default values
          0.06,0.001,10,0.1,5
117     bpy.ops.screen.animation_play()
118     register(get_data(400,scale_array)) #default value of 400
119
120 main()
```

The only thing that the user should change when executing this script is the randomise_agnw arguments, which are all given in micrometers. Among the shortcomings with this code is that the model should be scaled in the UI if the user wants to provide significantly different scales. Multiple improvements could be made to this script, for example generating the wires rather than randomising an existing array, allowing to vary the number of wires easily and perhaps avoid any interaction with the UI.

This script is to be used as part of a shell script, as below; however, the script could be pasted into the blender scripting window and used there, providing the call to quit blender is removed.

A.2. Model Parameters

The physics parameters for AgNW and CNF are provided in this section as screenshots from the blender user interface.

A.2.1. AgNW

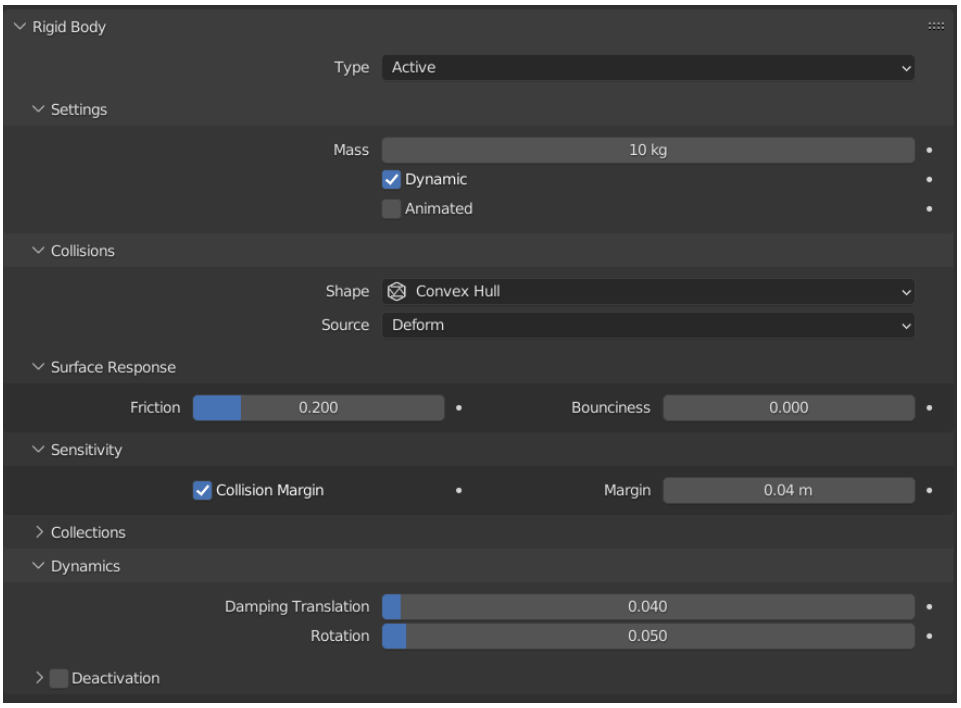


Figure 11: Rigid body physics parameters used for the silver nanowires in this report.

The substrate had simple parameters, mesh type, 0.1 friction, no bounciness and sensitivity of 0.04m

A.2.2. CNF

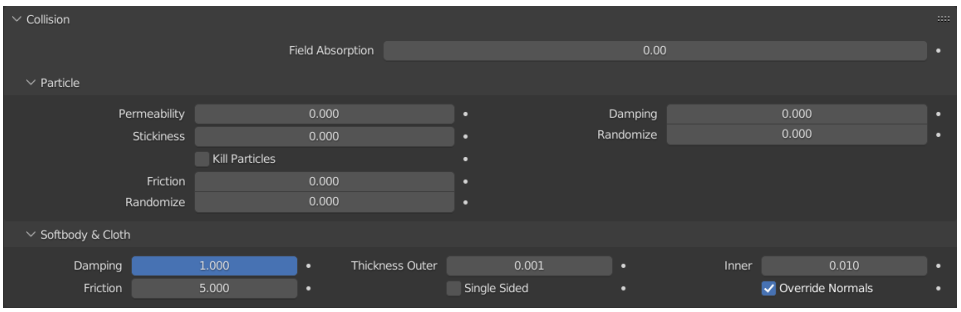


Figure 12: Collision physics parameters used for the cellulose nanofibrils in this report.

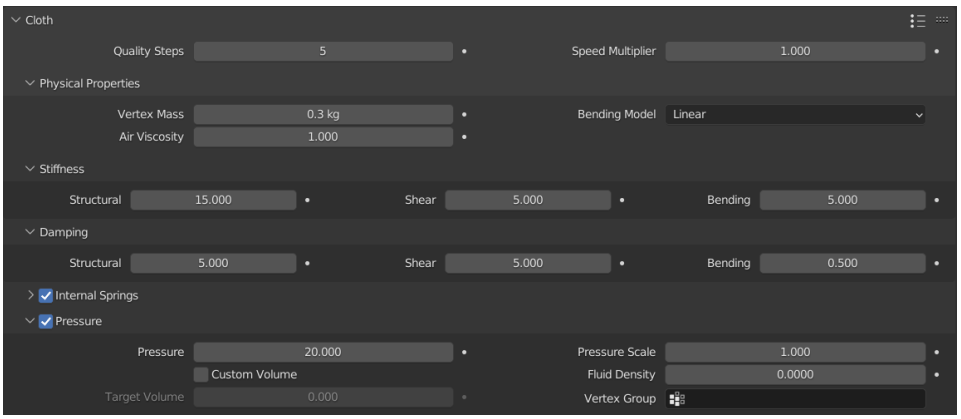


Figure 13: Cloth physics parameters used for the cellulose nanofibrils in this report.

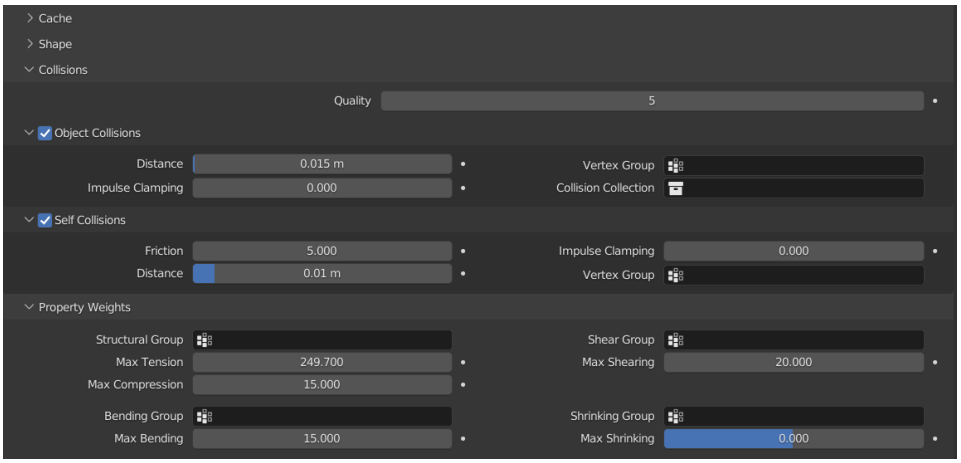


Figure 14: Additional Cloth physics parameters for reactions to collisions, used for the cellulose nanofibrils in this report.

The collision parameters for the substrate in this case were: damping; 0.1, thickness outer; 0.02, inner; 0.2 and friction; 80.

B. Automation

This appendix provides the final shell script used for this project, generating data from blender and simulating the GISAXS scattering pattern in each iteration. The segmented code for the data collection and GISAXS simulation is provided on the project repository [6], as well as the .bat script to produce blender data on a local microsoft device.

```
#!/bin/sh
# components of updated file name
 file_prefix ="agnw_2360_s"
 file_suffix =".npy"
# for loop from 1 to 256, in integer steps
for i in {1..256..1};
do
  #blender opened with base file and python script applied to it
  blender agnw_2360_cplx.blend -P agnw_script_2360_cplx.py
  #new file name constructed and python output overwritten
  new_file_name="agnw_2360_s$i.npy"
  mv agnw_temp.npy $new_file_name
  echo "Sample data $new_file_name produced!"

  #hipgisaxs simulation
  python3 main_u1.py $i
  echo "Sample $i GISAXS pattern produced!"
done
```

The only thing the user needs to change here is sample range, i.e. to start at 1 or 4, or finish at 256 or 10. Naturally the files should all be stored in the same folders.

The user is reminded that to use this code on Maxwell at DESY, they must call 'module load maxwell blender' in the terminal beforehand.

References

[1] Gilles Renaud, Rémi Lazzari, and Frédéric Leroy.

[2] P. Müller-Buschbaum. *A Basic Introduction to Grazing Incidence Small-Angle X-Ray Scattering*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[3] Alexander Hexemer and Peter Müller-Buschbaum. Advanced grazing-incidence techniques for modern soft-matter materials analysis. *IUCrJ*, 2(1):106–125, 2015.

[4] Rémi Lazzari. isgisaxs: A program for grazing-incidence small-angle x-ray scattering analysis of supported islands. *Journal of Applied Crystallography*, 35(4):406–421, 2002.

[5] Slim T. Chourou, Abhinav Sarje, Xiaoye S. Li, Elaine R. Chan, and Alexander Hexemer. hipgisaxs: A high-performance computing code for simulating grazing-incidence x-ray scattering data. *Journal of Applied Crystallography*, 46(6):1781–1795, 2013.

[6] P. Vautravers. Blender and hipgisaxs-2.0 gisaxs simulations with real space models. https://github.com/paulvautravers/HipGISAXS-DESY-2022, 2022. Accessed: 2022-10-31.