MATBOX: Microstructure Analysis Toolbox documentation

An introduction to microstructure analysis through an open-source MATLAB application

v1.0b – Matbox2021c

May 19, 2021

Francois L. E. Usseglio-Viretta

Center for Integrated Mobility Sciences, National Renewable Energy Laboratory (NREL)

About the authors/contacts:

To report bugs, suggest new algorithms or simply provide feedbacks or comments, all of them being appreciated, please send an email to the main developer: Francois.UsseglioViretta@nrel.gov with the email object starting with the text [MATBOX]. You can also let a message in the discussion section of the GitHub repository at https://github.com/NREL/MATBOX_Microstructure_analysis_toolbox/discussions

Contributions (excluding third-party software):

- Main developer and documentation writer: Francois Usseglio-Viretta (NREL)
- GUI development of the particle generation module: Prehit Patel (NREL)
- Contrast correction documentation/examples for *adapthisteq* in the ROI, filering and segmentation module: Elizabeth Bernhardt (NREL)
- Additive generation algorithm (energy-based method): Aashutosh Mistry (Argonne National Laboratory) and Partha P. Mukherjee (Purdue University)
- Meshing module code adapted for monolithic mesh: Jeffery Allen (NREL)
- Integration of TauFactor in the characterization module: Samuel J. Cooper (Imperial College London)
- Discussion and alignment with DOE's objectives: Kandler Smith (NREL)

About the National Renewable Energy Laboratory:

The National Renewable Energy Laboratory is a national laboratory of the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, operated by the Alliance for Sustainable Energy, LLC. Please visit https://www.nrel.gov/index.html for more information, as well as the transportation research / energy storage home page https://www.nrel.gov/transportation/energy-storage.html to see how NREL is contributing to the development of high-performance, cost-effective, and safe energy storage systems to power the next generation of electric-drive vehicles.

Acknowledgments

This software was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding for algorithm development was provided by the U.S. DOE Vehicle Technologies Office's Computer-Aided Engineering of Batteries (CAEBAT) program[1] (program manager Brian Cunningham). Application of the algorithm for fast-charge analysis was provided by the eXtreme Fast Charge Cell Evaluation of Lithium-Ion Batteries (XCEL) program (program manager Samuel Gillard).

I would like to thank my NREL colleagues, Donal Finegan, Andrew Colclasure and Jeffery Allen, as well as Samuel Cooper from University College London, for interesting discussions and suggestions about microstructure analysis. I would particularly like to thank my NREL supervisor, Kandler Smith, for its support and guidance all along the development of this toolbox. Lastly, I would like to thank Matt Keyser, our group manager, for giving me the opportunity to work at NREL.

License and third-party software included in the toolbox:

- The present toolbox

This toolbox uses BSD license. NREL Software Record number SWR-20-76. License file is in the repository root folder.

- Third-party licenses

The toolbox uses third-party open-source software: Taufactor[2], from Dr. Samuel Cooper for tortuosity factor calculations and Iso2mesh[3], from Dr. Qianqian Fang for microstructure meshing, respectively used in the microstructure characterization and meshing modules. In addition, the toolbox uses third-party algorithms: xlscol[4] from Kevin Crosby, DataHash[5] from Jan, Noiselevel[6] from Masayuki Tanaka, and Screencapture[7] from Yair Altman, respectively used to help exporting data in excel files, compare microstructure states in iterative algorithms through their hash value, estimate image noise level before and after image filtering, and copy figure from MATLAB App design. Algorithm from Dr. Aashutosh N. Mistry[8] is used for one of the methods employed in the microstructure generation module, for additive phase generation.

Taufactor, xlscol, DataHash, Noiselevel, Screencapture, and Mistry's algorithm use BSD license and are included in the toolbox repository.

Iso2mesh uses GPLv2 and is not included in the toolbox repository. User will have to download Iso2mesh manually. Installation steps are detailed in §III.

Third-party licenses are available in the 'Third-party licenses' folder.

Please contact us if you would like to implement your algorithm in this toolbox.

How to cite:

If you produce results using the toolbox, or use some or parts of the algorithms contained within the toolbox, please quote them accordingly:

- **For any results produced with the toolbox, please quote**: F. L. E. Usseglio-Viretta et al., *MATBOX: An Open-source Microstructure Analysis Toolbox for microstructure generation*, segmentation, characterization, visualization, correlation, and meshing, SoftwareX, submitted
- **If you are generating additive phase with the energy criterion method, then please <u>also</u> quote**: A. N. Mistry, K. Smith, and P. P. Mukherjee, *Secondary Phase Stochastics in Lithium-Ion Battery Electrodes*, ACS Appl. Mater. Interfaces 10(7) pp. 6317-6326 (2018), https://doi.org/10.1021/acsami.7b17771
- **If you are calculating tortuosity factor, then please <u>also</u> quote**: S.J. Cooper, A. Bertei, P.R. Shearing, J.A. Kilner, and N.P. Brandon, *TauFactor: An open-source application for calculating tortuosity factors from tomographic data*, SoftwareX, Volume 5, 2016, Pages 203-210
- **If you are generating unstructured mesh, then please <u>also</u> quote**: Q. Fang and D. A. Boas, *Tetrahedral Mesh Generation From Volumetric Binary and Gray-scale Images*, Proceedings of IEEE International Symposium on Biomedical Imaging 2009, 2009, Pages 1142-1145

The open-source microstructure community

Information below may be outdated at the time you read it, as it concerns third parties.

Open-source data:

Having access to numerous microstructure volumes is extremely valuable as not only it allows researchers to develop and test algorithms even though they do not have the experimental capabilities to produce such data in the first place, but it also allows establishing correlation between different microstructure properties as such statistics analysis requires a lot of data to be relevant. Lastly, it makes easier to reproduce and validate works done by other groups, as data is shared. Copyrights/open-source policy/legal information are specific for each institution, please make sure to understand them before starting using their data. As well, please adequately quote the institutions when you are using their data. Below is a non-exhaustive list of microstructure data library:

- **National Renewable Energy Laboratory** (NREL) hosts a variety of Lithium-ion cathode (nickel manganese cobalt [NMC]) and anode (graphite) electrode data samples, calendered and uncalendered with different loadings. Gray level and segmented data are both available at the NREL battery microstructure library: https://www.nrel.gov/transportation/microstructure.html

- **ETH Zürich** hosts NMC-based Porous Electrodes, Polyethylene (PE) Separator, Commercial Graphite Electrodes, and various commercial anodes at the ETH Zürich Battery Microstructure Project: https://made.ee.ethz.ch/research/open-source-data-and-software/battery-microstructure-project.html

Other open-source microstructure software:

NREL is not the only institution developing open-source microstructure-related software. Many other projects are listed below (non-exhaustive list):

- **ImageJ**, https://imagej.net/Welcome, and **Fiji**, https://fiji.sc/ are open platforms for scientific image analysis, with a lot of plugins thanks to their large community.

- **Neper**, http://www.neper.info/ is a software package for polycrystal 3D generation and meshing.

- **PoreSpy**, http://porespy.org/ is a collection of image analysis python tool used to extract information from 3D images of porous materials.

- **Openpnm**, http://openpnm.org/, an open source pore network modeling package.

- **The Microscopy Image Browser**, http://mib.helsinki.fi/index.html, is a high-performance Matlab-based software package for advanced image processing, segmentation and visualization of multi-dimensional (2D-4D) light and electron microscopy datasets.

TABLE OF CONTENTS

I. INTRODUCTION

Microstructure, in MATBOX, is a generic term used to describe a multi-phase/heterogenous medium, whatever its scale. It actually does not restrict to microstructure scale. This toolbox has been developed to support lithium-ion battery modeling, for which the heterogeneous medium is a porous electrode, and the relevant scale is the microstructure, thus the toolbox name. Because of this electrode-centered approach, algorithms proposed in this toolbox are relevant for battery electrode microstructures, although they may be relevant for other medium as well.

Electrode performances are strongly correlated with their microstructure, which then justify their analysis. To investigate their microstructure, they must be observed first experimentally. A large variety of imaging techniques exist[9]. While the experimental imaging is out of the scope of this document, it is important readers non familiar with electrode microstructures have a basic knowledge of the main techniques. Three are often reported in the battery/electrode literature: Scanning Electron Microscopy (SEM) for 2D imaging, and Focused Ion Beam Scanning Electron Microscopy (FIB-SEM) and X-ray computed tomography[10,11] (CT) for 3D imaging. These imaging techniques are not limited to electrode materials[12].

While some information can be extracted from a 2D image, anisotropic (i.e., with an orientation) and connectivity-related (e.g., diffusion, percolation) properties require a 3D volume to be accurately quantified. Field of view, i.e., the 3D volume imaged, in the battery field is typically ranging from ~$1e^3$ to ~$1e^6$ μm^3, with a voxel (pixel in 3D) size ranging from ~$1e^3$ to ~$1e^6$ nm^3, with smaller voxel size associated with smaller field of view. Typical electrode materials have particle diameter ranging from ~$1e^1$ nm (e.g. silicon anode) to ~$1e^1$ µm (e.g., graphite anode and nickel manganese cobalt oxides cathode). The raw data obtained from 3D imaging experiment is a 3D array for which each voxel has a brightness value, that reflects the local sensitivity of the medium in regards with the imaging experiments. While analysis can be done at this stage, it is not the desired state as voxels do not identify each phase precisely. There is instead a distribution of grey level value (usually represented through a histogram). Volumes are then 'segmented' or 'binarized' so that each voxel is assigned to a given phase, typically the pore or void domain, and the solid phase. If there are more than one solid phase, then each solid phase has its own voxel unique value to distinguish it from the other solid phases. This step is called segmentation.

Figure 1 provides an example of a battery electrode microstructure while figure 2 illustrates the main uses of MATBOX, which are further detailed below.

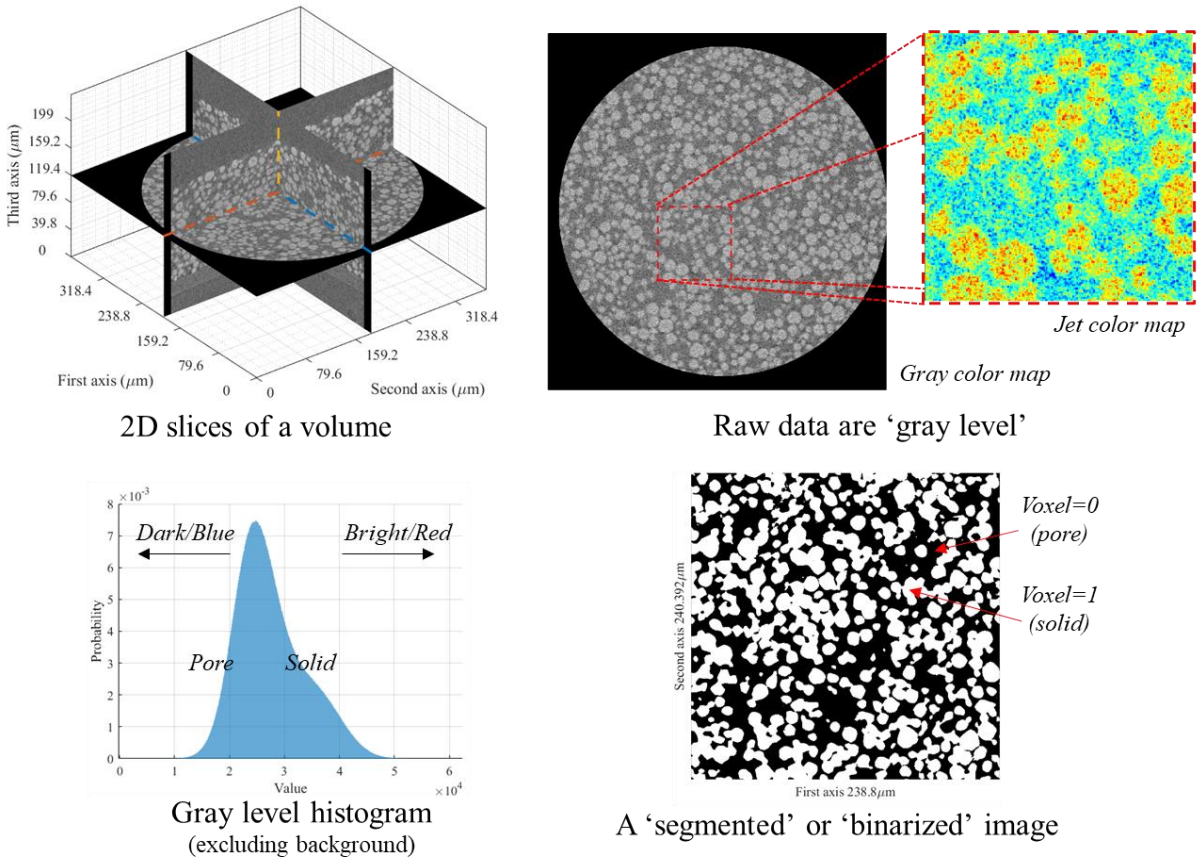


Figure I-1. (Top left) 3D visualization of a nickel manganese cobalt oxide electrode. (Top right and bottom left) the volume is initially described with gray level values. Bounds depend on the image datatype (in this case, 16 bits unsigned integer, thus from 0 to 65535). (Bottom right). After segmentation, each voxel has a non-ambiguous phase identification. Volume used for this example comes from the NREL open-source microstructure library[13]. Images have been generated with the present toolbox.

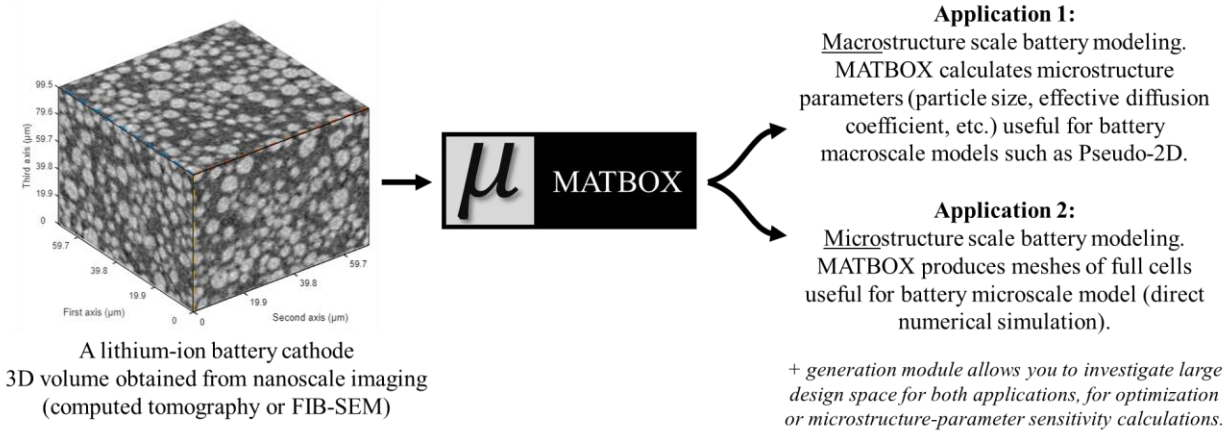


Figure I-2. Main applications of MATBOX.

Microstructure analysis encompasses different activities or tasks, covered to a certain extent by the toolbox modules briefly introduced below and illustrated in figure I-3:

- *Microstructure generation* (cf. §IV). Three-dimensional microstructure imaging requires expensive equipment, time, and skilled workers that limit availability and quantity of data. Furthermore, the microstructure design space is restricted by existing electrode materials, preventing upstream microstructure scale modeling. Microstructure generation algorithms allow to numerically generate 3D microstructures that can be used for design space analysis otherwise not possible. It is valuable to evaluate 'what if' microstructures that can later motivate manufacturers to experimentally test promising architectures. Microstructure generation includes active material generation and additives generation.
- *Region of interest (ROI) selection, filtering and segmentation* (cf. §V). Before analyzing microstructure data, several preliminary tasks are required. The field of view may be cropped and/or rotated to select the region of interest. Data type may be converted for ease and size reduction. Volume may be upscaled or downscaled to match the desired image resolution. Image quality can be evaluated through different metrics, that in turn can indicate what segmentation approach is best suited. Image quality can be improved through contrast correction and/or image filtering in order to facilitate segmentation. Lastly, segmentation identifies phases from gray level values. In addition, microstructure parameters sensitivity analysis with the segmentation can be performed to provide an error estimation of the segmentation.
- *Microstructure characterization and homogenization* (cf. §VI). Microstructure characterization and homogenization consists in extracting relevant parameters for macroscale models, such as particle size or effective diffusion, from the complex microstructure. Module offers automated voxel size dependence and representative volume element analysis, as well as batch calculations to characterize multiple volumes in a row.
- *Microstructure and results visualization* (cf. §VII). Other modules generate 2D graph figures to illustrate results. However, 3D visualization may be more relevant in some occasion. This module produces 3D visualizations of the microstructure itself and of microstructure parameters defined voxel-wise (e.g., particle size). Note that to visualize such microstructure parameters you will need first to run the characterization module to produce results. The module also provides side-by-side comparison of gray level and segmented volumes, to visually evaluate the relevance of the segmentation, with overlay and checkerboard visualization to help the comparison.
- *Properties correlation* (cf. §VIII). Establishing correlations between microstructure properties provides a better understating of their intrinsic relationships, help identifying independent parameters, and allow predicting microstructure properties. The most classic example would be the tortuosity factor as a function of porosity. This module allows you to load selected results from multiple microstructure characterization calculations quickly to produce

correlation matrix-type figures. This module is especially valuable if your database contains dozens of microstructures and can reveal unsuspected correlations.

- *Meshing* (cf. §IX). Microstructure scale finite element modeling requires a mesh representation of the microstructure. This module can produce both structured (with regular grid and cuboid shape) and un-structured (with mesh density control and smooth interface) tetrahedron-based 3D meshes from either a single segmented microstructure stack tif file or a combination of several stack tif file to mesh a full cell. Please note this module does not do finite element modeling.

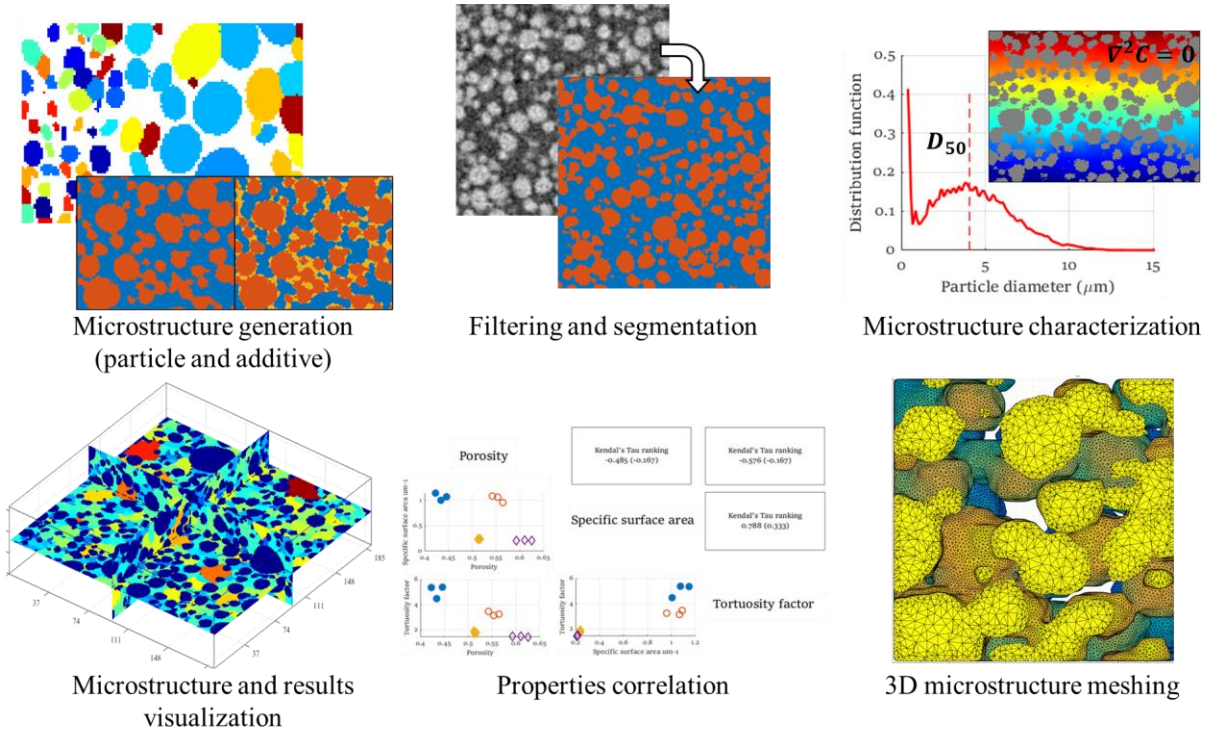


Figure I-3. Illustration of the Microstructure analysis toolbox modules

Toolbox modules are connected through their inputs/outputs, as described in figure I-4. The characterization module occupies a central place as it can takes inputs from the generation module and the filtering and segmentation module, while its outputs are used for the properties correlation module and the microstructure and results visualization module.

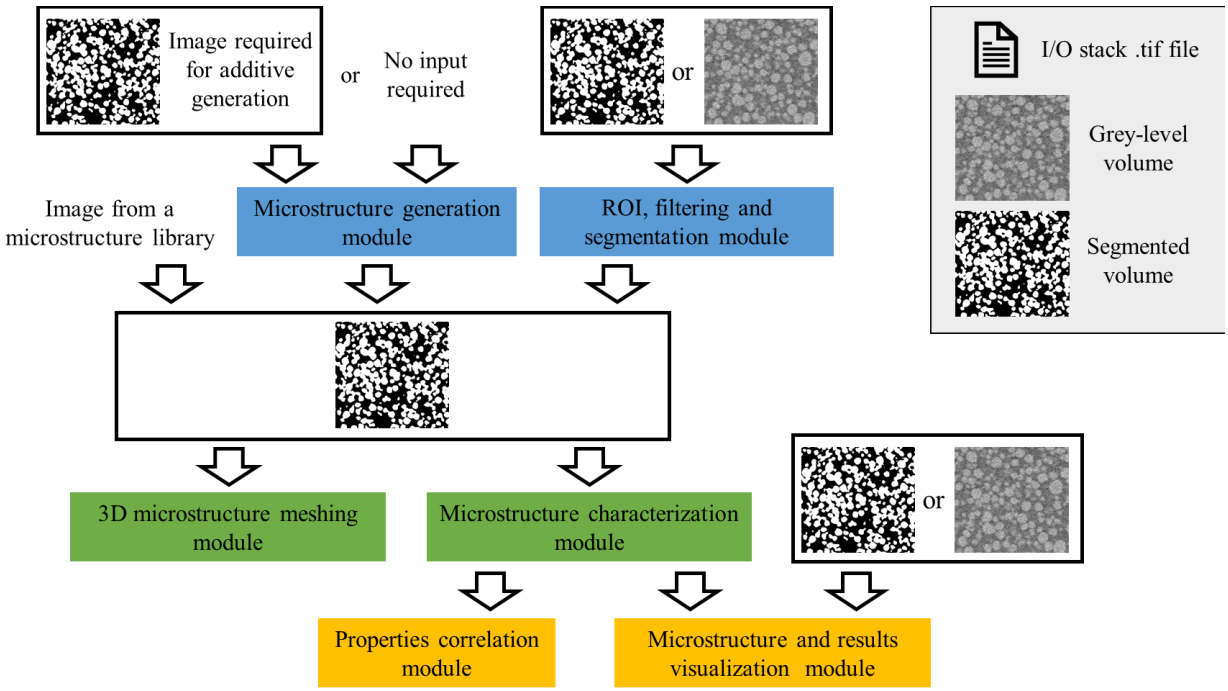


Figure I-4. Microstructure analysis toolbox module inputs/outputs connectivity.

This documentation can be considered as an entry point for newcomers in the microstructure field as it provides introduction, comments, references for each section while also describing algorithms in detail. At the end of this document, you can find a list of references, sorted by topics, that could help you getting started in the wonderful world of microstructure analysis (cf. §XII-1).

II. INSTALLATION AND REQUIREMENTS

Copy the folder MATBOX_Microstructure_analysis_toolbox from the GitHub repository https://github.com/NREL/MATBOX_Microstructure_analysis_toolbox in your computer and add it to the MATLAB path, **including subfolders** (MATLAB ribbon interface: Set Path/Add with subfolders).

Third-party software installation:

- Iso2mesh: Copy https://github.com/fangq/iso2mesh in your computer and add the folder in your MATLAB path. Or download it from https://github.com/fangq/iso2mesh. Make sure to choose the binary for your OS (Windows, Unix, or MacOS). You can check Iso2mesh is installed/in the Matlab path by typing:

```
>> iso2meshver
iso2mesh toolbox version: 1.9.5-Rev
```

  Once Iso2mesh files have been copied, added them to the MATLB path. Then open file vol2mesh.m and change line 46 (this modification only matters if you mesh polycrhistalline architecture):

```
if(length(unique(vol(:)))>100000 && dofix==1)
```

- TauFactor is already included in the toolbox. Although if you want to use it as a standalone app, or to be sure to use the last up to date version of TauFactor, please download it from the MATLAB file exchange. You can check TauFactor is installed/in the Matlab path by typing:

```
>> help TauFactor
 %%%%%%%%%%%%%%%%%%%   TauFactor
```

- Other third-party codes are already included in the toolbox repository.

The toolbox has been tested with MATLAB 2020b. The MATLAB toolbox "Image Processing" is required. To know if you this toolbox, simply type "ver" in the Matlab command windows (cf. Fig. II-1).

The toolbox has been tested in Windows. The meshing module performs better in a Unix environment (e.g., Ubuntu) due to a better RAM memory management.

For a better experience you can then package MATBOX in a MATLAB app. To do this, open with MATLAB app designer the file Main_menu.mlapp located in \MATBOX_Microstructure_analysis_toolbox\src\Main_menu\, then go to the MATLAB app designer ribbon and click on Designer / Share / MATLAB app and follow instructions. You will have to add the documentation pdf manually (look at "shared ressources and helper files"). Then go to the MATLAB ribbon and click on APPS / Install App and select the file MATBOX Microstructure Analysis Toolbox.Mlappinstall. Congratulations, MATBOX is installed (cf. Fig. II-2)!

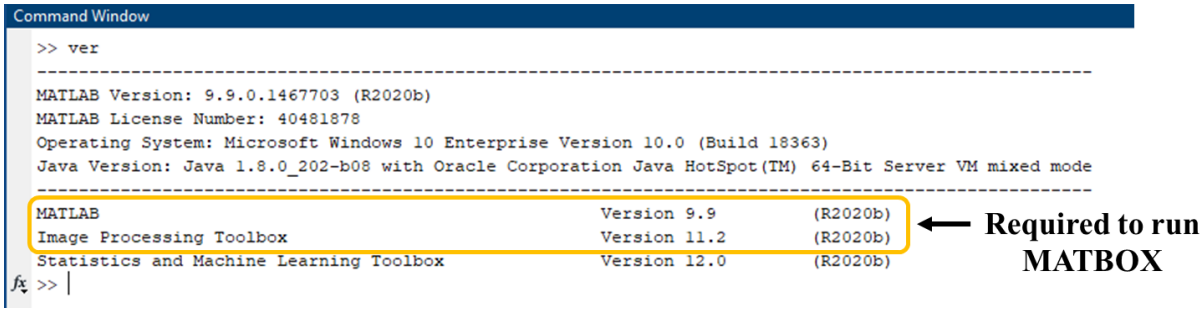


Figure II-1. MATBOX toolbox requirements

Figure II-2. MATBOX is installed and ready to be used as a MATLAB app!

III. HOW TO USE THE TOOLBOX

Once MATBOX installation is complete, open with MATLAB app designer the file Main_menu.mlapp located in \MATBOX_Microstructure_analysis_toolbox\src\Main_menu\ and run it to call the main menu (cf. Fig. III-1). Alternatively, if you have packaged the app as explained in §II you can call the main menu directly by clicking on MATBOX's icon on the MATLAB apps ribbon.

The main menu provides you with a choice of modules, with each module having its own graphic-user interface. Please refer to their respective section (cf. §IV-IX) for help. You can hover the mouse on each module to see a brief description. Clicking on "NREL", "About" and "Repository" will bring you, respectively, to the NREL energy storage webpage, the NREL transportation data and tools webpage (MATBOX will eventually be referenced there) and to the MATBOX GitHub repository. Clicking on documentation will summon this document in pdf.

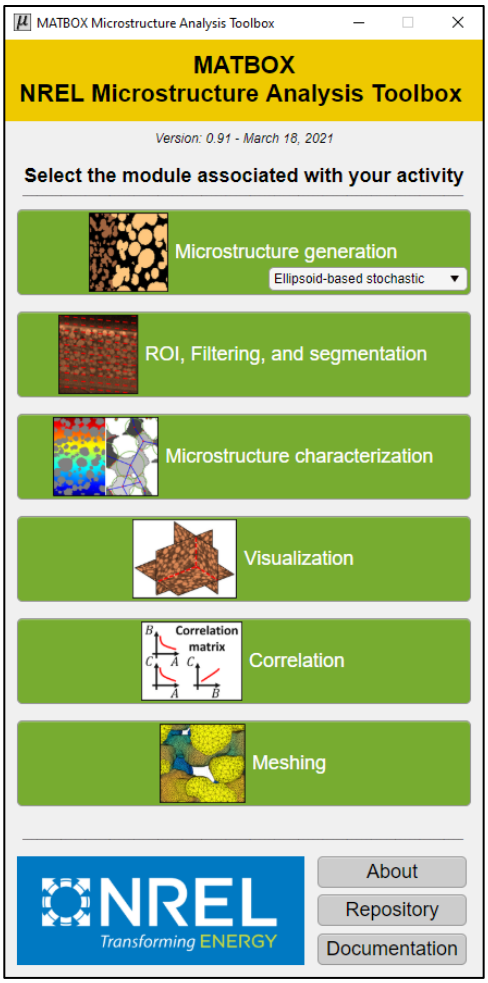


Figure III-1. Microstructure analysis toolbox main menu

IV. MICROSTRUCTURE GENERATION

1. Why microstructure generation and module purpose

a. Particle scale

Microstructure generation allows investigating various microstructures otherwise not available through experimental imaging or even manufacturing. Different methods are available in the literature[14,15], that can be classed, for instance, according to the type of microstructure produced. Some methods prioritize credibility over control: the generated microstructures look very similar to a real microstructure, but with a limited control of the particle features[14,15]. Other methods prioritize control over credibility: the generated microstructures look obviously generated, but with a fine control of the particle features. Both approaches are valuable and choosing one is dependent on your application.

From a microstructure design space analysis point of view, controlling finely the particle features is very interesting. For instance, if you are asking questions such as 'What is the impact of X on Y?', with for example X being the particle size distribution or the particle alignment and Y the effective diffusion coefficient or some metric of the electrochemical cell performance, being able to finely choose X is important. Furthermore, determining particles metrics from a microstructure volume is very challenging, and strongly depends on the chosen method (cf. §VI-5f), therefore choosing X beforehand as an input parameter of the generation algorithm is very valuable. Say otherwise, with a credibility over control approach, X can still be determined afterwards but with uncertainty, while with a control over credibility approach, X is an input and does not require to be calculated. The in-house generation algorithm of this module has been developed in the frame of a microstructure design space analysis, to investigate 'what if' electrodes in order to find promising lithium-ion battery electrodes suitable for fast charge application. Therefore, the algorithm proposed in this module follows the control over credibility approach.

Another possible application of microstructure generation consists in testing existing microstructure parameter relationships or establishing new ones. For instance, tortuosity factor-porosity and specific surface area-particle diameter-porosity relationships[16]. These correlations are microstructure dependent (e.g., they are different if the microstructure is sphere-based or ellipsoid-based) and have a range of validity (e.g., high porosity range). Generation algorithm can produce numerous microstructures to test shape dependence and investigate a large parameter range. Such microstructure parameter relationships are very valuable as they are typically used in macroscale electrochemical models. A control over credibility approach suits better such problem as microstructure parameters need to be clearly determined to be relevant.

b. Additive scale

Experimental imaging has intrinsic limitations depending on the method selected that may result in not being able to distinguish one or several phases of the medium. For instance, computed

tomography of lithium-ion battery electrodes such as NMC and graphite does not resolve the additive phases (carbon-black, for improved electron conductivity and binder, for mechanical integrity), that are not distinguishable from the pore domain[17]. Such additives have practical impact, as they modify the effective electrolyte diffusion coefficient and the interfacial electrochemically active area between the active material and the electrolyte[8,17,18]. Therefore, not being able to represent them hinders electrochemical model predictions. To remedy this issue, the additive phase can be generated in the pore domain of a microstructure. Various strategies have been considered in the literature, physics-based[8] and deterministic[19] approaches.

In this module, an original simple in-house deterministic approach is proposed in addition to the physics-based approached reported in[8].

2. Particle generation

a. Algorithm explained

The algorithm allows generating microstructures with the following specifications:

- N-solid-phase microstructure generation ellipsoid-based algorithm.
- Volume fractions, particle size distribution, particle elongation distribution, and particle orientation distribution defined for each slice along the microstructure thickness. It allows generating graded microstructures (i.e., with properties varying along their thickness) with in-plane heterogeneities (since distributions functions are defined per slice).

The algorithm main numerical features are described below:

- Stochastic (i.e., probabilistic) iterative algorithm, with particles generated sequentially.
- Each time a new particle is generated, all probabilities for the slices that cut through the new particle are updated accordingly to match the user-defined volume fraction, particle size, elongation, and orientation distribution targets. For instance, after a particle is generated, the probability that a particle of the same phase is generated later in the same region is decreased accordingly in order to match the user-defined volume fraction targets (as a result probabilities for the other phases are adjusted too). Ideally, with an infinite domain's size such probability correction would not be required: if you toss a coin an infinite number of times, each side appears with a probability of 50% in average. But if you only consider the first ten times, it is very likely the probability will be off. The correction used in the algorithm ensures that volume fraction targets (and similarly for size, orientation and alignment distributions targets) are achieved even though the volume size is limited and that we 'toss' particles only a limited number of times. It allows generating microstructures that match user-defined parameters with a relatively low domain's size, effectively reducing CPU time. Nevertheless, such correction artificially reduces heterogeneity: performing representative volume element analysis with such generated microstructures is not recommended.

Updating probability to generate a particle that belongs to the phase *i* within the slice *x* is done as follow. First the remaining volume fraction $r_i(x)$ is deduced, and a random number is picked from 0 to 1 to test if a particle will be generated locally. As the generation process is moving on, the current volume fraction will increase, and eventually reach the target volume fraction, thus preventing overshooting the target volume fraction.

$$r_i(x) = \frac{target\ volume\ fraction_i(x) - current\ volume\ fraction_i(x)}{\sum_{j=1}^{N} target\ volume\ fraction_j(x) - current\ volume\ fraction_j(x)} \qquad \text{[IV-1]}$$
$$if\ rand < r_i(x)\ then\ try\ generating\ a\ particle\ within\ slice\ x$$

Updating probabilities of a distribution function, i.e., the probability P a parameter (such as particle diameter) takes a value v_j among n different values is done by normalizing the distribution function depending on the current distribution:

$$P_i(x, v_j) = \max\left(\frac{P_i(x, v_j)_{target} - P_i(x, v_j)_{current}}{\sum_{j=1}^{n} P_i(x, v_j)_{target} - P_i(x, v_j)_{current}}, 0\right) \qquad \text{[IV-2]}$$

For instance, if the local target size distribution is 40% with a diameter v_1 and 60% with a diameter v_2 (n=2) and that during the generation process, we have generated 10% particles with a diameter v_1 and 50% with a diameter v_2, (we still need to generate the remaining 40%, 100% corresponding to reaching the target volume fraction of the current phase), then the probability distribution considered at this stage of the generation process is: (0.4-0.1)/(0.4-0.1 + 0.6-0.5)=0.75 for v_1 and (0.6-0.5)/(0.4-0.1 + 0.6-0.5)=0.25 for v_2. At this stage, the chance to select the diameter v_1 are higher since we have already almost reached the target for v_2.

- Particle overlapping is controlled independently for each phase *i* - phase *j* permutation, with a gradation allowing fine control of the overlapping. If overlapping is authorized, there is the risk a particle gets overlapped successively by other particles up to the point its initial morphology has been excessively degraded. To avoid this, you can set a constraint to the minimum volume a particle must conserve, taking as reference its volume when first generated. Overlapping is particularly interesting as it enables generating microstructures with very high percolation or connectivity (similarly with actual volumes obtained with computed tomography lithium-ion battery electrodes) and allows reaching high density microstructures (effectively avoiding packing density limit). Indeed, for ideal non-overlapping spheres, the maximum packing density is 74,0%, which decreases to 63.4%[20] if spheres are randomly distributed. This means the minimum porosity you could achieve after numerous tentative with non-overlapping spheres would be 36.6%.

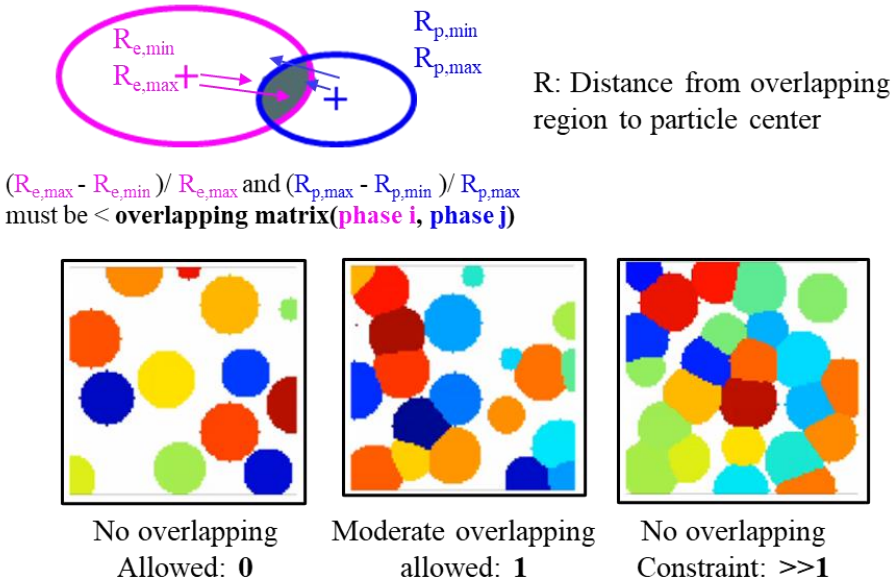


Figure IV-1. Particle overlapping. Color is coded with particle id.

- Naïve user-defined parameters that correspond to an impossible microstructure will lead the algorithm to stall. For instance, a bi-phase microstructure (pore and one solid) with unisize non-overlapping sphere and a porosity of 10% is theoretically impossible due to packing density.
- The generation order can be tuned to avoid the algorithm to stall for some microstructures that could be challenging to generate without generation order control. To illustrate this, let us consider the case of a bi-phase material (pore and one solid) with a bi-modal size distribution for the solid, with the small particles being significantly smaller compared with the large particles. If we try generating the solid without order control, we may generate smaller particles that will block the introduction of the larger ones. Chance of success are much higher if we decide to first generate the large particles, and only then the small ones, as illustrated in the figure below. To reproduce this generation behavior, the user can define several 'generation pass' the algorithm will iterate on. For each pass, the user can decide the volume fraction ratio each phase should reach before the algorithm moves to the next phase. In our example, instead of generating a bi-phase, the user would generate a tri-phase material (pore, small particles, large particles,) and ask for 2 generation pass, with for the first pass a ratio 0.0 for the small particles and 1.0 for the large particles, and for the second pass, a ratio 1.0 for both phases. Once done, the two solid phases can be easily merged.

```
Loop over pass
   Loop over phase
      While volume fraction(phase) ≤ target ratio (pass) x target volume fraction(phase)
         Generate new particle
```

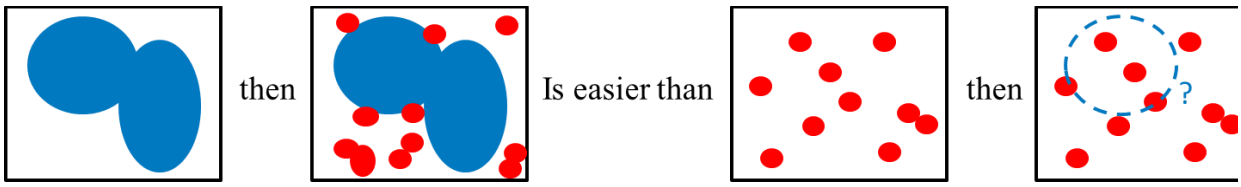


Figure IV-2. Controlling generation order increase chance of success

- Once the generation is finished, an in-house upscaling algorithm can be used. It is much faster to generate using a coarse grid resolution, and then upscale to the desired image resolution while generating directly at the fine scale.

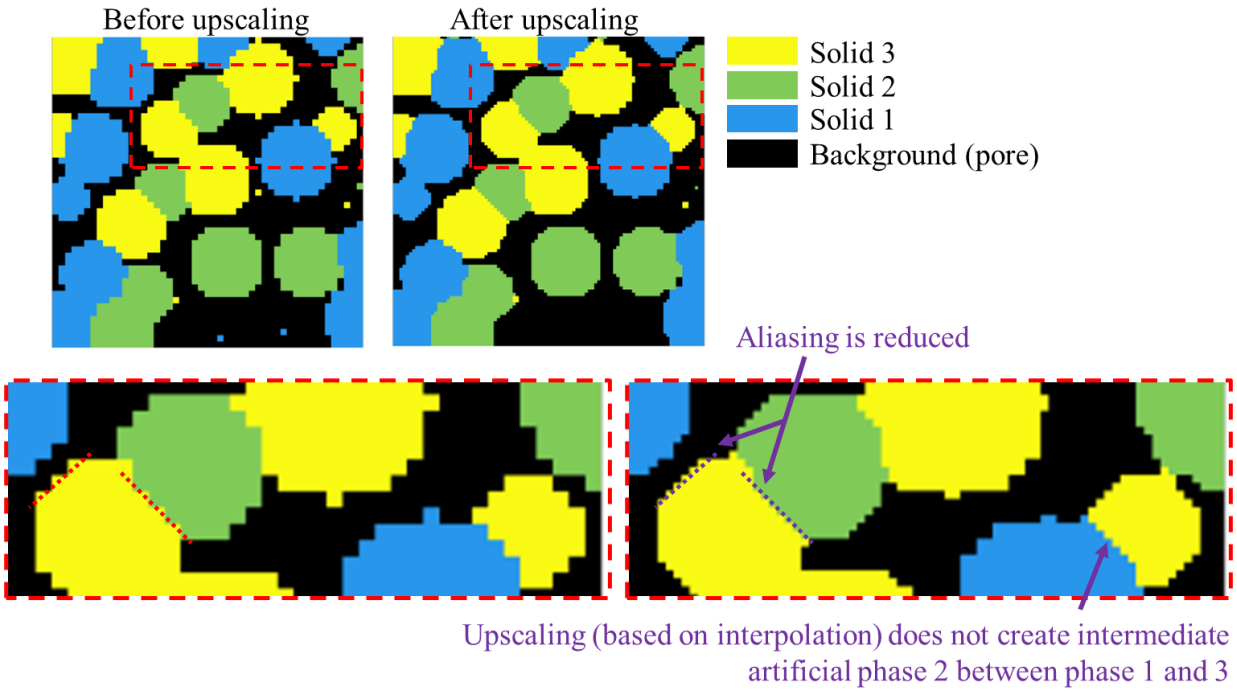


Figure IV-3. Custom upscaling algorithm works for n-phase and reduce aliasing

The generation algorithm is available in function_generate_ellipsoid_microstructure.m and is presented in detail in figure IV-4.

Loop over pass

Loop over phase

While volume fraction(phase) <= phase(phase).fillratiotarget(pass) * desired volume fraction(phase)

If new particle has been generated

- Update volume fractions, diameters (dx), elongations (dx/dy, dx/dz) and rotations (rx, ry, rz) histograms probability only for the slices z modified by the new particle (*Chance to pick a certain phase, diameter, elongation, and orientation for a new particle are modified by the existing distribution).*
- Update index of un-assigned voxels

- Choose randomly the particle center (x,y,z) among the un-assigned voxels

If rand < remaining volume fraction(phase, z)

- Choose diameter dx from the size histogram(phase, z) (*each slice has its own histograms, allowing user-defined in-plane heterogeneities)*

If diameter==1

- Simple case 1-voxel particle. New particle has been generated. Continue while loop.

Else (n-voxels particle case)

- Choose elongation dx/dy and dx/dz from their respective elongation histogram(phase, z)
- Create ellipsoid in a subdomain of size dx, dy, dz
- Choose rotation rx, ry, rz from their respective rotation histogram(phase, z)

If elongation~=0 and rotation ~=0 (particle is not a perfect sphere)

- Apply rotation matrix to the ellipsoid subdomain

- Crop the ellipsoid subdomain if it does not fit within the whole domain (e.g. sphere whose center is at the domain's edge will be cut in half).
- Select subdomain from the whole domain that overlaps with the ellipsoid subdomain

If subdomain is empty (no particles have been generated yet)

- Insert ellipsoid: subdomain = ellipsoid subdomain. New particle has been generated. Continue while loop.

Else (One or several particles have been already generated within this subdomain)

- Calculate intersection volume (overlapping): subdomain ∩ ellipsoid subdomain

If subdomain ∩ ellipsoid subdomain **is empty (there is no particle overlapping)**

- Insert ellipsoid: subdomain = subdomain + ellipsoid subdomain. New particle has been generated. Continue while loop.

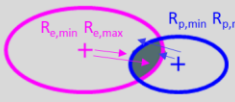
Else (new particle is overlapping with existing particles)

Loop over all existing particles within subdomain

If particle ∩ ellipsoid satisfies user-defined overlapping criteria, continue testing the other existing particles. **If** one particle does not meet one of these criteria, continue while loop.

User-defined overlapping criteria:

1st criterion



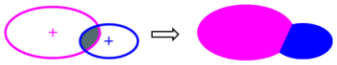
Euclidean distance (bwdist) between all voxels of the overlapping region and the two centers

$(R_{e,max} - R_{e,min})/ R_{e,max}$ and $(R_{p,max} - R_{p,min})/ R_{p,max}$ must be < max_overlapping(phase, particle phase)

2nd criterion

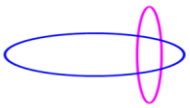
(Volume particle – overlapping volume/2) / Volume particle when created must be > Minimum_particle_volume(phase). Checked for particle and ellipsoid.
Loss of volume due to overlapping is cumulative, and should not exceed a certain value in order to preserve the particle shape and characteristics when created.

- Distribute all the overlapping region between the existing particles and the ellipsoid



Using distance map (bwdist) of the overlapping region (parallelized)

- Check particle continuity has been preserved for the existing particles and the ellipsoid

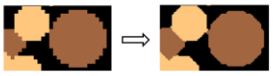


Connectivity is checked with bwlabeln
If such case is detected, continue while loop.

- Insert ellipsoid and overwrite existing particles of the subdomain.
- Update volume fraction for the phases of the modified existing particles

New particle has been generated. Continue while loop

- Upscale generated microstructure if requested. Export 3D volumes in .tif and .mat



This allows to run the generation algorithm on a small domain (i.e., with a coarse resolution) to save CPU-time.

- Verify generated microstructure satisfy the user-defined inputs (volume fraction, particle size, elongation and rotation)

Figure IV-4. Microstructure generation pseudo-code algorithm

b. How to use

Click on the 'microstructure generation' button from the main menu GUI, and then click on 'ellipsoids-based'. GUI is still in BETA but is usable.

Please follow instructions provided in each tab. Additional information is provided below.

In the first tab, ('Phase and volume fractions'), you need to enter the domain's size and the scaling factor. The generated volume in the example below will a 3D array of size (100,100,150), that will be upscaled afterwards to a size of (200,200,300). The voxel size is only indicative, as the algorithm is adimensional. You then need to enter the number of solid phase as well as their id (code) and name. In the example below, the generated volume would be three-phase materials: {pore, solid 1, solid 2}.

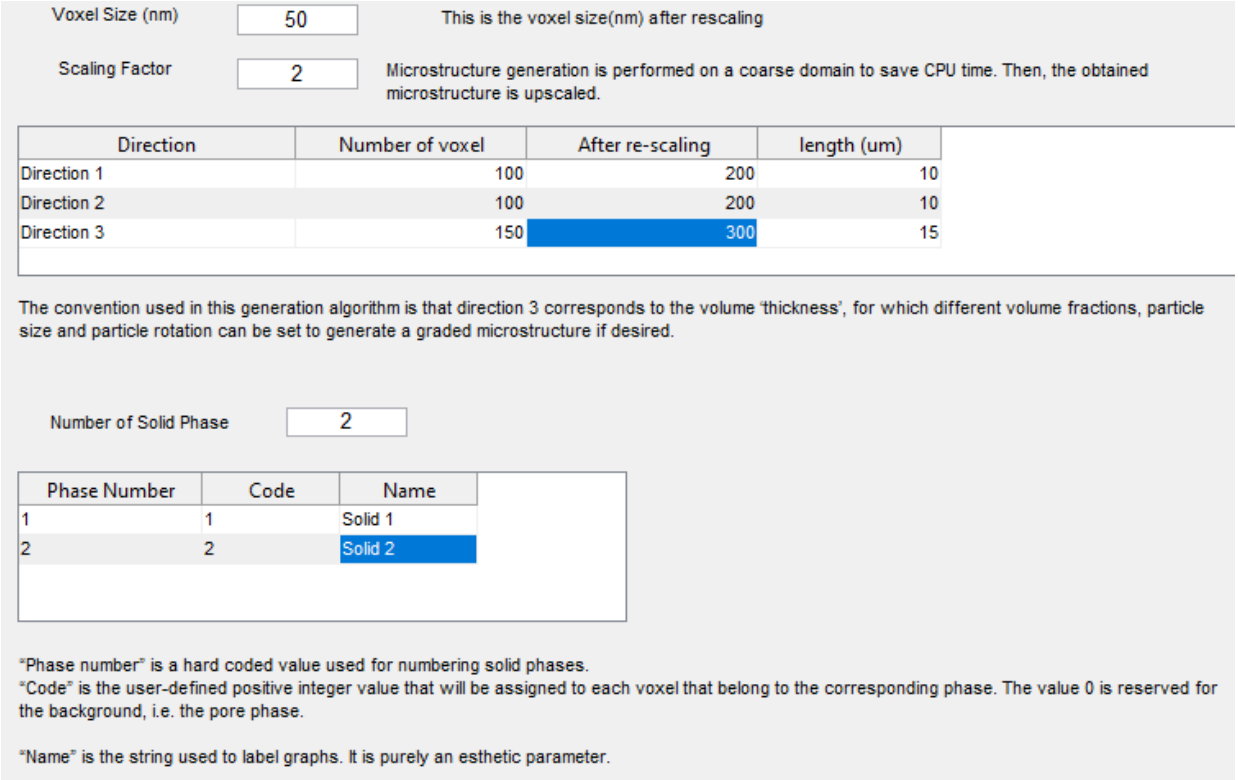


Figure IV-5a. Domain 'size and number of phases.

On the right side of the tab, you can choose the volume fractions for the solid phases.

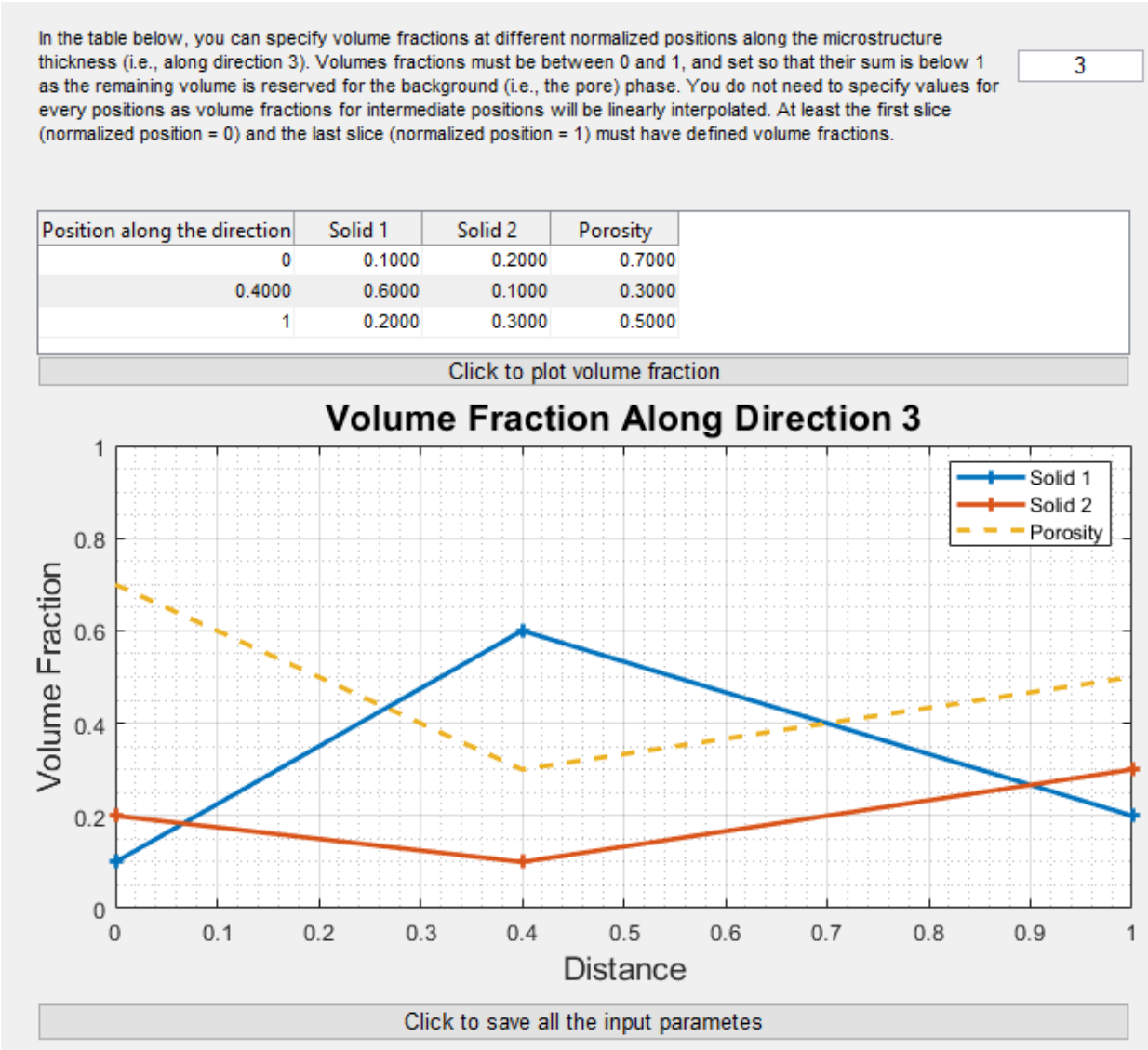


Figure IV-5b. Volume fractions are defined for few user-selected slices (here 3: z=0, 0.4 and 1, along the third axis), and linearly interpolated in between. Graph shows the parameter inputs.

Before moving the next tab, make sure to validate your choice by clicking on 'Click to save all the input parameters'.

On the second tab ('Particle size'), you can specify for each phase the size distribution.

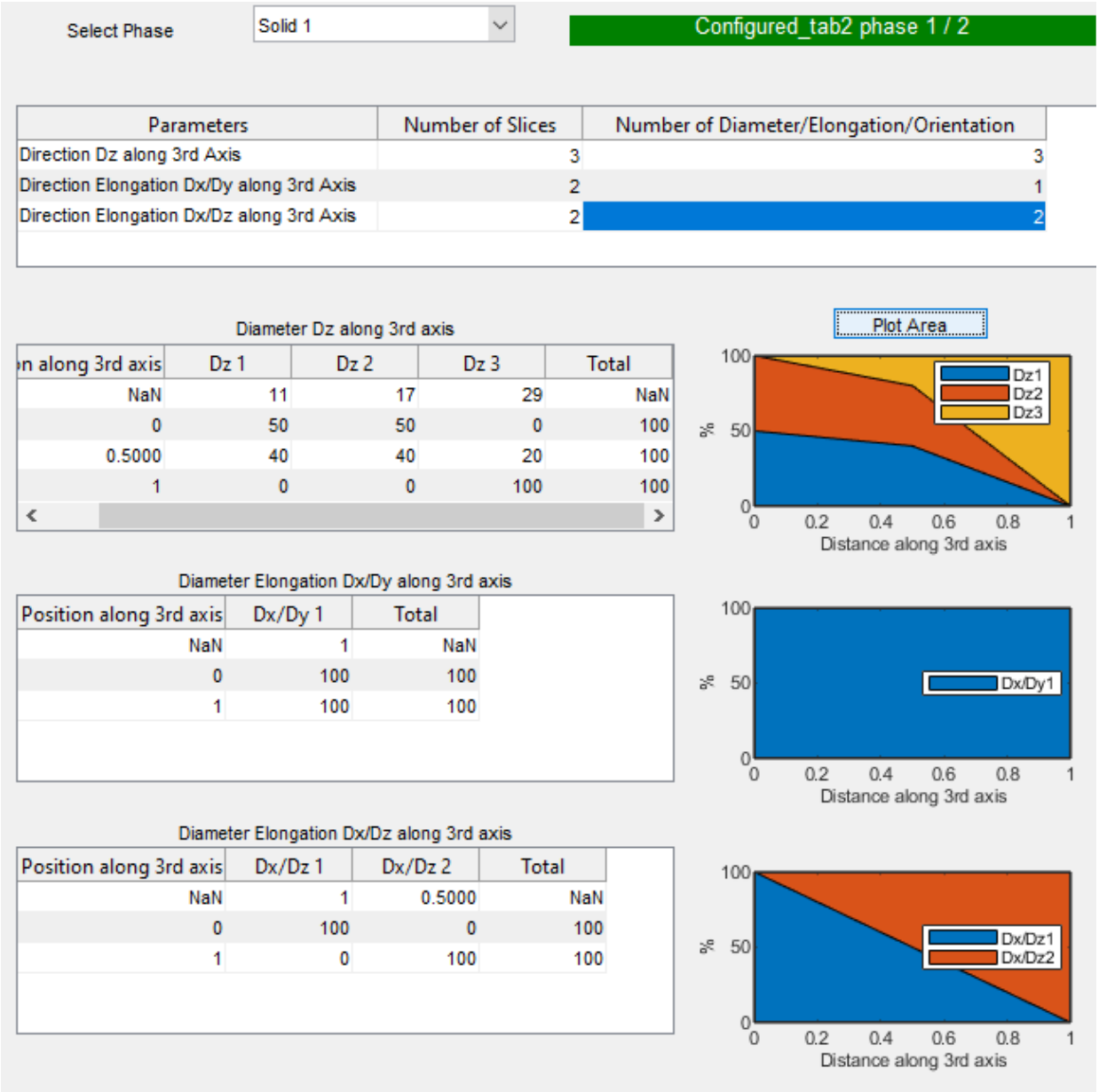


Figure IV-5c. Particle size is defined for three slices, and linearly interpolated in between. For instance, for the middle position (z=0.5) 40% of the particle phase have a 11-voxels length diameter D3, 40% a 17-voxels length diameter D3, and 20% a 29-voxels length diameter D3. The two other tables control particle elongation D1/D2 and D1/D3. Percentages correspond to the phase volume, and not to the number of particles.

On the right side of the tab, you can visualize a particle example to help you choosing the diameter parameters.

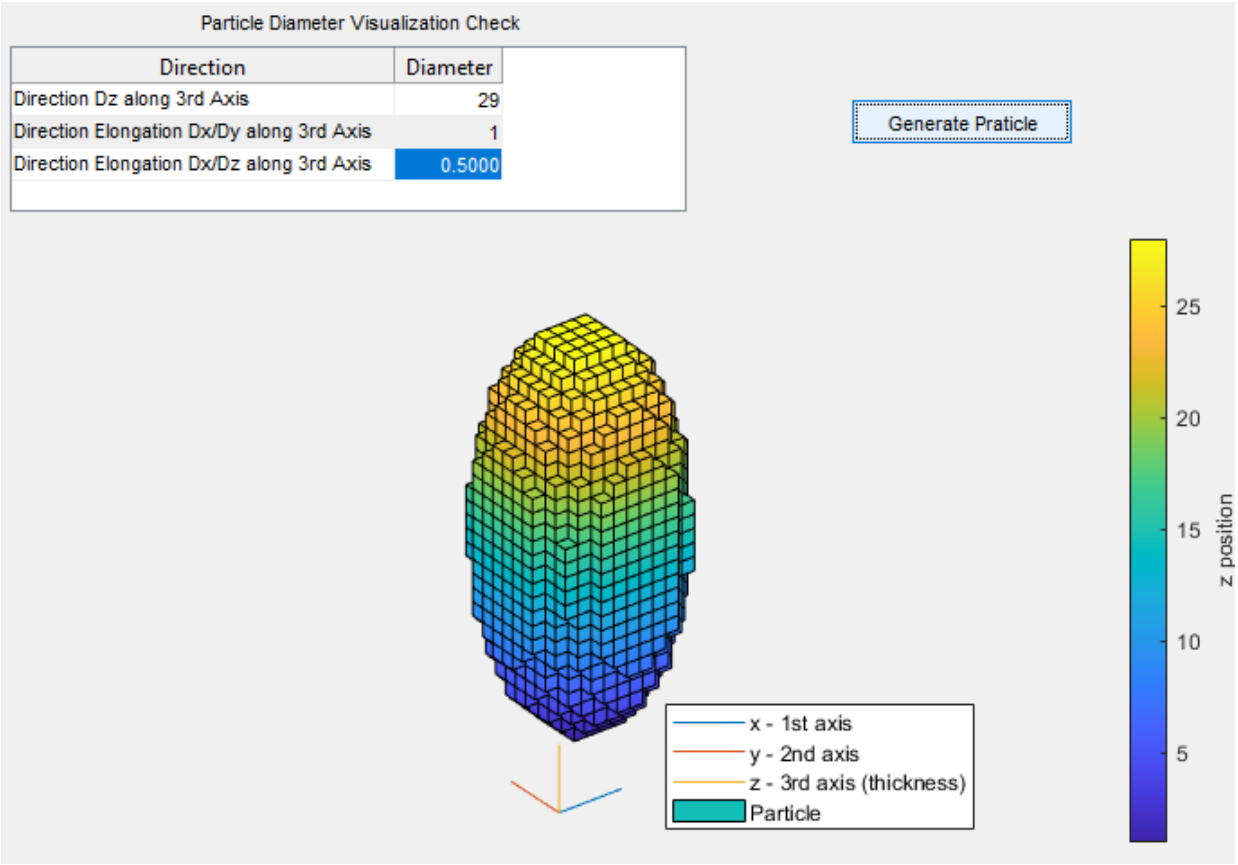


Figure IV-5d. Particle visualization for a given set of particle diameter and elongation.

You need to provide the diameter information for each phase (and save them) before moving to the next tab.

The third tab ('Particle orientation'), is similar with the diameter tab. As well, you can plot a particle example to have a better insight on the orientation parameters.

In the fourth tab ('Particle overlapping'), you can set the phase overlapping, the minimum volume to conserved, and the order of generation.

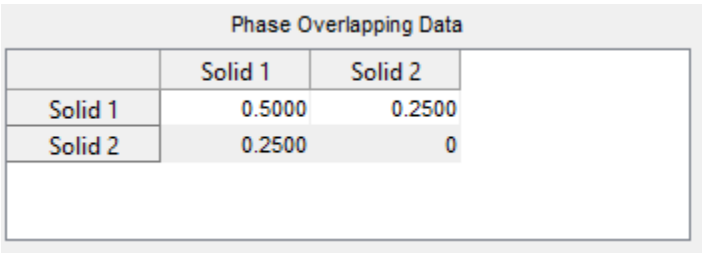


Figure IV-5e. Overlapping factor between particles of phase 1 is 0.5, and 0.25 between particles of phase 1 and 2. No overlapping is allowed between particles of phase 2.

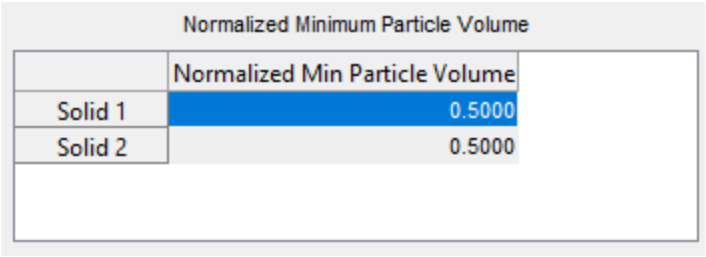


Figure IV-5d. Particle overlapping is restricted so that any particle of both phase conserve at least 50% of its initial volume when initially generated.

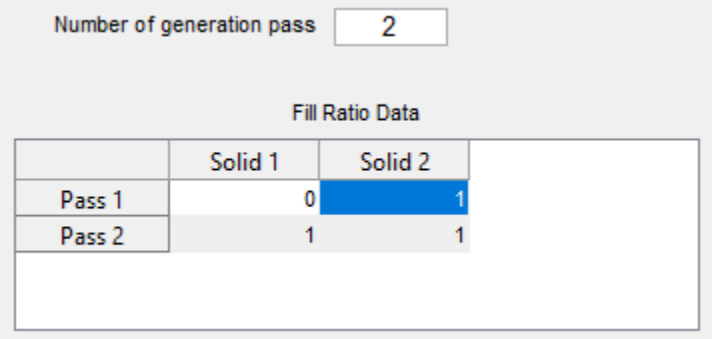


Figure IV-5e. In the first pass, all the solid 2 will be generated. In the second pass, the solid 1 will be generated.

In the next tab ('Post-processing and Save options'), select your save folder, and choose if you wish to calculate the tortuosity factor once the generation will be finished. To run the generation process, enter the number of times you wish to run the generation algorithm in the 'run code' tab, then click on the button 'Generate Microstructure'. After generation is finished, figures are automatically displayed that show the difference between the input parameters and what has been obtained.

In the 'generated microstructures' tab, volume fractions as well as tortuosity factor (if the option has been checked) are displayed for each generated volume, which is useful to verify the algorithm reproducibility. If you observe a significant difference between volumes, you need to increase the domain's size and/or to reduce the absolute diameter length (i.e., increasing the number of particles in the domain).

c. Examples of generated microstructures

Figure below illustrates generation capabilities offered by the particle generation algorithm. While microstructures look obviously generated, features such as particle size and alignment are clearly identifiable as they are inputs (control over credibility approach, cf. §IV-1a). Bilayers can be easily generated using four slices for the various microstructure parameters to setup a step function. For instance, position along direction 3: {0, 0.39, 0.41, 1} associated with

volume fraction $\{0.6, 0.6, 0.3, 0.3\}$. Pore former are simply obtained by re-assigning one of several phases to the background after the generation.

Example files are available in the repository at \Data_example\Numerically generated\.

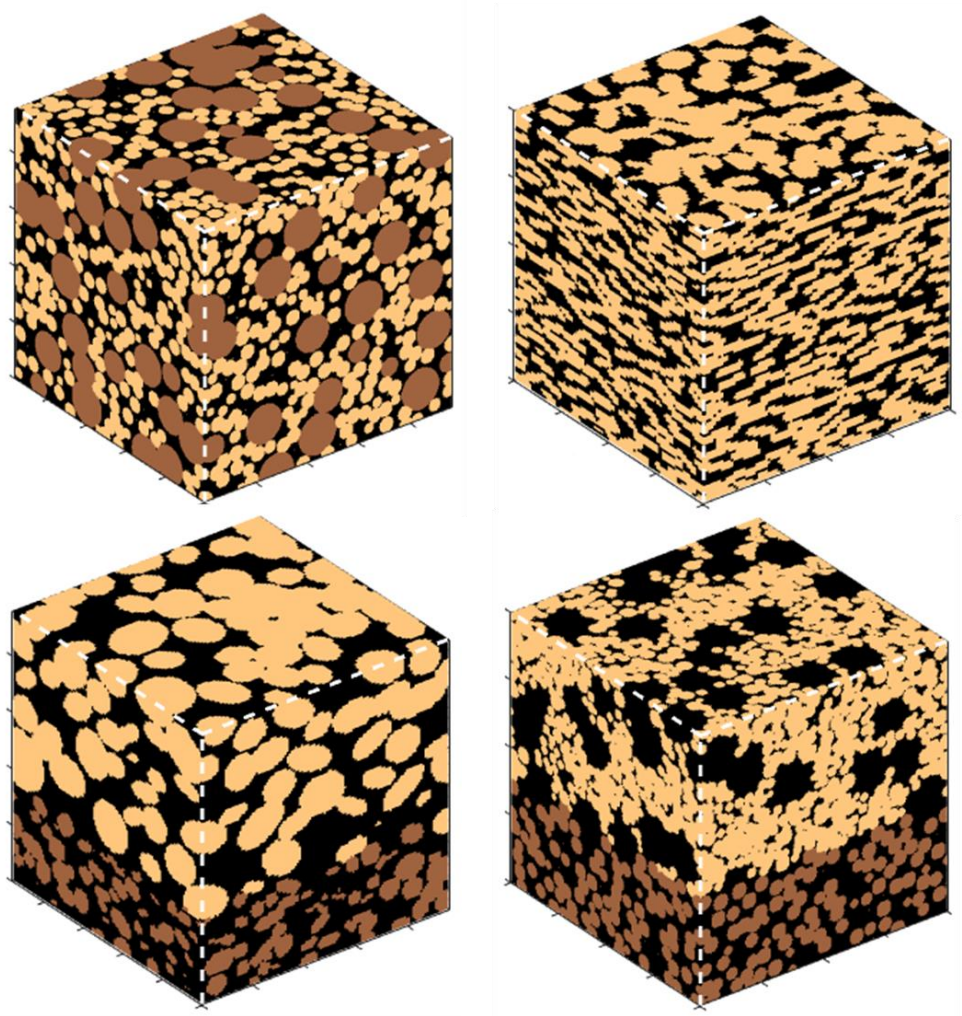


Figure IV-6. (Top left) bi-modal size distribution, (top right) ellipsoids with a preferential orientation, (bottom left) bi-layer with different porosity, particle size, particle alignment and orientation between the two layers, and (bottom right) bi-layer with pore former. Images have been generated using the visualization module (cf. §VIII).

3. Additive phase generation

a. Deterministic 'bridge' approach

Additive phase is generated assuming it is preferentially located between particles non far from each other, so that it will create bridges between particles. First the pore particle size $c - PSD(x)$ is calculated using a spherical approach (method detailed in §VI-5f-i). The original algorithm then iterates over a distance d, marking pore voxels for which $c - PSD(x) < d$. If the

as marked voxels are not enough to reach the additive target volume fraction, then all of them are assigned to the additive phase and then d is incremented until the additive volume fraction is reached. If too many voxels have been identified, only a subset of them are assigned to the additive phase to match its target volume fraction.

The sphere-size $c - PSD(x)$ tends to assign numerous one-voxel size particles at the phase boundary. To avoid generating a thin-like additive surface – which is not the goal of this particular generation algorithm – the user can specify a minimum distance ('minimum distance from boundary in voxel length', by default 1) for which voxels are not considered below.

The user can add some stochasticity in the process by only adding, for each increment of d, only a ratio of all the voxels that verify the criterion $c - PSD(x) < d$. In this case, voxels are added connected cluster per connected cluster until reaching the ratio. The parameter 'Randomize cluster selection' control the level of stochasticity (with a default value of 0, i.e. false, otherwise the value between 0 and 1 corresponds to the ratio).

```
binary_phase=zeros(size(Microstructure)); % Initialization
binary_phase(Microstructure == background_id) = 1;
[distance_bnd2bnd] = Function_particle_size_CPSD_Algorithm(binary_phase);
distance_bnd2bnd(distance_bnd2bnd==0)=9e9; % Remove background
distance_bnd2bnd(distance_bnd2bnd<=minimum_dist*sqrt(3))=9e9;

ntot = numel(Microstructure);
current_additive_volumefraction = 0; % Initialize
d=0; % Initialize
while current_additive_volumefraction < target_volume_fraction
    d=d+1; % Increment distance
    if d>minimum_dist*sqrt(3)
        potential_additives = distance_bnd2bnd<=d;
        n = sum(sum(sum(potential_additives)));
        if n>0
            if randomize_level==0
                if n/ntot <= target_volume_fraction-current_additive_volumefraction
                    Microstructure(potential_additives)=additive_id;
                else
                    [L,ncluster] = bwlabeln(potential_additives,18);
                    cluster_possibility=1:1:ncluster;
                    while current_additive_volumefraction <= target_volume_fraction
                        cluster_choice=randi(length(cluster_possibility));
                        Microstructure(L==cluster_possibility(cluster_choice))=additive_id;
                        current_additive_volumefraction =
sum(sum(sum(Microstructure==additive_id)))/ntot;
                        cluster_possibility(cluster_choice)=[];
                    end
                end
            else
                [L,ncluster] = bwlabeln(potential_additives,18);
                cluster_possibility=1:1:ncluster;
                volume_increment=0; % initalize
                while volume_increment <= (1-randomize_level)*n
                    cluster_choice=randi(length(cluster_possibility));
                    loc = L==cluster_possibility(cluster_choice);
                    Microstructure(loc)=additive_id;
                    volume_increment = volume_increment + sum(sum(sum( loc )));
                    cluster_possibility(cluster_choice)=[];
                end
            end
        end
    end
    distance_bnd2bnd(Microstructure==additive_id)=9e9;
    current_additive_volumefraction = sum(sum(sum(Microstructure==additive_id)))/ntot;
end
```

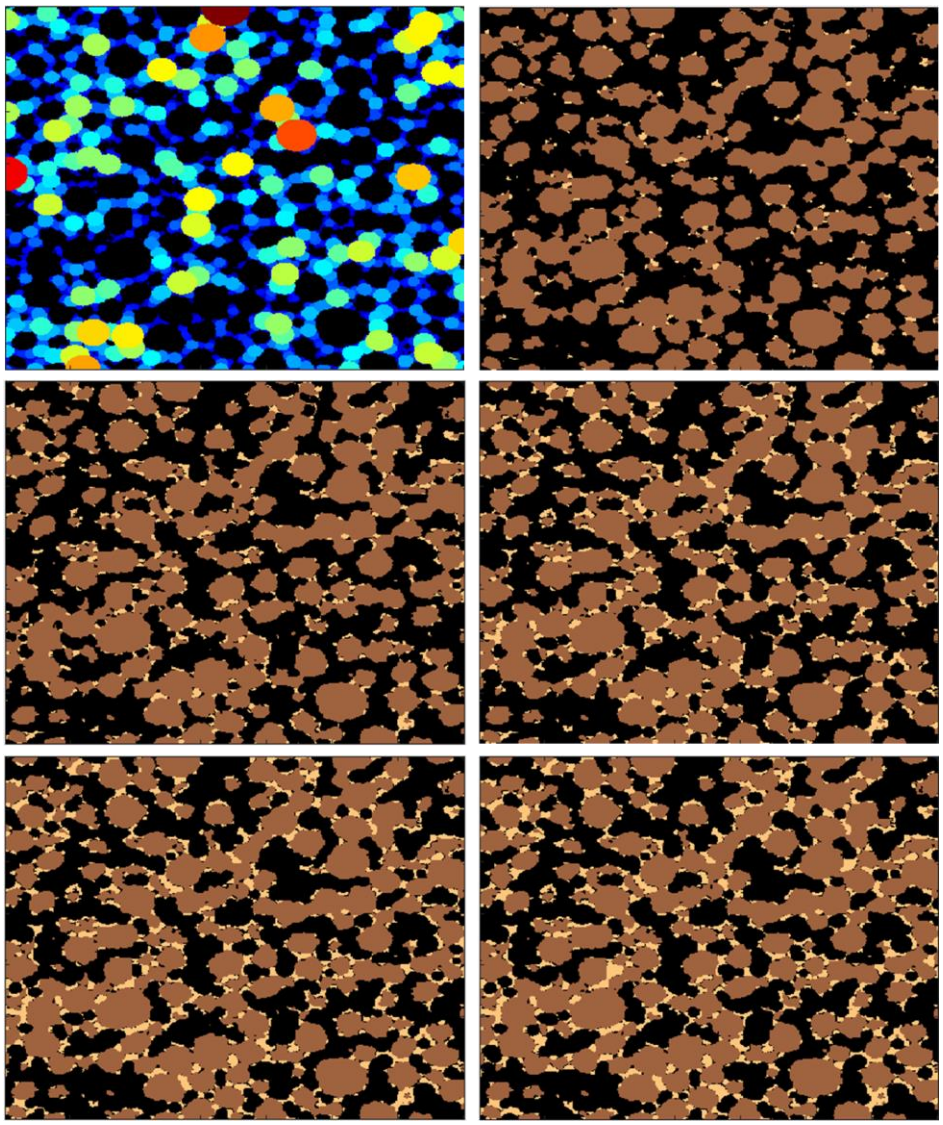


Figure IV-7a. (Top left) Sphere-size $c - PSD(x)$ of the pore domain. (Others) Pore voxels for which $c - PSD(x) < d$ are assigned to the additive phase iteratively.

Surface irregularities typically lead to small irrelevant additives and pore regions at the particle surface. These regions are corrected applying erosion steps sequentially, as described below:

- User can specify an erosion distance ('erosion depth in voxel length') to assign additive regions with a spherical size below this erosion distance to the background.

```
binary_phase=zeros(size(Microstructure)); % Initialization
binary_phase(Microstructure == additive_id) = 1;
[distance_bnd2bnd] = Function_particle_size_CPSD_Algorithm(binary_phase);
distance_bnd2bnd(distance_bnd2bnd==0)=9e9; % Remove background
Microstructure(distance_bnd2bnd<(erosiondepth*sqrt(3)+0.0001))=background_id;
```

- Because the initial generation has been restricted with a minimum distance ('minimum distance from boundary in voxel length'), the algorithm results in tiny pores between the solid and the additive phases. A similar approach is used to remove them:

```
binary_phase=zeros(size(Microstructure)); % Initialization
binary_phase(Microstructure == background_id) = 1;
[distance_bnd2bnd] = Function_particle_size_CPSD_Algorithm(binary_phase);
distance_bnd2bnd(distance_bnd2bnd==0)=9e9; % Remove background
Microstructure(distance_bnd2bnd<minimum_dist*sqrt(3)+0.0001)=additive_id;
```

- This last operation has likely added more additive than required. Additive connectivity is calculated, and clusters (from smaller to larger) are iteratively removed until the target volume fraction is reached.

```
delta = round(sum(sum(sum( Microstructure==additive_id  ))) -
numel(Microstructure)*target_volume_fraction);
erosion_cluster = true;
while delta>0
    app.TextArea.Value = {'Resorbing volume fraction','difference...'}; pause(0.01);
    binary_phase=zeros(size(Microstructure));
    if erosion_cluster==false
        % Random voxel erosion
        binary_phase(Microstructure==background_id) = 1;
        distance_map = bwdist(binary_phase,'chessboard');
        distance_map(Microstructure ~= additive_id)=2;
        idx = find(distance_map<2);
        if length(idx)<=delta
            Microstructure(idx)=background_id;
        else
            tmp = randi(length(idx),delta,1);
            Microstructure(idx(tmp))=background_id;
        end
    else
        % Cluster erosion
        binary_phase(Microstructure==additive_id) = 1;
        [L,~] = bwlabeln(binary_phase,18);
        % Determine cluster size
        [C,~,ic] = unique(L);
        a_counts = accumarray(ic,1);
        size_cluster = [C, a_counts];
        size_cluster = sortrows(size_cluster,2); % Sort in ascending order
        while size_cluster(1,2)<delta && delta>0
            Microstructure(L==size_cluster(1,1))=background_id; % Smallest cluster is removed
            delta = round(sum(sum(sum( Microstructure==additive_id  ))) -
numel(Microstructure)*target_volume_fraction);
            size_cluster(1,:)=[]; % Smallest cluster is removed from the list
        end
        erosion_cluster = false; % Switch to voxel erosion
    end
    delta = round(sum(sum(sum( Microstructure==additive_id  ))) -
numel(Microstructure)*target_volume_fraction);
end
```

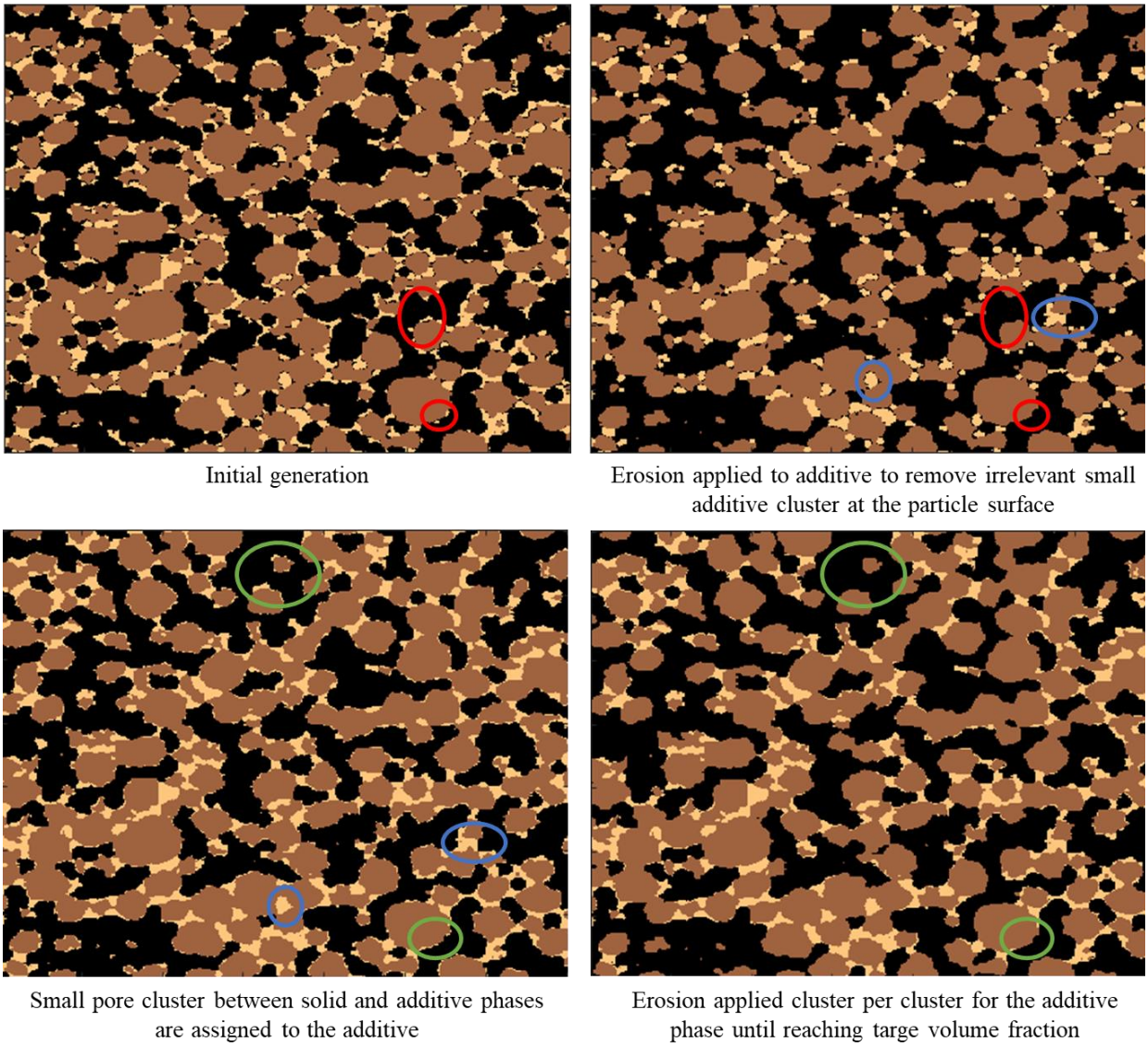


Figure IV-7b. (Top left) Initial generation, (Top right) after additive voxel erosion, (bottom left) after pore voxel erosion, and (bottom right) after additive cluster-per-cluster iterative erosion

Once all erosions steps are applied, most additives are located between neighbored particles, as intended.

b. Energy based approach

Method is presented in detailed in[8], and has been developed by Aashutosh Mistry (Chemical Sciences & Engineering Division, Argonne National Laboratory, Lemont, Illinois 60439, United States, ORCID: 0000 – 0002 – 4359 – 4975) and Partha P. Mukherjee (School of Mechanical Engineering, Purdue University, West Lafayette, Indiana 47907, United States, ORCID: 0000 – 0001 – 7900 – 7261).

Candidate voxels (candidate voxels have at least one solid neighbor and are within the background phase) assignment to the additive phase is based on an energy towards deposition on the pre-existing active material network or on the pre-deposited new additive phase. A user-defined morphology parameter ω (adimensional, [0,1]) is used to bias the energy towards deposition for the candidate voxel k, e_k, preferentially on existing active material ($\omega = 0$) or on pre-deposited additive phase ($\omega = 1$):

$$e_k = \frac{\omega}{6} N_{additive} + \frac{(1-\omega)}{6} N_{pre-existing\ active material} \qquad \text{[IV-3]}$$

With N_i the number of voxels adjacent to the candidate voxel that belong to the phase i. The two extremes values, 0 and 1, maximize the deposition energy for candidate voxels which are surrounded, respectively, by the pre-existing solid network or by the previously deposited additive phase. Therefore, a near 0 value will result in generating a thin layer of additive phase at the particle surface, while a near 1 value will result in generating a dendritic-like additive phase. The local energy deposition is normalized by dividing by 6 as $N_{additive} + N_{pre-existing\ solid\ network} \leq 6$ due to the cuboid voxel representation of the microstructure. The energy deposition cumulative function is then calculated, considering only candidate voxel with a non-zero energy deposition:

$$E(K) = \sum_{k=1}^{K} e_k \qquad \text{[IV-4]}$$

The cumulative function is used to select candidate voxels, preferentially those with a high energy deposition, that will be assigned to the additive phase as explained in the figure below.

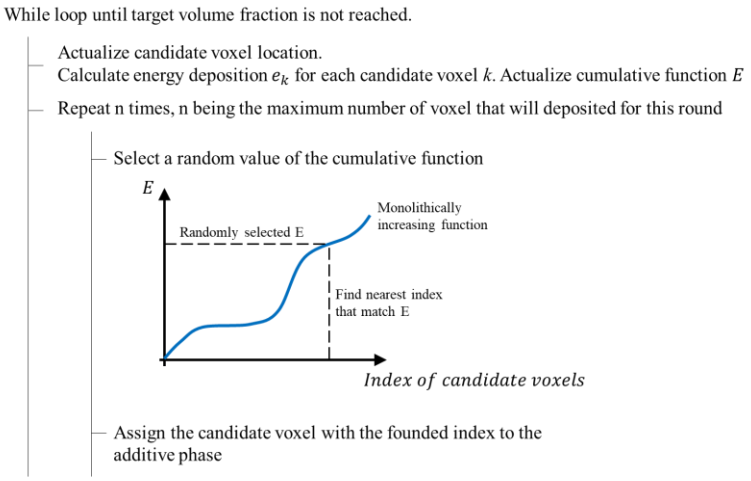


Figure IV-7c Candidate voxels are assigned to the additive phase based on the energy cumulative function according to a stochastic (i.e., a random) process.

The key point being that candidate voxels with low energy deposition corresponds to a flat region of the cumulative function. Therefore, these candidate voxels have a lower probability to be selected compared with voxels with higher energy deposition that correspond to the sharp variation of the curve. Note that this property is true only for a non-random energy deposition among the candidate voxels (empirically verified for microstructures). The average energy deposition among the selected voxels is then higher than the average energy deposition among the candidate voxels. The stochastic process allows for some candidate voxels to be selected even though they have a low energy deposition. This choice is motivated from the fact that low energy events are possible but with a lower probability and then should not be discarded entirely from the deposition process.

The initial candidate voxels are typically not enough to reach the additive phase volume fraction target. Therefore, the deposition process is iterative (the whole loop of figure IV-7c), with each new round of deposition starting with updating the energy deposition since the microstructure has been modified with the voxels assigned to the additive phase from the previous iteration. Each round of deposition adds a maximum fraction of the additive phase volume target volume fraction (by default 1%). The user can select a larger value, resulting in less iterations, while a smaller value will require more iterations but will provide a more progressive growth of the additive phase as the energy map will be actualized more often.

The deposition process is summarized in the figure IV-7d.

Articles based on this code:

- Mistry, Smith & Mukherjee (2018) doi: 10.1021/acsami.7b17771
- Mistry, Smith & Mukherjee (2018) doi: 10.1021/acsami.8b08993
- Usseglio-Viretta *et al.* (2018) doi: 10.1149/2.0731814jes
- Trembacki *et al.* (2018) doi: 10.1149/2.0981813jes
- Mistry & Mukherjee (2019) doi: 10.1039/C8CP05109G
- Mistry, Verma & Mukherjee (2019) doi: 10.1021/acsami.9b05468
- Mistry, Smith & Mukherjee (2020) doi: 10.1021/acsami.9b23155
- Mistry *et al.* (2020) doi: 10.1149/1945-7111/ab8fd7

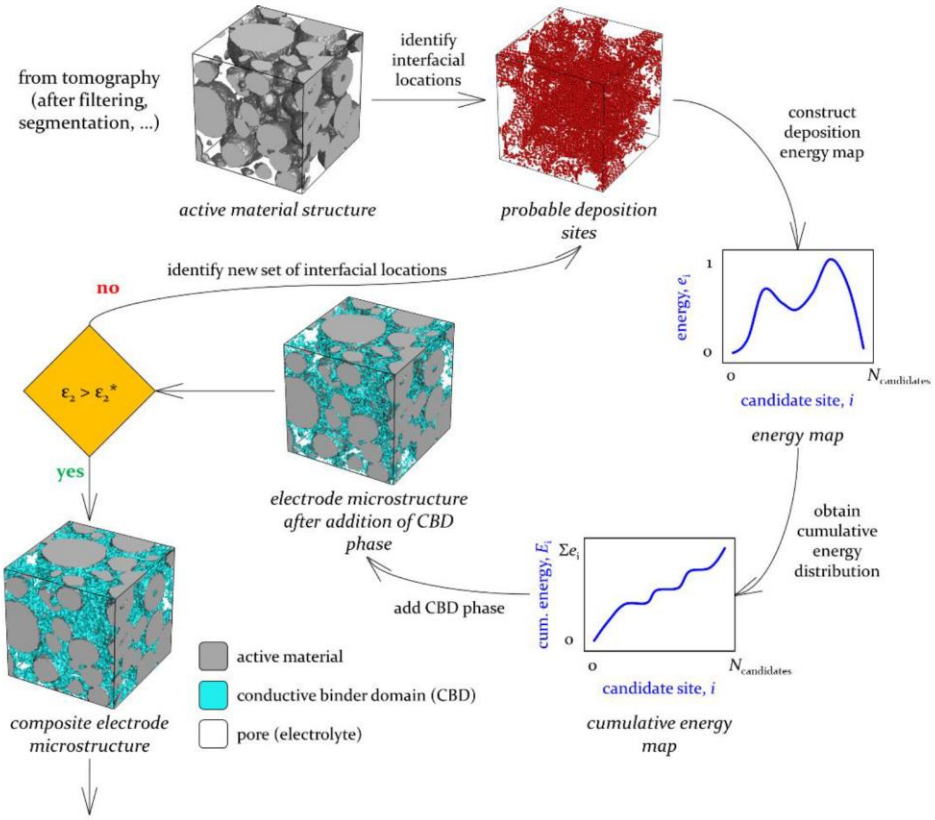


Figure IV-7d A flow chart explaining the flow of logic and data between different routines to add CBD phase in an active material skeleton[8,21].

c. How to use

Click on the 'microstructure generation' button from the main menu GUI, and then click on 'Additives' to load the additive generation module. In the first tab ('Load volume and volume fraction'), load a tif segmented volume, select the background phase within which the additive phase will be generated (by default the first phase), enter the volume fraction of the additive phase, and lastly enter the code to assign for voxels that will belong to the additive phase. The target volume fraction is deduced from the dense volume fraction and the nanoporosity:

$$\varepsilon_{target} = \varepsilon_{dense} \times \frac{1}{1 - \varepsilon_{nanoporosity}} \qquad \text{[IV-5]}$$

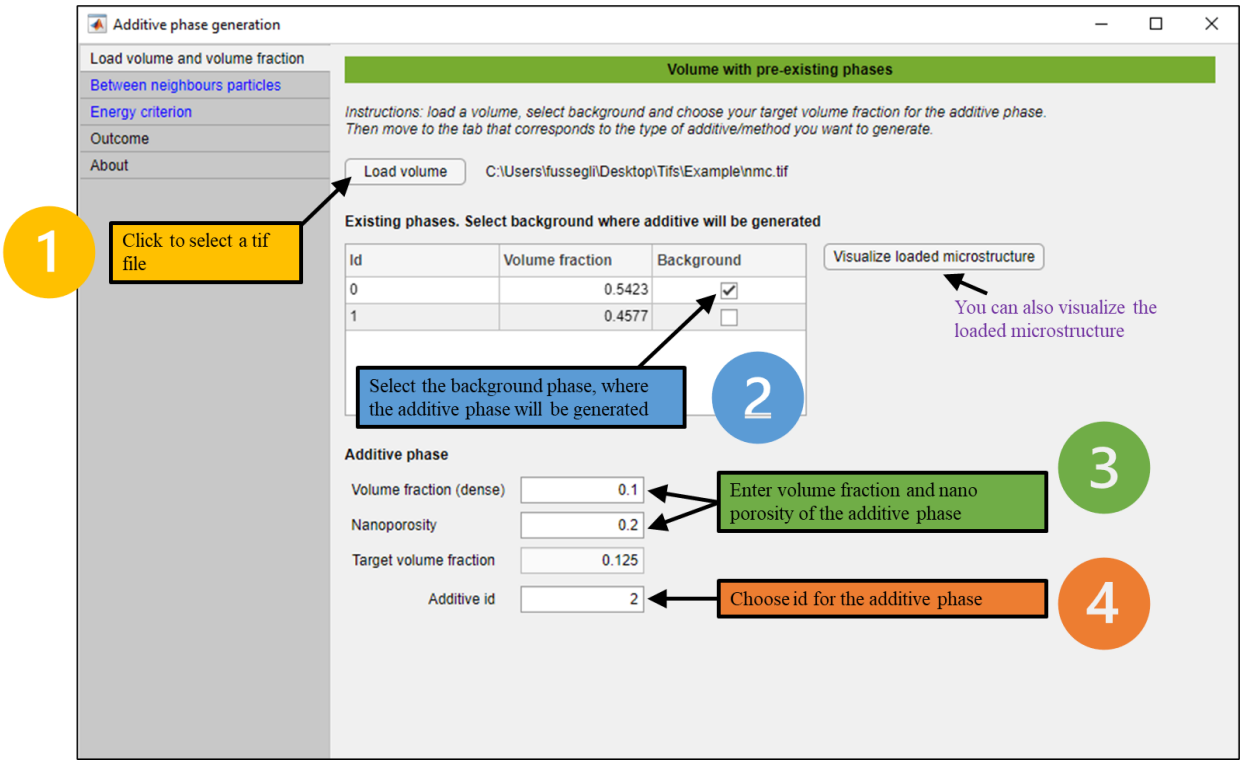


Figure IV-8a. Load volume tab

Depending on the generation method you want to use, go to the second tab ('Between neighbours particles') or the third tab ('Energy criterion'). Each tab allows you to modify the method's parameters (within their bounds) from their default values. Parameters are described in the previous sections.

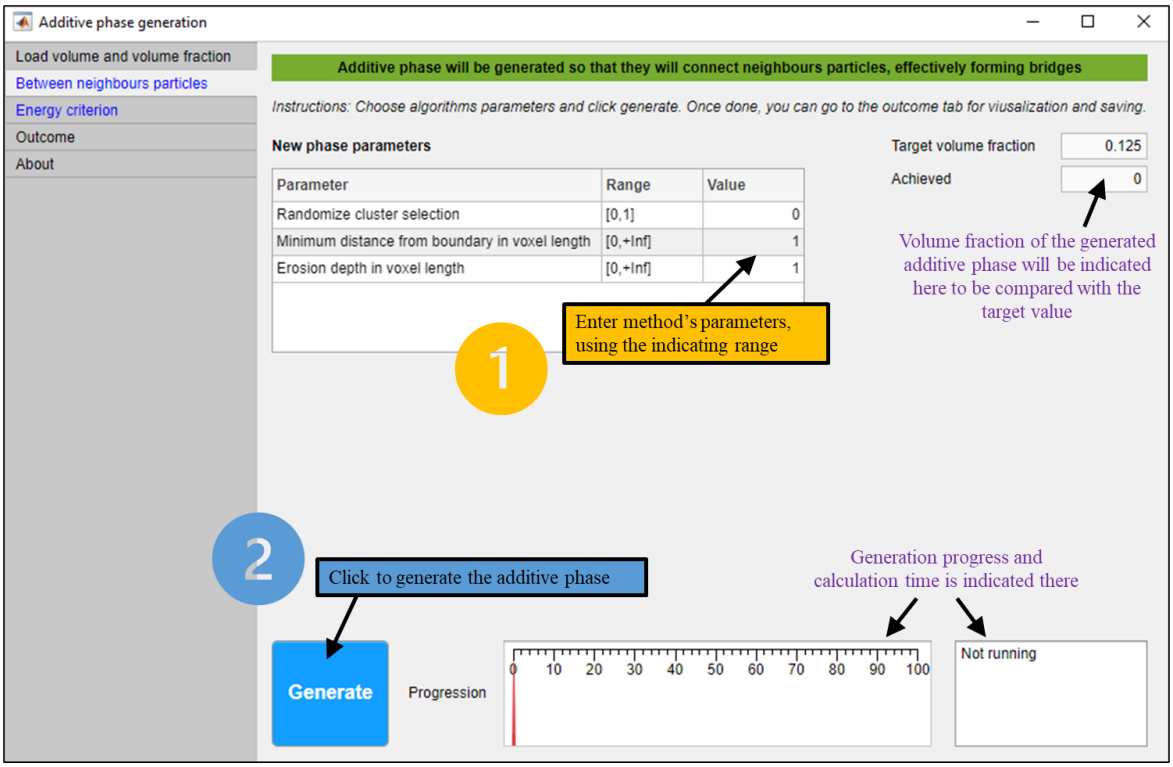


Figure IV-8b. Option to generate additive phase between neighbours particles.

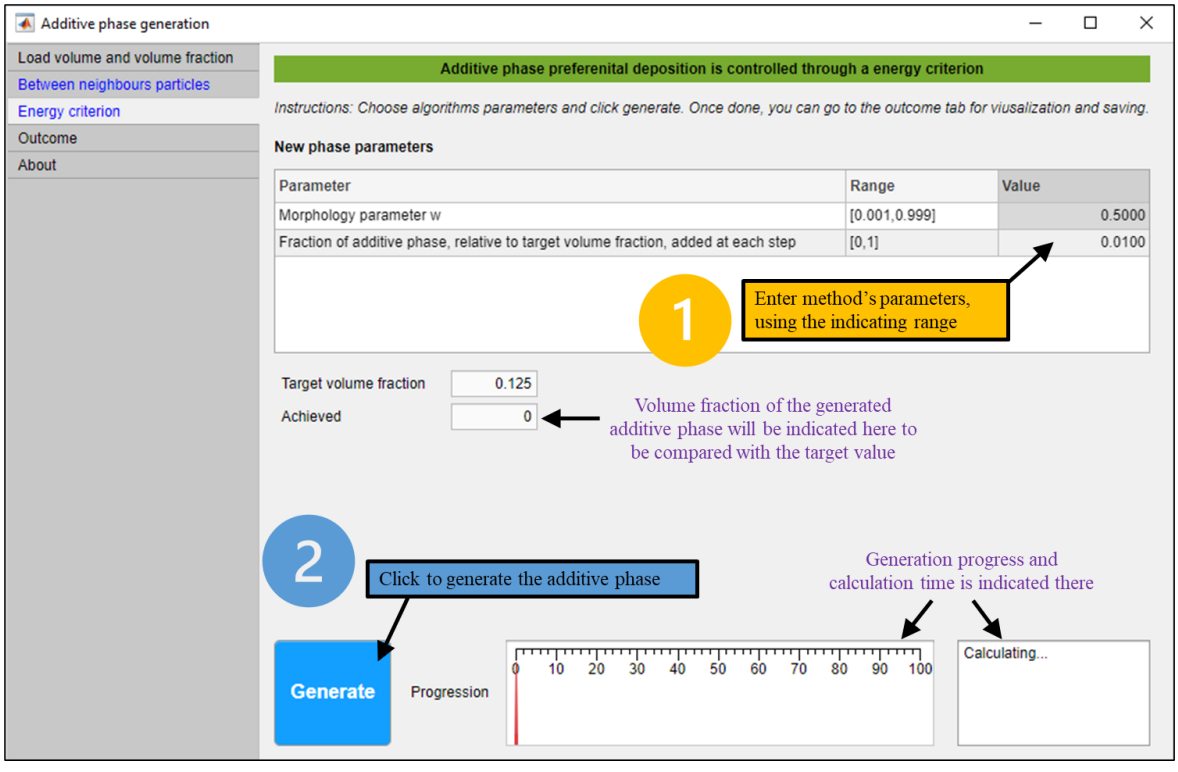


Figure IV-8c. Option to generate additive phase using energy criterion.

The ‘Outcome’ tab allows the user to verify the generation process, both globally through the volume fractions, but also locally by visualizing the modified microstructure. It’s also the place where you can save the modified microstructure.

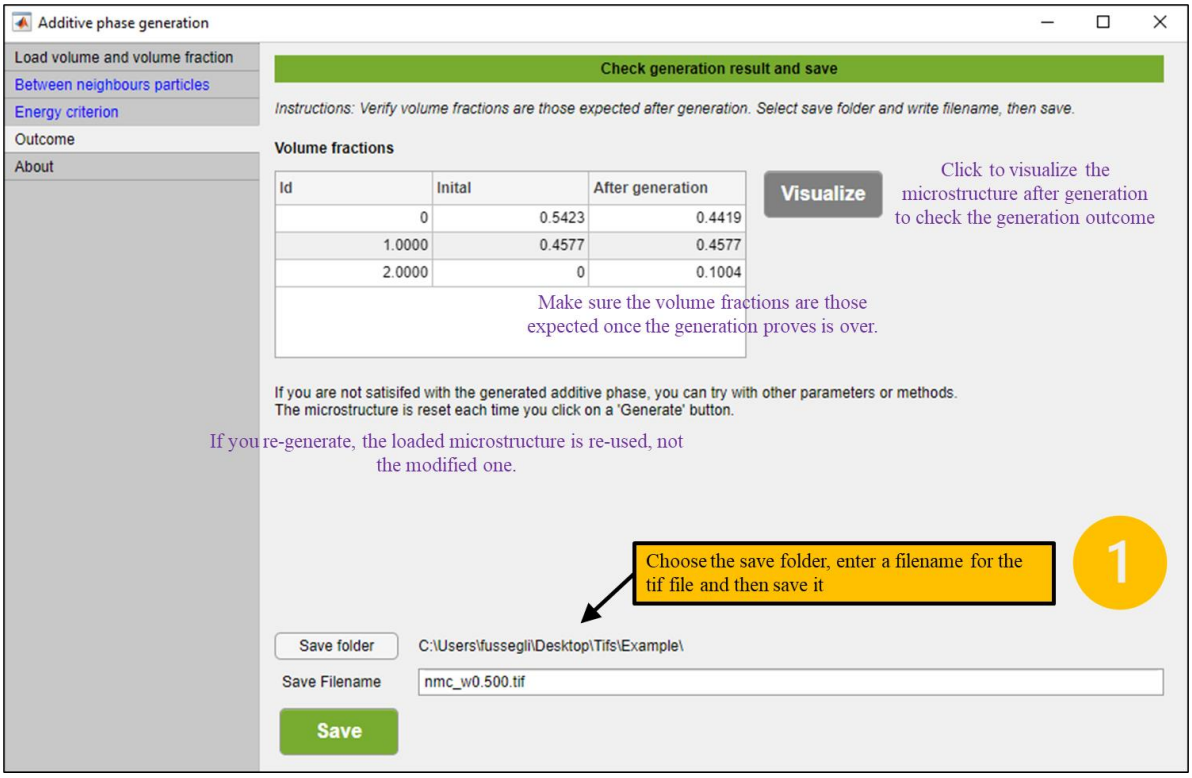


Figure IV-8d. Verify generation outcome and save modified microstructure.

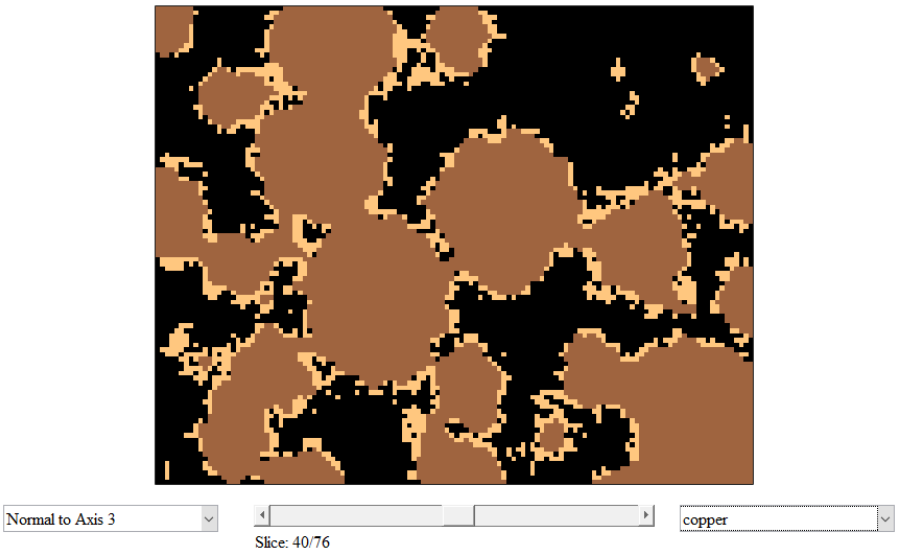


Figure IV-8e. Interactive figure provided with the ‘Visualize’ button (energy criterion method, $\omega = 0.5$).

The last tab provides general information about the module, and how to quote results produced with it.

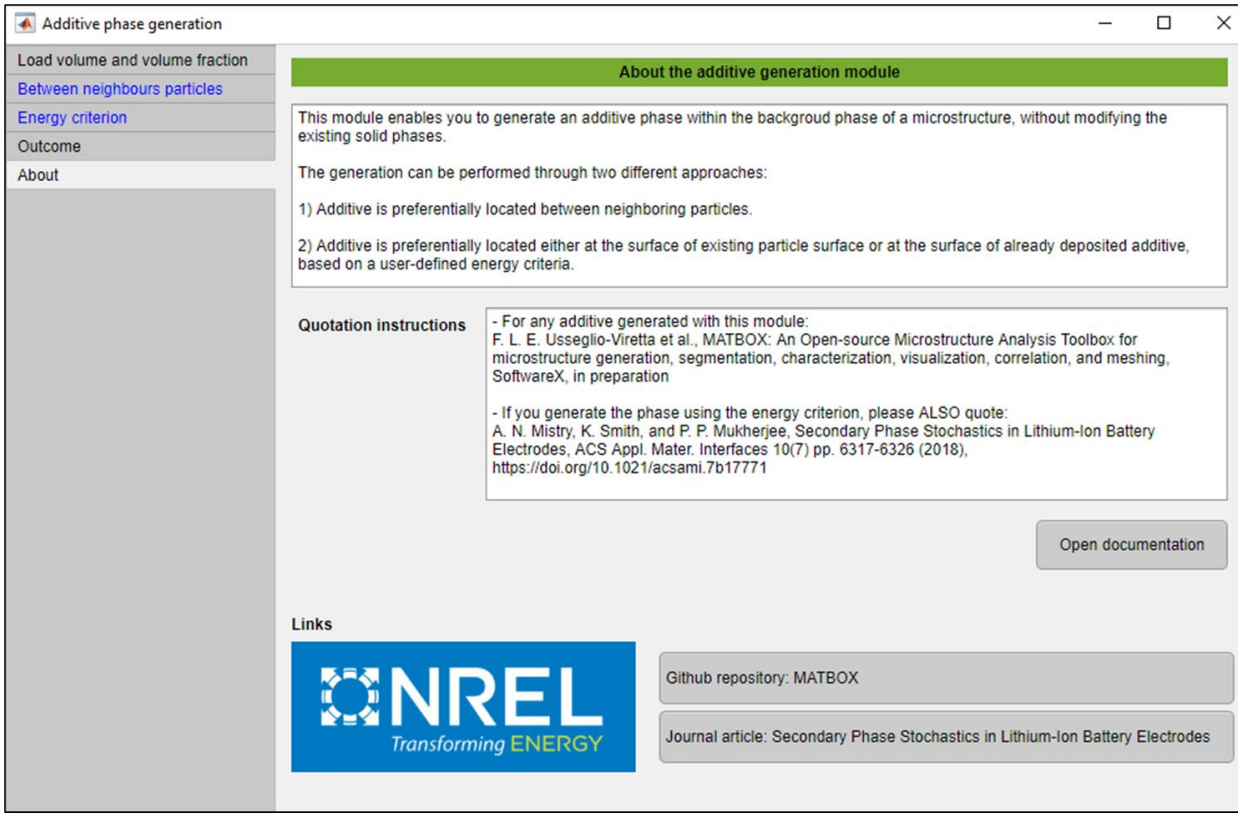


Figure IV-8f. General information about the additive phase module, and quotation instructions.

d. Examples of generated additives

Example files are available in the repository at \Data_example\From computed tomography with additives numerically generated\.

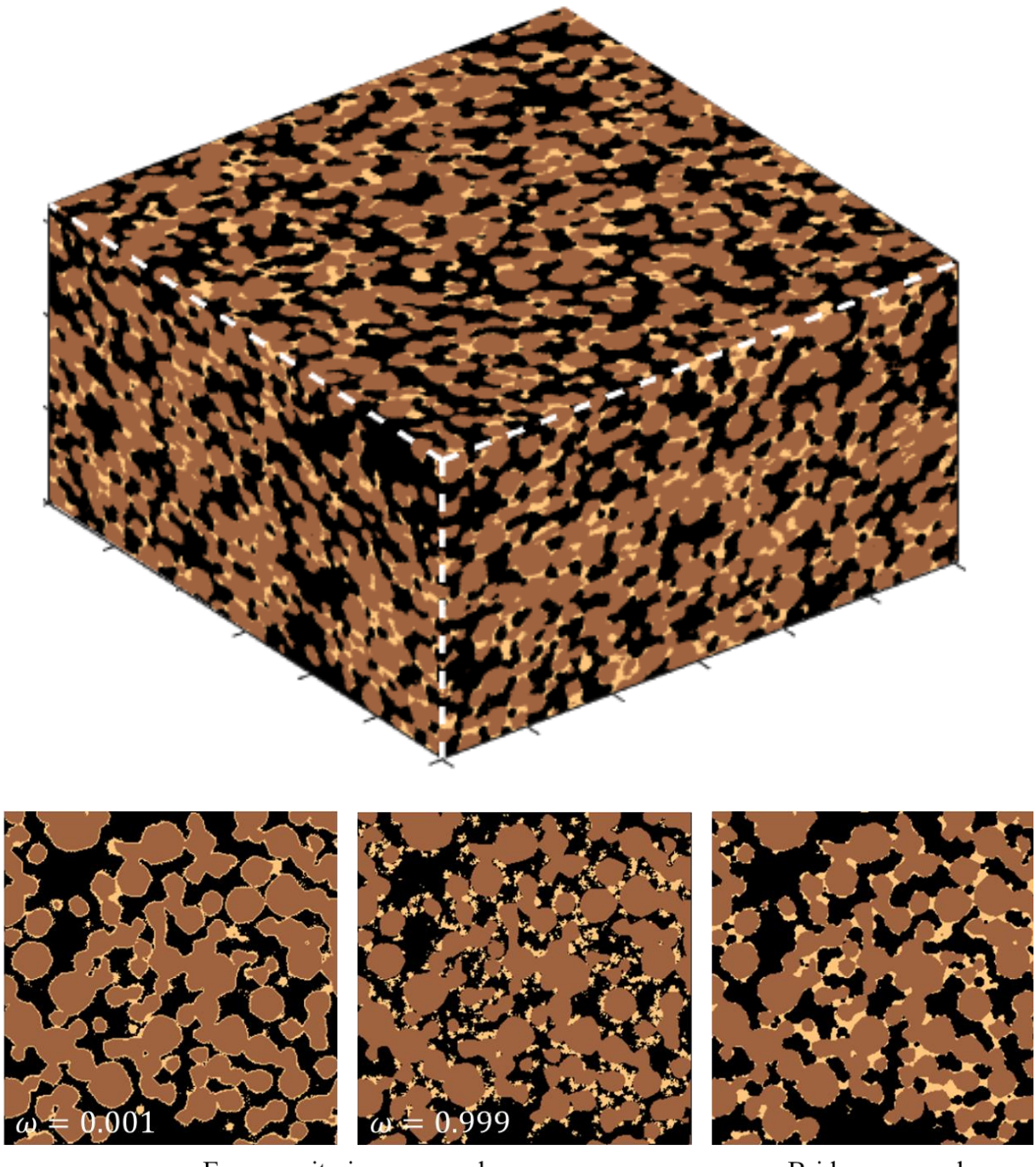


Figure IV-9. Additive phase (0.1 volume fraction) added to an existing microstructure. Images have been generated using the visualization module (cf. §VII). (Top) 3D view of microstructure with additive generated with the bridge approach, and (bottom row) 2D views comparing the different approaches/parameters.

V. FILTERING AND SEGMENTATION

1. Module purpose

Image filtering[22,23] and segmentation[24,25] have their own field of research and are abundantly covered in the literature. Other open-source software, such as ImageJ, have extensive library of numerical methods to filter and segment 2D and 3D images. This module does not have the ambition to provide an exhaustive list of methods. Instead, it focuses on delivering a user-friendly experience to improve productivity of various pre-characterization tasks, that can be time-consuming and frustrating with less intuitive tools. While methods currently available in this module are quite simple, they nevertheless proved to be enough to segment NMC and graphite electrode volumes from the NREL microstructure library[13]. For more advanced methods, the user is invited to rely on ImageJ. Module instructions are provided in the first tab.

2. Importing a volume, saving progress, and keeping tracks of change

Most tabs are empty until you load a volume. To do so, click to Volume/Load volume in the menu selection. **Double/32bits format tiff import is not supported, so please convert in 16 bits before importing.** Once the volume has been loaded, basic information is displayed in the first tab. In the second tab named 'Save/display options' you can select the save folder, if you want to save figures generated during various tasks (e.g. grey-level histogram), and the figure font name. Note that the 'current file name' uses the same name of the loaded file, but concatenated with '_step_X.tif', with X being initially equal to 1 as the step 1 corresponds to the volume loading. Each time a task *that modifies the microstructure volume* (e.g., cropping) is performed the step number is incremented and the history log (see last tab, named 'History') is updated. If you click to Volume/Save volume/Default location, the current state of the volume is saved in the save folder using the 'current file name'. Furthermore, when a volume is saved, the history log is saved as well in an excel file in the same folder and with the same name (except for the file extension). This avoid overwriting files and help keeping tracks of changes. This is particularly valuable to remind how a volume has been segmented months after having done it. You can also save volumes using Volume/Save volume/Custom location to manually select a folder and name. It is sometimes interesting to save a volume at different steps to document the various steps required for its segmentation.

Tasks that change the volume data (e.g. cropping, filtering etc.) require the user to click on the blue button 'Do' to realize them. Once done, the 'Do' button is greyed out while the orange 'Undo' button and the green 'Save' button are enabled. If not in your current tab, you can always visualize the modified volume in the 'Region of Interest/View' tab. If you are satisfied with the change, please click on the 'Save' button. Be careful as you cannot undone changes after clicking on the save button. History log and 'current file name' are updated only after clicking on the save button. If you are not satisfied with the change, please click on the 'Undo' button. Note that you

cannot perform any other changing tasks until you have either clicked on the ‘Save’ or ‘Undo’ buttons as other ‘Do’ buttons are greyed too.

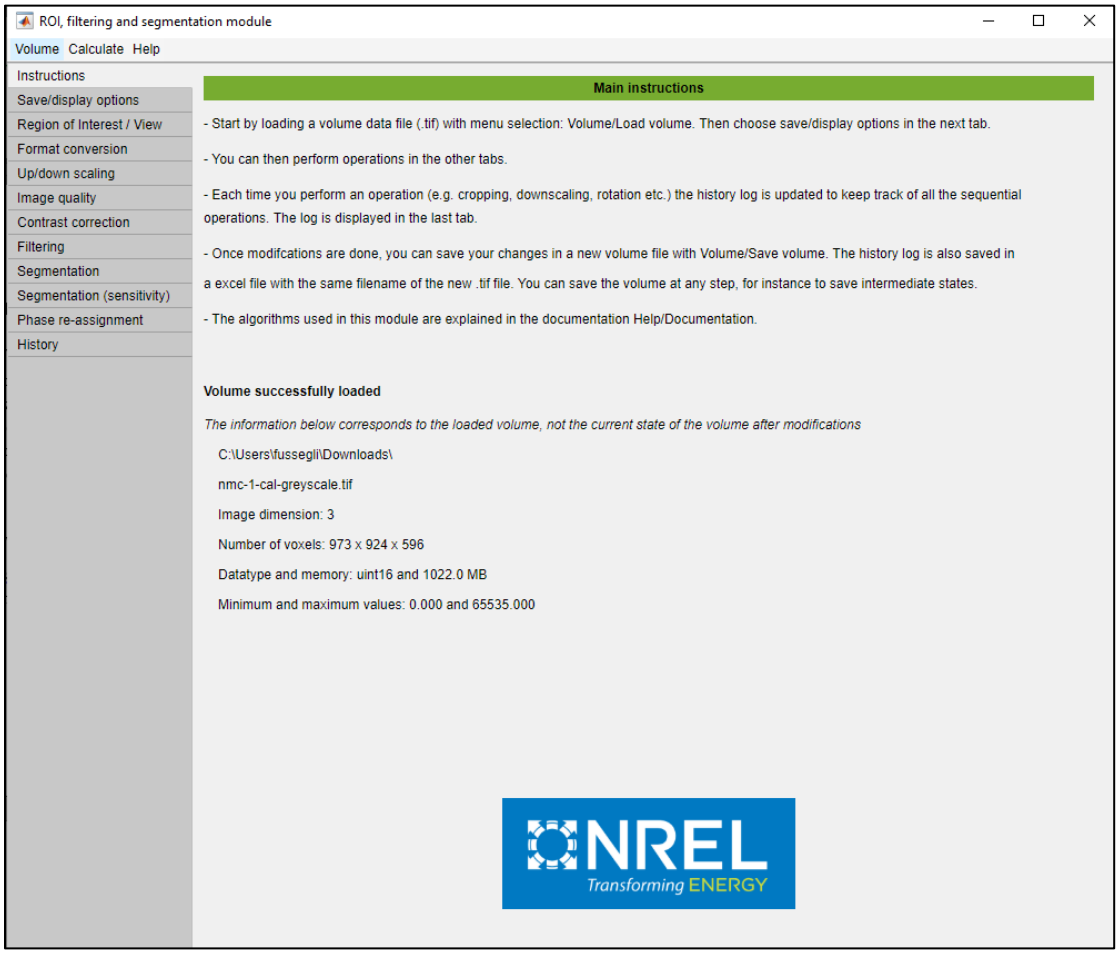


Figure V-1. First tab after a volume has been successfully loaded

Each operations performed on the volume are recorded in the table below.

Step	Operations	Parameters	Elapsed time
-	Start date	Tuesday, 12:55:13 -0600, June 2, 2020	n/a
-	User name	fussegli	n/a
-	Computer name	FUSSEGLI-34154S	n/a
-	Operating system	Windows_NT	n/a
-	MATLAB version	9.7.0.1261785 (R2019b) Update 3	n/a
1	Loading file	C:\Users\fussegli\Downloads\nmc-1-cal-greyscale.tif	13.6s
2	Crop volume	Axe 1: 172-803 Axe 2: 163-794 Axe 3: 1-596	0.3s
3	Swap axis 1 with axis 3	No parameters	4.1s
4	Crop volume	Axe 1: 70-550 Axe 2: 1-632 Axe 3: 1-632	0.7s
5	Rotation volume	Normal to axe 2: 6.5 degrees	3.2s
6	Rotation volume	Normal to axe 3: -4.5 degrees	3.5s

Figure V-2. Example of an history log after few basic steps

3. Region of interest and microstructure visualization

Visualization and basic region of interest (ROI) operations are accessible in the 'Region of Interest / View' tab. Visualization options are located at the bottom right. 'Save current 2D view' will simply save the current visualization with the title indicating the slice number. Note that figures are saved only if the option is checked in the second tab. 'Plot and save 3D view (orthogonal slices)' shows a good overview of the microstructure communicable through a unique picture. Plotting the orthogonal slices is done using the MATLAB built-in function *slice*, which is very RAM expensive, and may not be appropriate for very large volume. '3D view' calls the MATLAB built-in function *volshow*, mostly valuable for segmented volume. Since figure names are concatenated with the step number, saving during the imaging processing is a good way to keep a visual track of the various modifications applied to the volume.

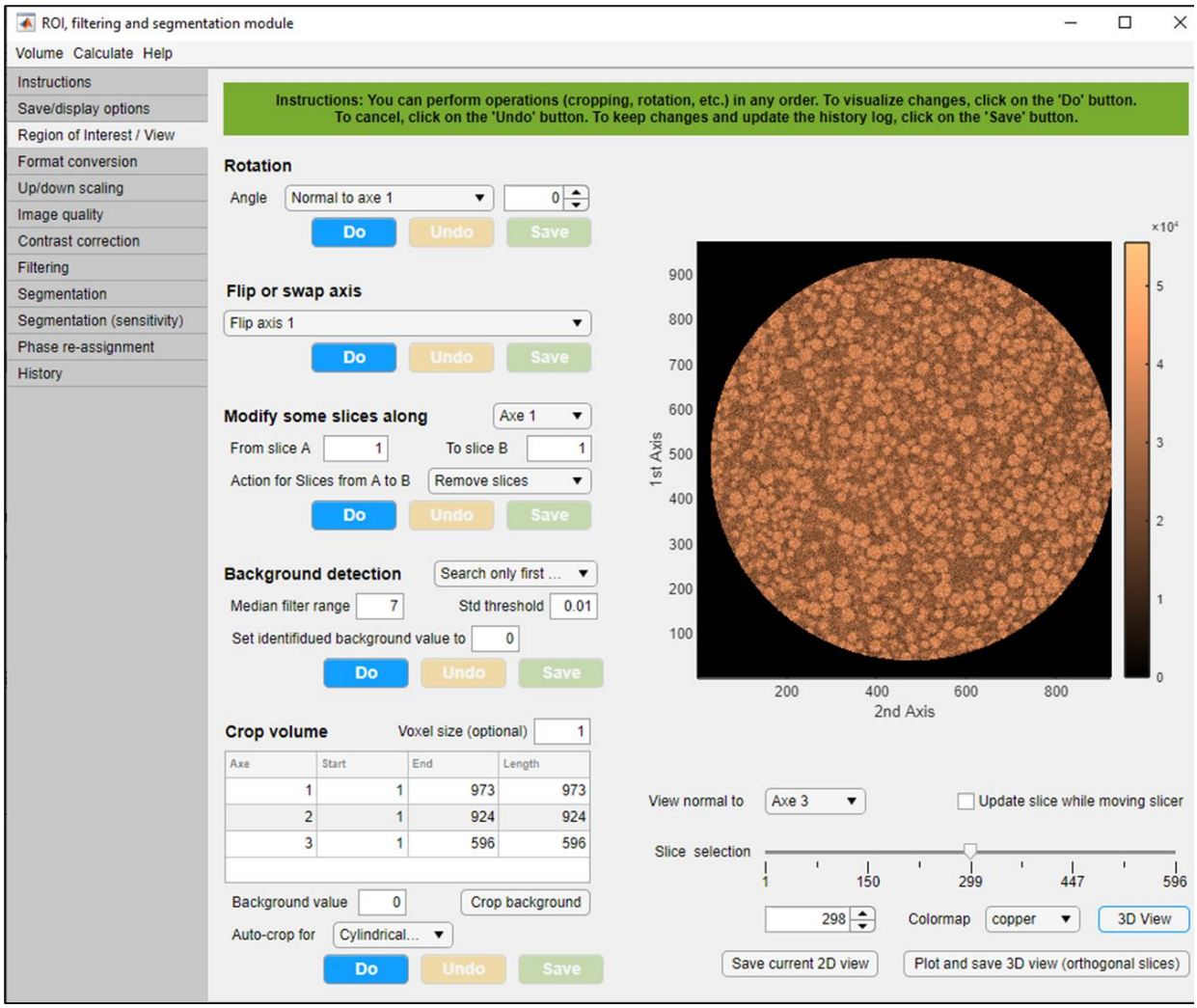


Figure V-3. Region of interest selection and microstructure view

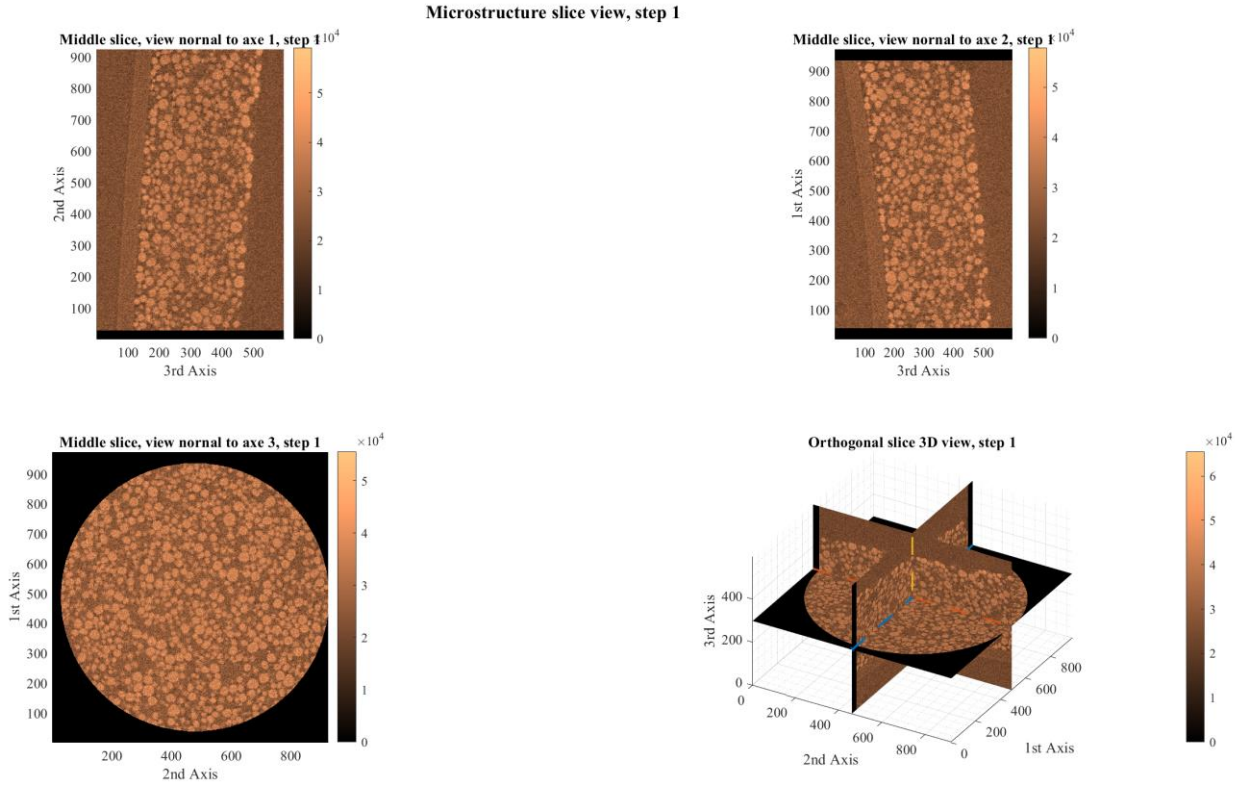


Figure V-4. Figure generated with 'Plot and save 3D view (orthogonal slices)'.

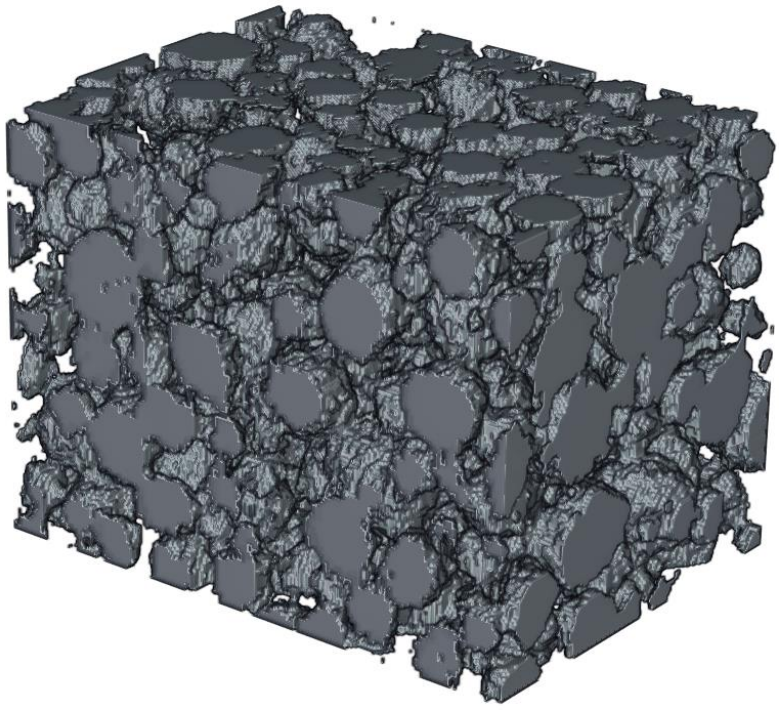


Figure V-5. Figure generated with '3D view' for a segmented volume.

You can perform several operations in this tab:

- *Cropping*. You can enter directly cropping bounds in the table of the 'Crop volume' section. Bounds are represented with red dashed line to help the user choosing them. Voxel size is purely optional and is used to multiply the domain's size (value is reported in the length column), to provide the physical dimension of the domain. If the background value is known (in this example: 0), an auto-cropping can be realized by first selecting the auto-crop option from the menu selection ('Cylindrical FOV' or 'After rotation') and then on clicking on the 'Crop background' button. Auto-cropping requires the background to be consistent along all slices. To finalize the cropping, you must click on the 'Do' button.

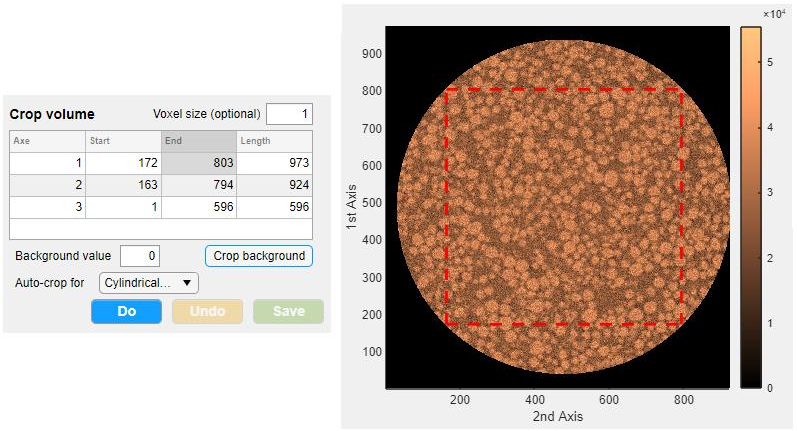


Figure V-6a. Cropping bounding automatically detected using the 'cylindrical FOV' menu selection. The third axis must be normal with the field of view as in this figure (use swapping axis otherwise to make it so). First slice is used to detect the background. The cylindrical FOV is typical for Computed tomography experiments.

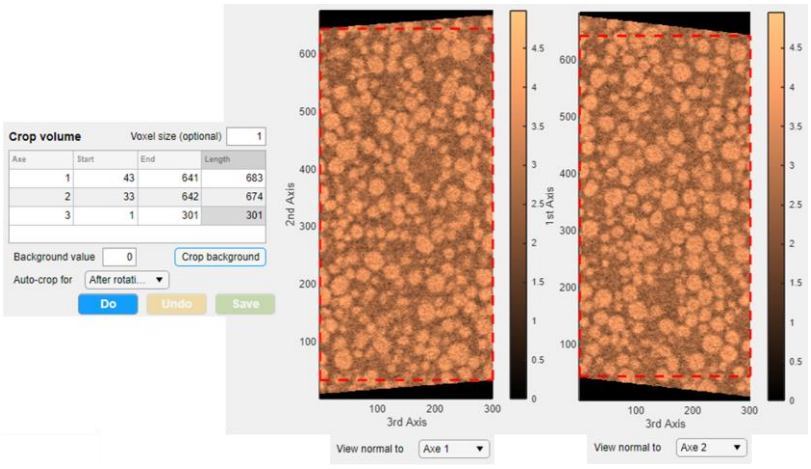


Figure V-6b. Cropping bounding automatically detected using the 'After rotation' menu selection. Only bounds of axis 1 and 2 will be found, so you may have to swap axis to set your volume orientation as in this image, crop axis 3, and then do the auto-cropping.

- *Flip or swap axis*. 'Flip' returns an axis, while 'swap' switches two axis. This is valuable to orient a microstructure according to your axis convention. This is particularly relevant when analyzing several volumes, for which imaging experiments may have different orientations. For instance, for electrode battery one can systematically uses the third axis for the electrode thickness.

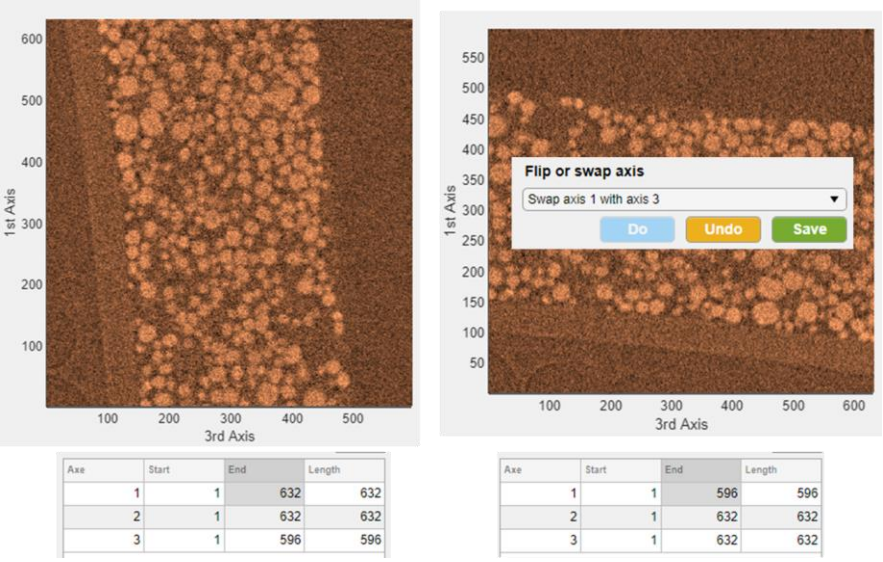


Figure V-7a. (Left) Before and (right) after swapping axis 1 with axis 3.

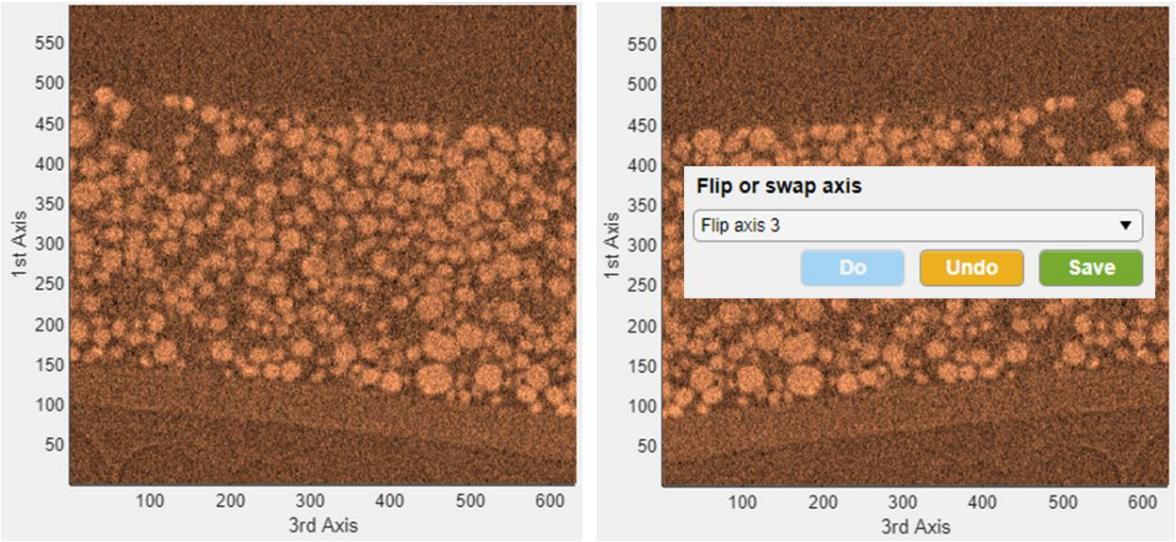
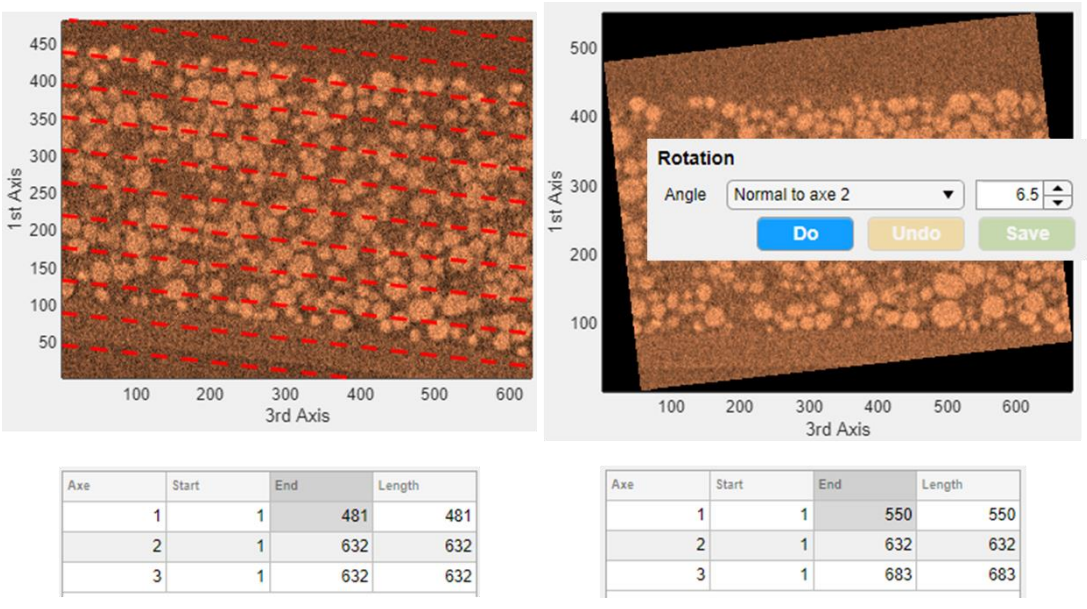


Figure V-7b. (Left) Before and (right) after flipping axis 3.

- *Rotation*. Imaging experiments usually try to keep the material orientation, if any. For electrode battery, it means aligning one axis with the electrode thickness, that is normal to the current collector plane. However, alignment is often slightly off and you may have to correct it especially if you are interesting with calculating oriented properties such as tortuosity factors.

To do so, you can use the 'Rotation' section of the tab. First select the axis rotation, and then choose your angle rotation. Red dashed lines are draw automatically to indicate the angle rotation (if you do not see them, you are using the wrong angle axis rotation). Because rotation uses a lot of RAM, a good practice consists in cropping your volume first, and then do the rotation. You may also swap axis before doing the rotation for ease. The MATLAB documentation provides details about geometric transformations (see https://www.mathworks.com/help/images/matrix-representation-of-geometric-transformations.html). The pseudocode below explains how rotations are applied.

```
angle_rad = deg2rad(angle_deg); % Angle in radians
-For 3D volume:
transformation matrix that rotates the image around the 1st axis
t = [cos(angle_rad) 0      -sin(angle_rad)   0
                  0 1                    0   0
     sin(angle_rad) 0       cos(angle_rad)   0
                  0 0                    0   1];
transformation matrix that rotates the image around the 2nd axis
t = [1                0              0 0
     0  cos(-angle_rad) sin(-angle_rad) 0
     0 -sin(-angle_rad) cos(-angle_rad) 0
     0                0              0 1];
Transformation matrix that rotates the image around the 3rd axis
t = [ cos(angle_rad) sin(angle_rad) 0 0
     -sin(angle_rad) cos(angle_rad) 0 0
                   0              0 1 0
                   0              0 0 1];
-For 2D image:
Transformation matrix that rotates the image around the 3rd axis
t = [ cos(angle_rad) sin(angle_rad) 0
     -sin(angle_rad) cos(angle_rad) 0
                   0              0 1];
Then pass the matrix to the affine3d object constructor.
t_form = affine3d(t);
Microstructure = imwarp(Microstructure,t_form); Apply geometric transformation to image
```



Axe	Start	End	Length
1	1	481	481
2	1	632	632
3	1	632	632

Axe	Start	End	Length
1	1	550	550
2	1	632	632
3	1	683	683

Figure V-8. (Left) Before and (right) after a 6.5-degree rotation normal to axis 2 used to align the volume. Note that dimensions have changed.

Once the rotation done, you may have to crop the useless background added during the geometric transformation. You can do it manually or use the auto-crop feature presented in the 'cropping' section.

- *Background detection*. The background is not always clearly identified. Although, background is often much more uniform compared to the microstructure combined pore and solid material phases. In the 'Background detection' section, you can set parameters (standard deviation threshold and mean filter range) used to identify background based on this assumption. First, standard deviation is averaged using a 2D mean filter: large background regions only contain background, while large microstructure regions contain both pore and solid phase(s) with significant different grey level values. If the grey level value dispersion within the background is inferior to the grey level value dispersion within the microstructure, then the average standard deviation will be lower within the background and a threshold can identify it. The identification can be done only for the first slice of the 3rd axis and then used for the others slice to speed up the calculation (relevant if the background location is identical among all slices), or slice per slice. Below is a pseudocode of the background identification process:

```
sdImage = stdfilt(Microstructure(:,:,k)); % Calculate standard deviation of slice k
sdImage_filtered = medfilt2(sdImage,[Median_filter_range Median_filter_range]); % Apply moving
averge filter on square regions of size Median_filter_range* Median_filter_range
index_background=find(sdImage_filtered<Std_threshold); % Threshold it and find coordinates
Microstructure(index_background,k)=Set_identified_background_value;
```

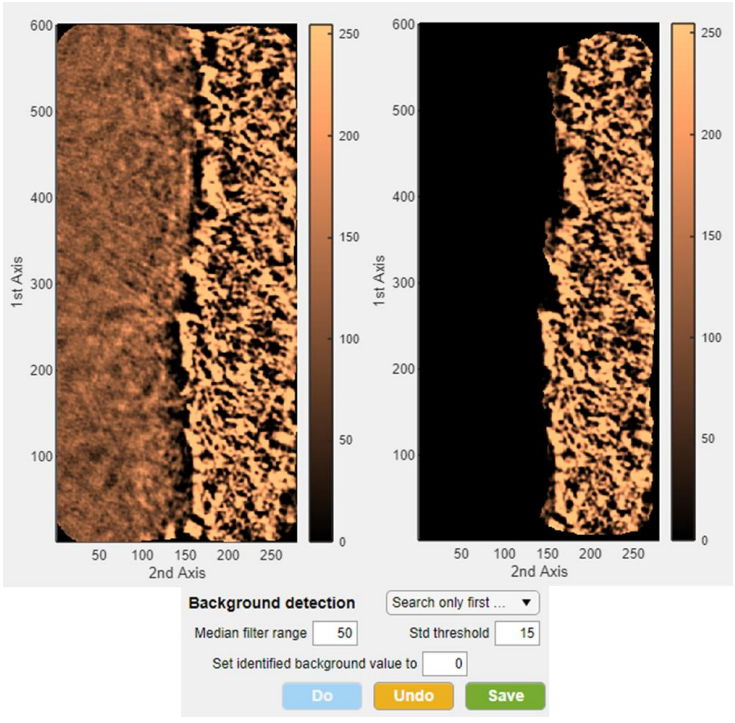


Figure V-9. (Left) Before and (right) after background identification.

- *Remove/modify slices*. This section is useful if some intermediate slides are corrupted or abnormally blurred. You can remove these slices or replace with interpolated values.

Interpolating is recommended as removing slices will create a discontinuity within the electrode. See section ‘Modify some slices along’.

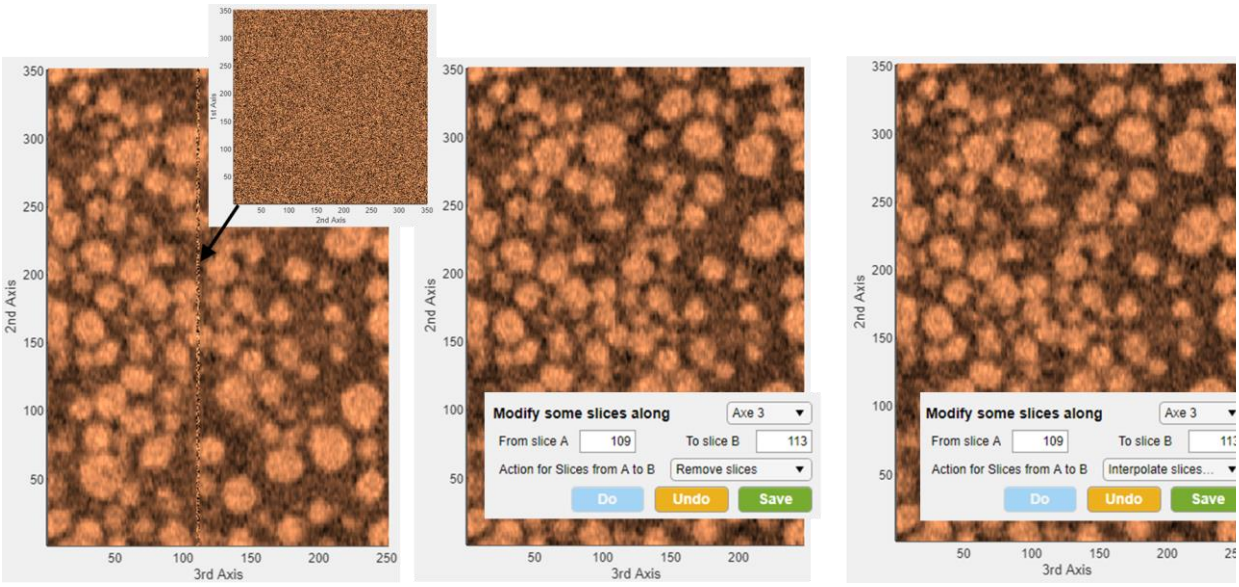


Figure V-10. (Left) Before, (center) after removing slices, and (right) after interpolating slices.

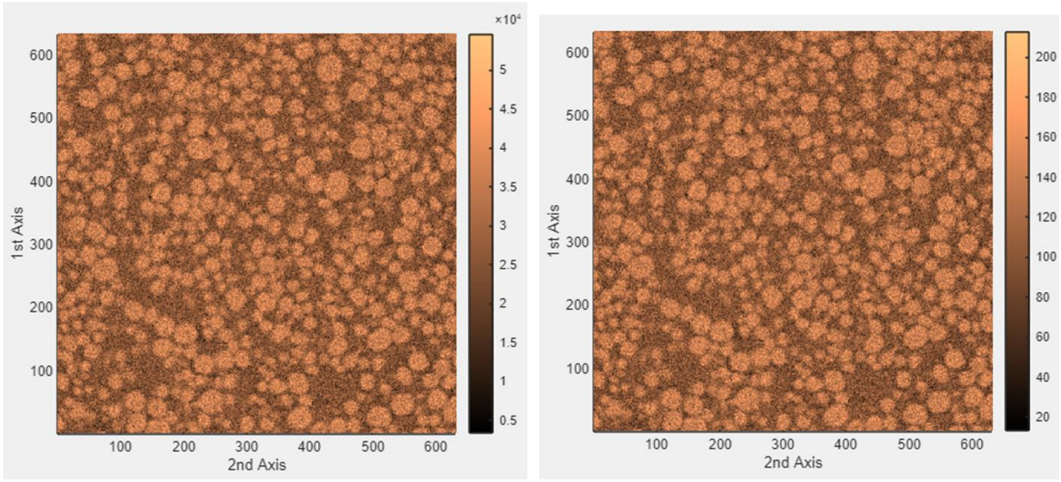
4. Modifying image format

Microstructure data can be saved in various formats. For segmented volume, the format of choice is 8-bit unsigned integer arrays, as the number of different phases is typically inferior to 256 and positive. For grey-level volumes, the choice depends on the image quality and noise. Typically, it is recommended to use a format that provides just enough granularity to perform the segmentation. For instance, if moving the segmentation threshold of 1 grey-level, from the reference threshold, changes too much the volume fractions, it indicates the image has been compressed too much. On the other side, using a too large data range uses more storage space, and algorithms that iterate over grey level values will be slower. Therefore, finding a compromise is required. You can quantify the segmentation sensitivity with the threshold in the ‘Segmentation (sensitivity)’ tab (cf. §V-9b), if required. You can also compare the shape of the grey-level histogram (cf. §V-6) before/after format change, as a change would reveal a too high compression. An indicator of an over or under sized data format is the utilization ratio, which is the number of grey-level value used over the data range length. In the figure V-11, the 16-bit unsigned image uses ~73% of the 65536 possible values, and ~89.5% of the 256 possible values once converted in an 8-bit unsigned format. After compression, the image looks nearly identical suggesting the 8-bit unsigned format is enough. You can convert image format in the tab ‘Format conversion’, with the following choices:

Choice	**MATLAB command**

8-bit unsigned integer arrays	Microstructure =uint8(Microstructure)
8-bit unsigned integer arrays (rescale)	Microstructure =im2uint8(Microstructure)
16-bit unsigned integer arrays	Microstructure = uint16(Microstructure)
16-bit unsigned integer arrays (rescale)	Microstructure = im2uint16(Microstructure)
32-bit unsigned integer arrays	Microstructure = uint32(Microstructure
64-bit unsigned integer arrays	Microstructure = uint64(Microstructure
double precision array (rescale)	Microstructure = im2double(Microstructure

Please refer to the MATLAB documentation for the operators used. Always check the microstructure in the 'Region of Interest / View' tab before clicking the 'Save' button to make sure you did not degrade the image. Please note that can you also specify a custom data range (without modifying the data format) in the 'contrast correction' tab (cf. §V-7).



	Current	New
Minimum	560	2
Maximum	60226	234
Format	uint16	uint8
Utilization ratio	7.346e-01	8.945e-01
Size (MB)	153.1	76.6

To visualize impact on histogram: Calculate/Histogram

☑ Calculate utilization ratio of data type Update table

Figure V-11. (Left) Before, and (right) after format conversion from 16 bits unsigned integer to un 8 bits unsigned integer with the rescale choice. Note that with the default 8bit conversion, not rescale, (i.e., uinnt8), the result is a uniform 255. (Bottom) Table indicates volume values bounds, format, utilization ratio and size.

5. Upscaling and downscaling

You can perform 3D microstructure upscaling and downscaling in the 'Up/down scaling' tab. Initial voxel size is optional, as you only need to enter the scaling factor (new voxel size = initial voxel size x scaling factor) and choose the data type (grey-level of label). Image resolution change is done with the file function_scaling.m (src\Miscellaneous) and is used by other modules as well. Upscaling algorithm is mostly used in the generation module (it is faster to generate a microstructure with a coarse representation, and then upscale it, rather than generate it directly at the fine resolution), while downscaling is mostly used in microstructure characterization (to reduce calculation time for some CPU expensive algorithms and to perform image resolution analysis). Both upscaling and downscaling algorithms are used in the meshing module. Changing image resolution algorithm depends on the image type:

- For a grey-level image, a linear interpolation is used using the MATLB built-in function *imreszie3*. Figure V-12a illustrates it.

```
Microstructure_resized = imresize3(Microstructure,scaling_factor,'linear')
```

- For a segmented image, linear interpolation is not recommended as it will generate irrelevant intermediate value if more than two phases exist (cf. Fig. V-12b). The function imresize3 could be used instead with the option 'nearest' (Nearest-neighbor interpolation) to remove this issue. Although it is suboptimal for upscaling segmented image. Instead, an in-house custom scaling algorithm is employed, that will refine the interface (cf. Fig. V-12c):

```
domain_size = size(Microstructure);
% Create a binary volume
binary_volume = zeros(domain_size);
binary_volume(Microstructure~=background)=1;
% Upscale or downscale it
allphases=imresize3(binary_volume,scaling_factor,'linear');
idx_void = allphases<=0.5;
allphases(idx_void)=0;
allphases(allphases>0.5)=-1; % The shape we want to reach, with the phase information

phases_id = unique(Microstructure);
phases_id(phases_id == background)=[];
number_phase = length(phases_id); % Minus background
for current_phase=1:1:number_phase
    % Create a binary volume
    binary_volume = zeros(domain_size);
    binary_volume(Microstructure==phases_id(current_phase))=1;
    % Upscale or downscale it
    phase_resized=imresize3(binary_volume,scaling_factor,'linear');
    idx_phase = phase_resized>0.5;
    phase_resized(idx_phase)=1;
    phase_resized(phase_resized~=1)=0;
    allphases(logical(phase_resized))=phases_id(current_phase);
end

% -1 values are the voxels that refine the interface, but they still miss the phase information
assigned_resizedvolume = zeros(size(allphases));
assigned_resizedvolume(allphases>0)=1;
[~,Idx_distance_map] = bwdist(assigned_resizedvolume);
unassigned_idx = find(allphases==-1);
Idx_distance_map(unassigned_idx);
allphases(unassigned_idx) = allphases(Idx_distance_map(unassigned_idx));

% And finally for the pore
```

```
allphases(idx_void)=0;
Microstructure_resized = allphases;
```

You can move the slider to explore the microstructure volume to compare side by side before and after the image resolution change before validating your change (cf. Fig. V-12a).

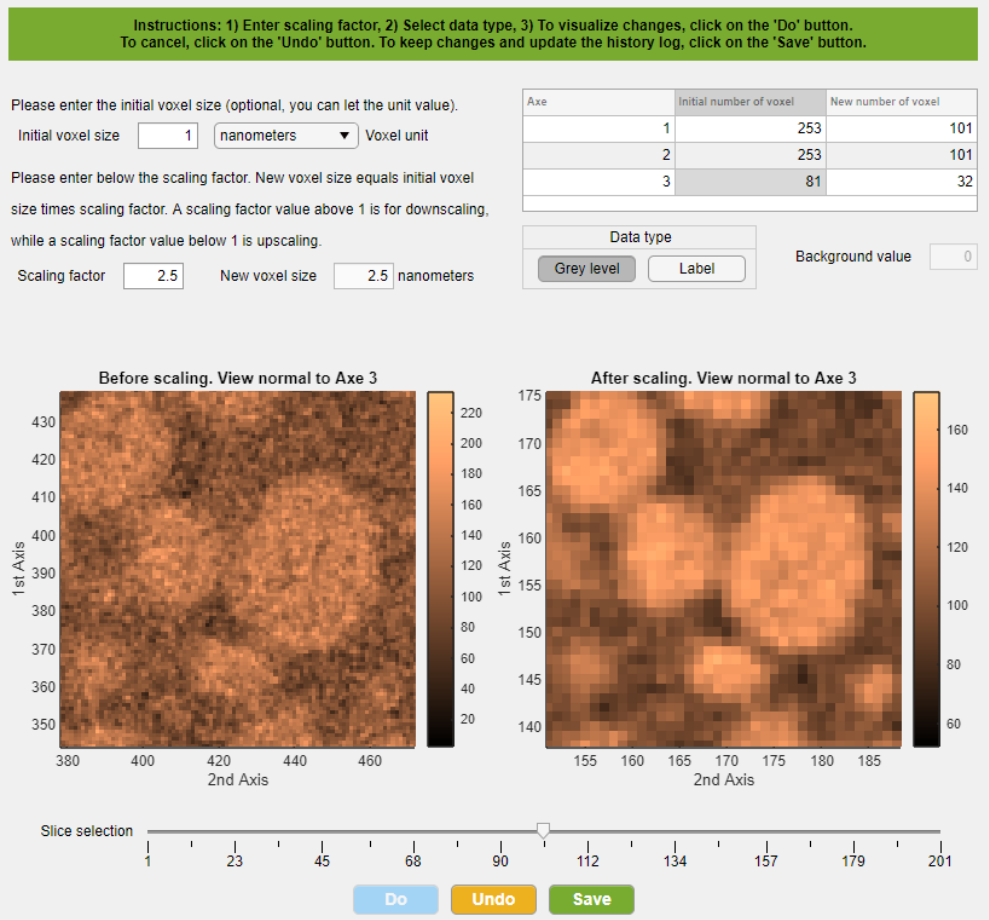


Figure V-12a. Downscaling (scaling factor=2.5 >1) of a grey-level 3D microstructure. Note that the scaling factor can be any real number. Imaged had initially 201 slices, now 81.

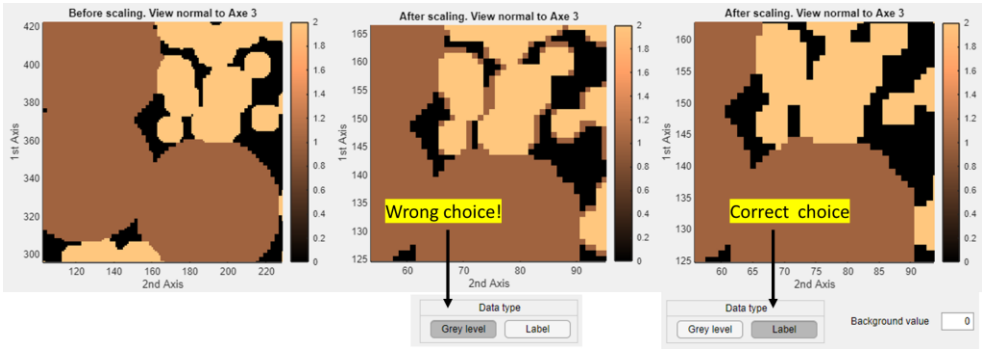


Figure V-12b. Downscaling (scaling factor=2.5 >1) of a segmented 3D microstructure.

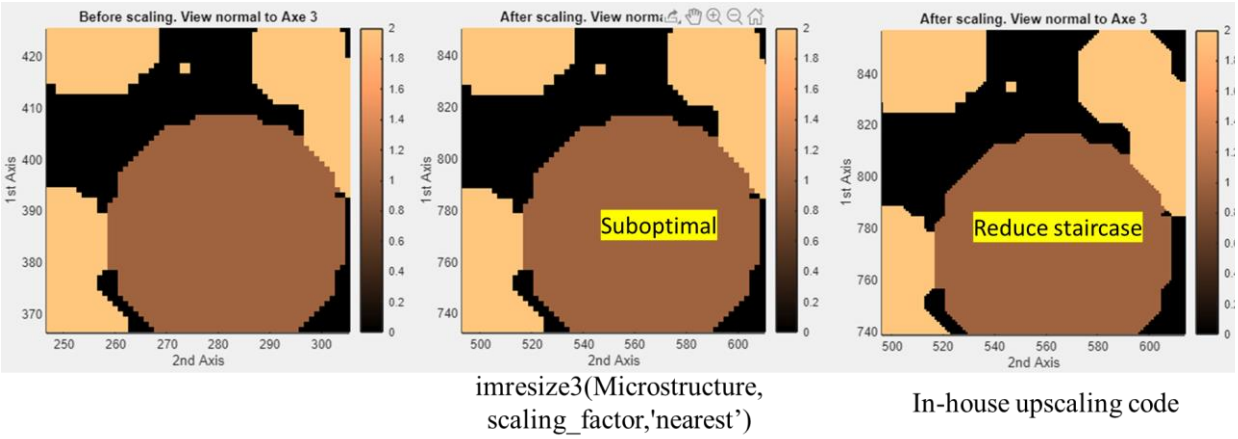


Figure V-12c. Upscaling (scaling factor=0.5 <1) of a segmented 3D microstructure.

6. Quantifying image quality

Evaluating the image quality is an essential step before segmentation, as it will help choosing the adequate segmentation method. You have access to image quality estimators in the menu/Calculate and in the 'Image quality' tab.

a. Grey level histogram and spatial homogeneity

Grey level histogram is a simple representation of the grey level value distribution within an image. Ideally, phases would be clearly marked with distinct peaks separated by valleys. For such an ideal case phase information is not ambiguous, not convoluted, and segmentation is straightforward. One objective of contrast enhancement and image filtering consists in improving the grey level histogram. Such an example can be found in[26].

- Grey level histogram

You can choose to plot the probability density of the grey level using the MATLAB built-in function histogram, either with automatic binning (Calculate/Plot histogram/for the whole volume/histogram(x) – bin auto) or with one bin per unique value (Calculate/Plot histogram/for the whole volume/histogram(x) – on bin per value).

```
if 'one_per_uniquevalue'
    nbins = length(unique(Microstructure));
    h=histogram(Microstructure,nbins);
elseif 'auto'
    h=histogram(Microstructure);
end
set(h,'LineStyle','none','Normalization','probability');
```

You can also choose to calculate the probability function with in-house functions (Calculate/Plot histogram/for the whole volume/histogram: probability density function). Normalized histogram is first calculated with function_calculate_histogram.m, then its probability density function is obtained with Function_probability_density.m, and plotted with function_probability_distribution_figure, both files being located in src\Miscellaneous. You can

manually edit the function parameters in the code (src\Filtering_and_segmentation\Segmentation.mlapp) if you want to filter differently the probability density function.

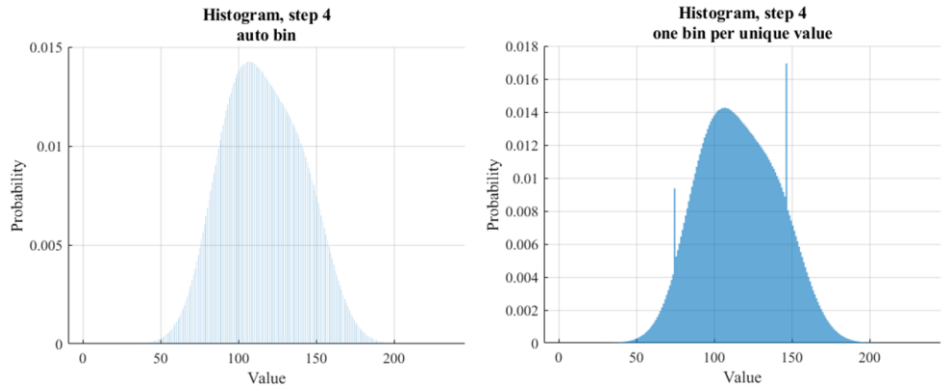


Figure V-13a. Probability density function of the grey level plotted with (Left) automatic binning, and (right) 1 bin per unique value. In this example, the two phases (pore and solid) are highly convoluted as no distinct peaks identified them clearly.

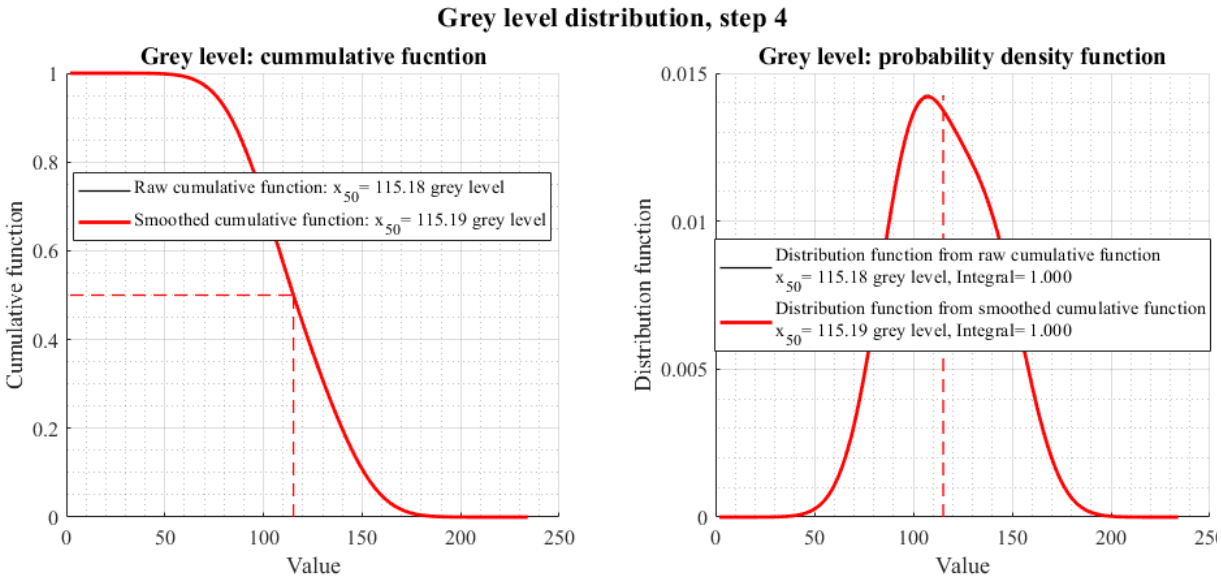


Figure V-13b. (Left) Cumulative function, and (right) probability density function calculated with in-house functions. Cumulative function is smoothed with a moving average filter so that its derivative is less noisy.

Grey-level calculated on the whole volume may not be representative of the microstructure. User can calculate the grey-level slice per slice (Calculate/Plot histogram/for each slice) to see if peaks and valleys are coherent all along the volume dimensions. If significant deviations are spotted, then contrast correction can be used to trying remedying it. If unsuccessful, global threshold methods are not relevant for the volume, and other approaches are required (e.g., slice-per-slice thresholding).

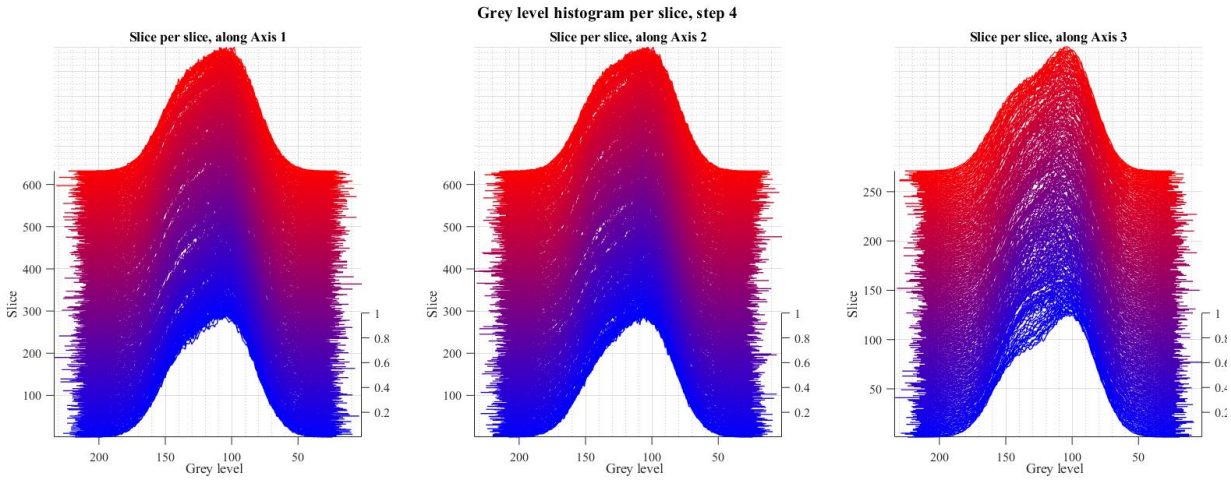


Figure V-14a. Grey level histogram calculated slice per slice. No deviation for all axis.

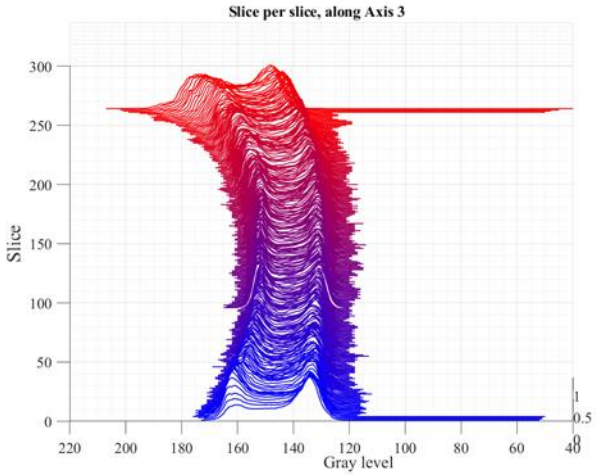


Figure V-14b. Grey level histogram calculated slice per slice for another volume, that exhibits deviation along its 3rd axis: global threshold cannot be used for segmentation!

- Grey-level spatial distribution

For non-graded electrode, the grey-level distribution should be spatially homogenous. User can check this by plotting average, extremums, and standard deviation of gray-level along volume dimension (Calculate/Plot grey/as a function of position).

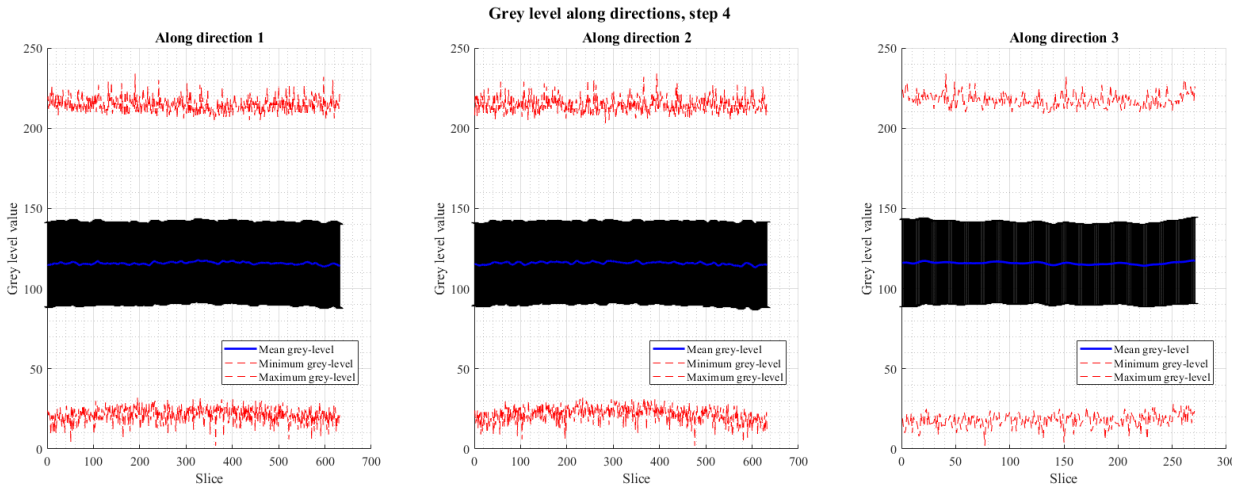


Figure V-15a. Grey level plotted along volume axis (mean, standard deviation, and extremums). No deviation for all axis.

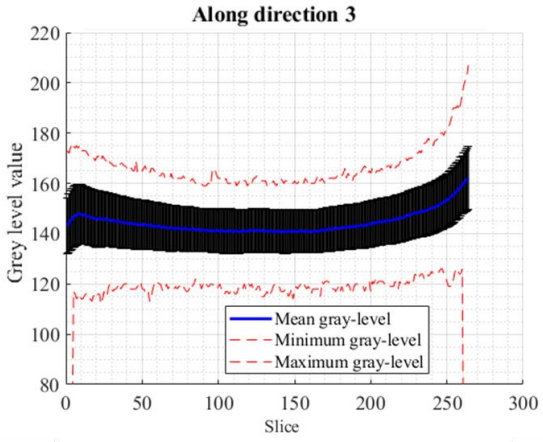


Figure V-15b. Grey level plotted along volume axis (mean, standard deviation, and extremums) for another volume, that exhibits deviation along its 3rd axis: global threshold cannot be used for segmentation!

As well, user can plot the averaged two-dimensional map of the grey level (Calculate/Plot grey/in averaged 2D maps), performed with function_2Dmap_3Darray.m in src\Filtering_and_segmentation, as:

```
% Normal to axe 3
greylevel_normal(3).map =  zeros(domain_size(1),domain_size(2));
for x=1:1:domain_size(1)
    for y=1:1:domain_size(2)
        line=reshape(Microstructure(x,y,:),[domain_size(3), 1]);
        greylevel_normal(3).map(x,y)=mean(line);
    end
end
```

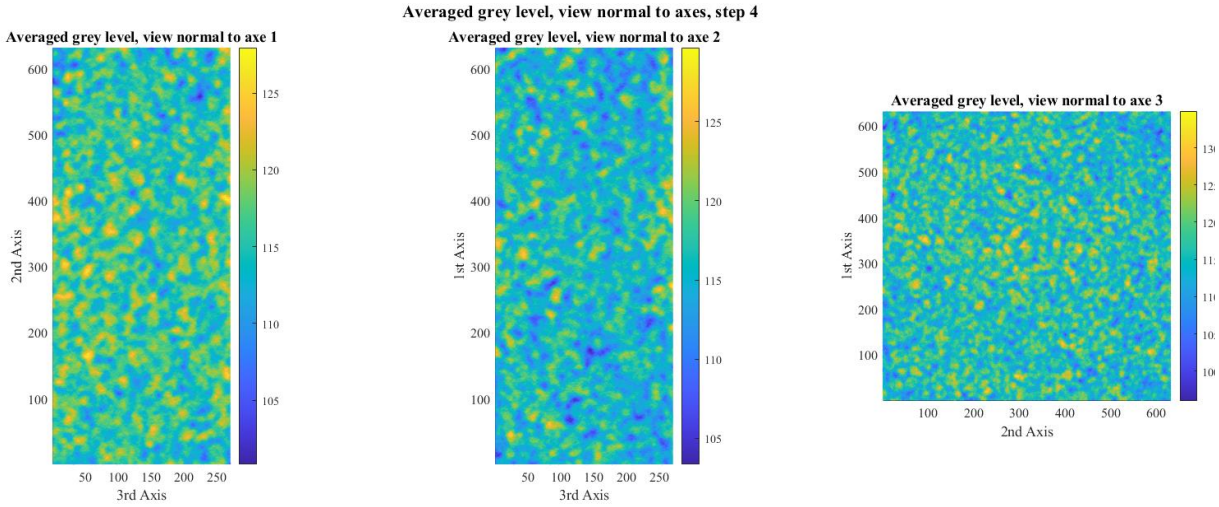


Figure V-16a. Average gray level with uniform brightness distribution

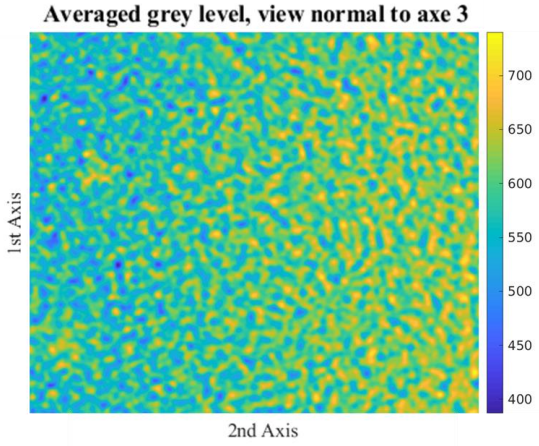


Figure V-16b. Average gray level calculated for another volume that exhibits an in-plane deviation that should be corrected with contrast enhancement before segmentation

b. Phase separability

Otsu's algorithm[27] is a histogram-based method used to segment images. Method is applied to microstructure in[26]. The algorithm consists in finding the threshold values that maximize the between-class variance σ_B^2 (phases have different grey-level values between them) and minimize the intra-class variance σ_W^2 (each phase have uniform grey-level values within). The total variance of the image is the sum of the two: $\sigma_T^2 = \sigma_B^2 + \sigma_W^2$. The phase separability criterion is then defined as $\eta = \sigma_B^2/\sigma_T^2$ and ranges from 0 (worst case: histogram is flat) to 1 (best case: each phase has a unique grey level value). The recommended thresholds are those maximizing η. The Otsu's algorithm is performed with the file Function_otsu_algorithm.m in src\Filtering_and_segmentation.

For a segmentation task, one is only interested in finding this maximum. For an image quality task, the slope near the maximum is the main interest. Ideally, η would decrease sharply from its maximum, indicating the algorithm strongly recommend using this value. On the opposite, if the slope near the maximum is low, it indicates the algorithm almost equally recommends a wide range of threshold without a clear recommendation, which is likely due to phases being highly convoluted (i.e., poor image quality). In such a case, any threshold chosen within this wide maximum region is coherent with the Otsu's method. Another metric interesting to evaluate the image quality is the evolution of the separability criterion along the volume dimensions. Non-constant values indicate the image quality is not uniform all along the microstructure dimension. A very low value locally suggest it may be adequate to crop this region due to its very poor quality that will bias the analysis. As well, it is interesting plotting the evolution of the recommended threshold slice per slice, based on the separability criterion calculated slice per slice. In case the threshold is varying significantly, a global threshold method is to be avoided in favor for a more local approach. You can calculate the separability criterion η in the 'Image quality' tab (you need to select the number of phases, as it is the only algorithm parameter). Note that no segmentation is performed.

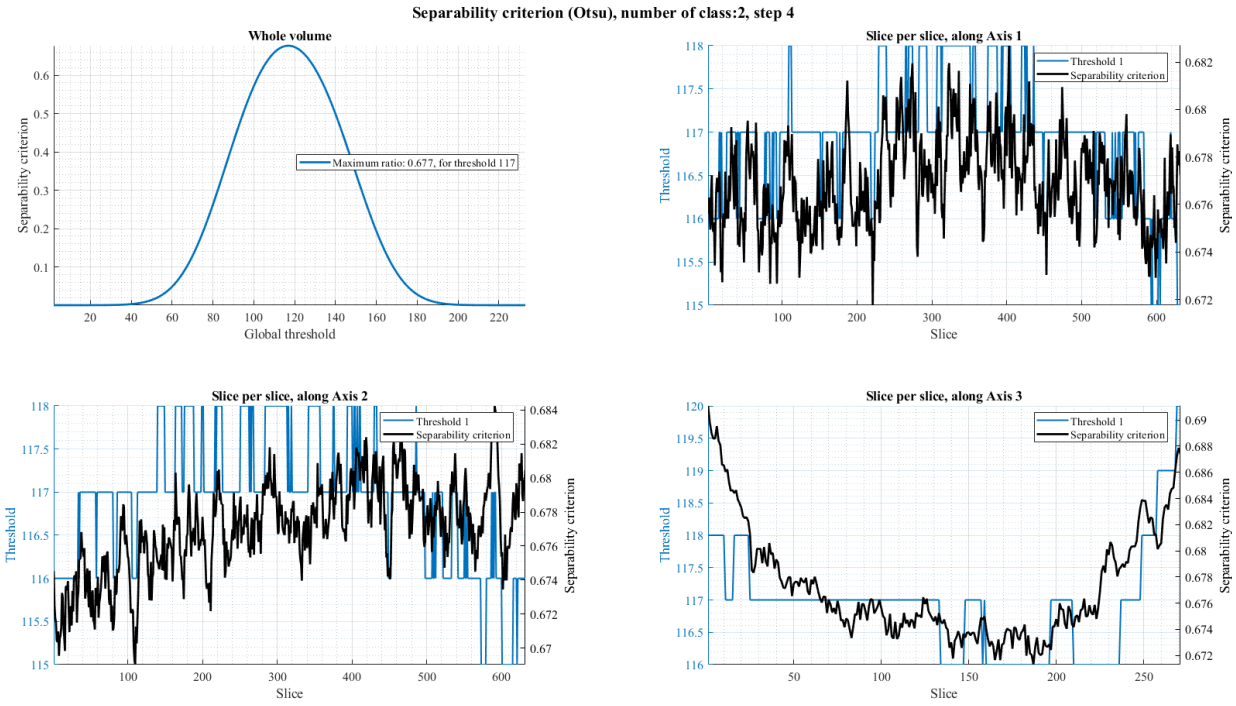


Figure V-17. (top left) Separability criterion calculated for the whole volume. While recommended threshold is 117, using a threshold ranging from 110 to 125 seems equally good (~6.3% of the grey level range). (Others) maximum separability criterion and associated threshold plotted slice per slice along volume dimensions.

c. Image noise

Noise level is currently calculated with the function NoiseLevel.m from Masayuki Tanaka[6]. It is particularly relevant to quantify locally image noise to identify problematic regions, and to evaluate the impact of image filtering in noise removal. You can calculate image noise with Calculate/Plot noise level/for each slice.

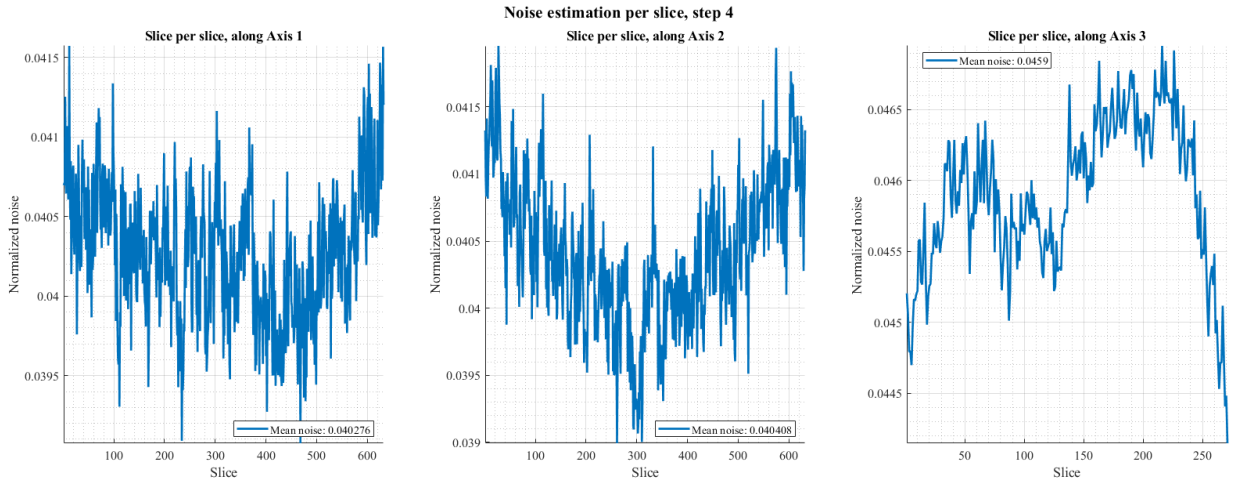


Figure V-18. Image noise calculated slice per slice.

7. Contrast enhancement

If the image quality is too low for segmenting the image, contrast enhancement algorithms can be employed to improve it. Few methods are available in the 'contrast correction' tab. You can compare side by side the impact of contrast correction moving the slider at the bottom of the tab. MATLAB provides interesting documentation on contrast correction: https://www.mathworks.com/help/images/contrast-adjustment.html?s_tid=CRUX_lftnav

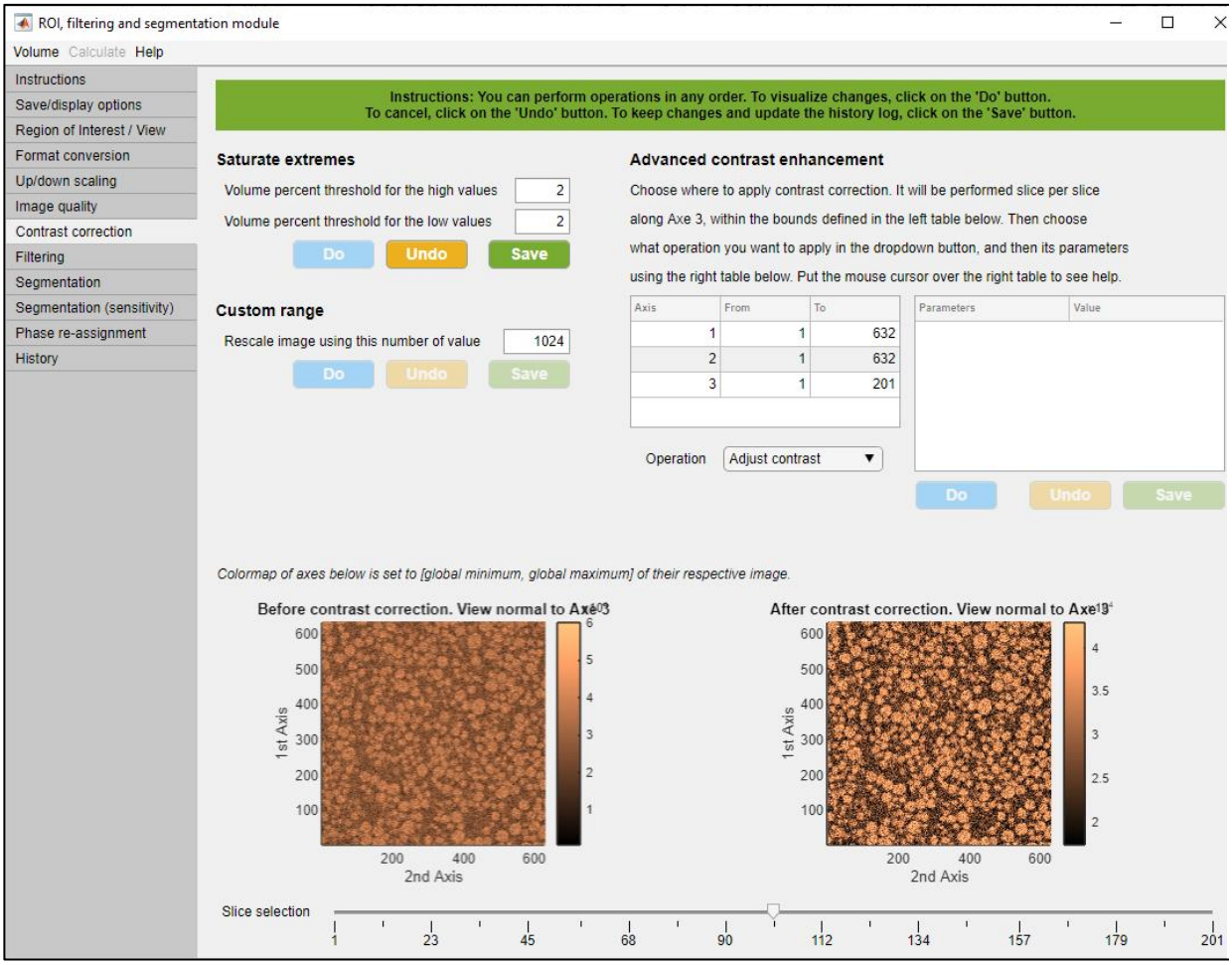
a. Saturate extreme.

Grey level threshold for which x % of the voxels have higher values is first found, then all these voxels are assigned to this threshold. The lower values are removed similarly. This is particularly relevant to remove the tails of the histogram that may use a very large range of grey level values. This allows to re-center the grey level values.

b. Set custom grey level range

Sometimes the data format does not provide enough flexibility for the desired grey level granularity. For instance, you may consider 1024 grey level to be a good compromise between the 256 values offered by 8-bits and the 65536 by 16 bits (even though, if you use 1024 grey level values, you are using at least a 16-bit data format). You can specify your custom range, and then microstructure will be re-scaled accordingly. It is recommended to first saturate the extreme and then set your custom grey level range so that all the grey level range is used more equally.

```
max_=max(Microstructure);
min_=min(Microstructure);
initial_delta=max_-min_;
final_delta=Custom_greyscale-1;
Microstructure=  round( (((Microstructure_undo-min_) ./ initial_delta) .* final_delta)+1 );
```



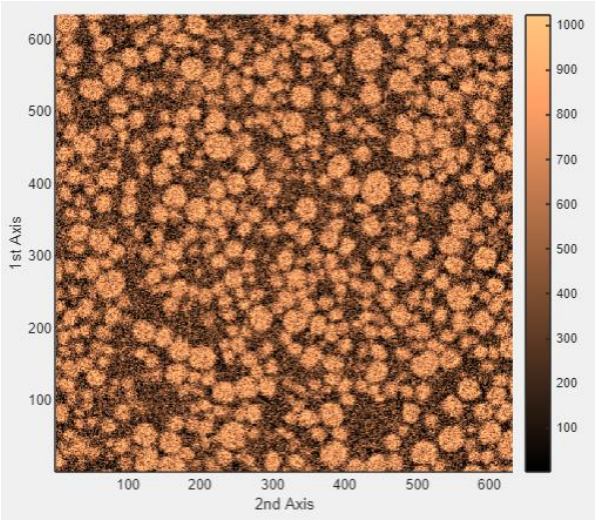


Figure V-19. (top) Contrast correction tab illustrated with saturate extreme and (bottom) microstructure after custom range (1024).

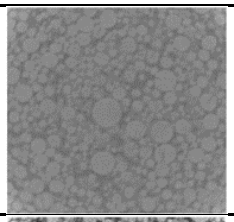
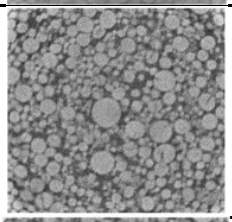
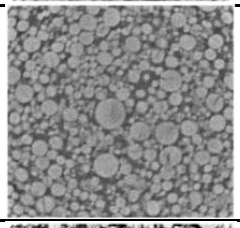
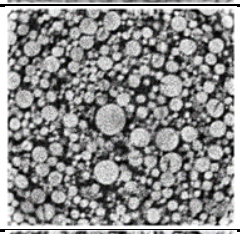
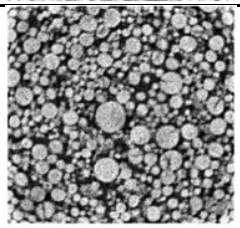
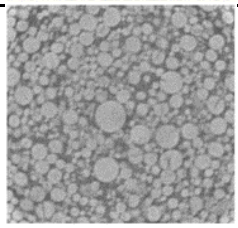
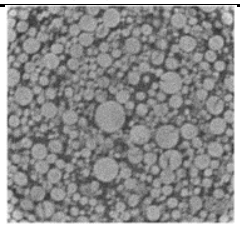
c. Advanced contrast enhancement

In the top right section of the tab, you have access to other contrast correction methods. First select the method with the popup menu 'Operation' and then fill the parameter in the table on the far right. You can also choose to apply the contrast correction locally specifying bounds in the mid-table. Note that these functions are applied slice per slice along the third axis.

- **Adjust contrast**. Uses the MATLAB built-in function *imadjust*. J = imadjust(I,[low_in high_in],[low_out high_out],gamma) maps intensity values in I to new values in J such that values between low_in and high_in map to values between low_out and high_out, where gamma specifies the shape of the curve describing the relationship between the values in I and J. If you set the last parameter 1, then J = imadjust(I) saturates the bottom 1% and the top 1% of all pixel values.
- **Histogram equalization**. Uses the MATLAB built-in function *histeq*. 'J=histeq(I,n) transforms the grayscale image I, returning in J an grayscale image with n discrete gray levels. A roughly equal number of pixels is mapped to each of the n levels in J, so that the histogram of J is approximately flat. The histogram of J is flatter when n is much smaller than the number of discrete levels in I.
- **Contrast-limited adaptative histogram equalization**. Used the MATLAB built-in function *adapthisteq*. J = adapthisteq(I,…) enhances the contrast of the grayscale image I by transforming the values using contrast-limited adaptive histogram equalization (CLAHE). This algorithm identifies a 'neighborhood' around each pixel of an input image. These neighborhoods are referred to as *tiles* (or *contextual regions* as defined by Pizer et al.[28] and Zuiderveld[29]) with the number of tiles along each in-plane axis set by NumTiles A and B in the parameter table. The algorithm applies a contrast transformation to each tile, as opposed to the entire image. The transform used maps each tile's histogram of pixel values to a new histogram distribution, specified by the user ('uniform', 'rayleigh', and 'exponential', respectively to create a flat, a bell-shaped, or a curved histogram). Because adaptive histogram equalization tends to overamplify contrast in homogeneous regions, the clip limit (or contrast factor) is a useful parameter to lessen the amplification of noise when using this algorithm. The contrast is amplified proportionally to the slope of the grey level cumulative distribution function for each tile, and so by clipping each tile's histogram before computing the cumulative distribution function, the noise amplification is limited. There is no explicit method to select the most adequate parameters for a given image, thus their determination requires some trial-error experimentations. The table below illustrates the impact of a selection of some parameters for an NMC electrode slice, with for this particular case the clip limit parameter providing the best image enhancement.

Please refer to the MATLAB documentation for more details about the built-in functions mentioned above. It is recommended to work with uint8 or uint16 file format.

Table V-1. Impact of CLAHE algorithm's parameters on an NMC slice.

Parameter *Other parameters with default values*	**Otsu's separability criterion η for a two-phase segmentation (1 is ideal), cf. §VI-6b**	**2D Slice**
Original volume	0.6466	
Default parameters	0.7072	
NumTiles = [16 16]	0.7078	
ClipLimit = 0.04	0.7524 (maximum among the investigated parameter sets)	
NBins = 1024	0.7501	
Rayleigh, alpha = 0.6	0.7007	
Exponential, alpha = 0.1	0.7067	

'Histogram equalization' and 'Contrast-limited adaptative histogram equalization' are particularly valuable to correct contrast deviation, if any (c.f. §V-6). Figure V-20a illustrates through-plane and in-plane contract deviation correction, while Figure V-20b illustrates ring artifact (defines as a radial variation of the grey level from the image center to its edges) correction.

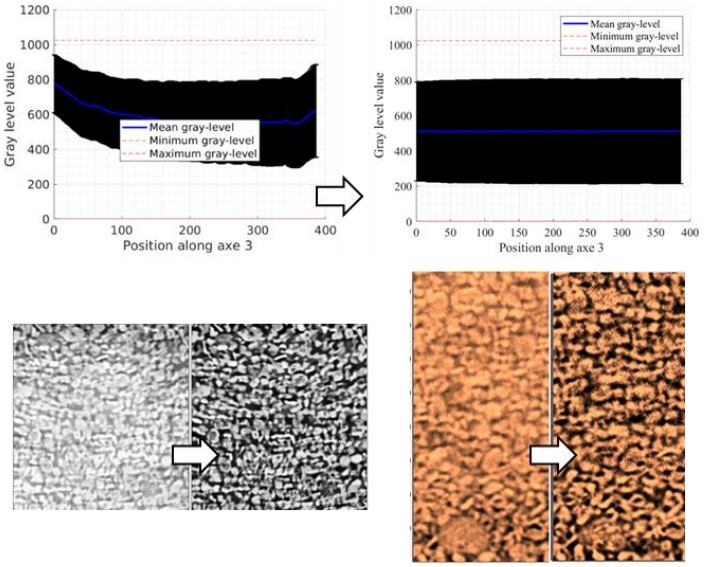


Figure V-20c. Example of (Top) through-plane and (bottom) in-plane grey level deviation correction achieved with contrast correction.

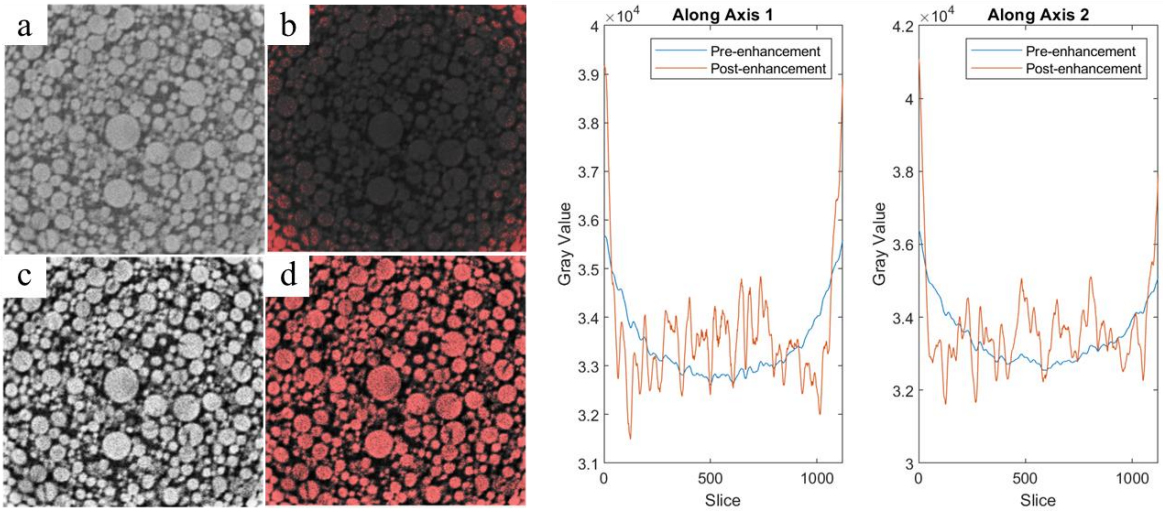


Figure V-20c. A depiction of the impact of contrast enhancement on segmentation. (Left): (a) Raw 2D slice, (b) tentative segmentation on the raw image reveals grey level is not uniform, (c) grey level image after applying CLAHE, and (d) segmentation on the contrast enhanced slice is now possible. (Right) Average grey-level along both in-plane axes, before (blue), and after (red) applying CLAHE. Note that while the ring artifact has been mostly removed by the contrast enhancement algorithm, it has also induced some noise. A subsequent image filtering step is then recommended.

8. Image filtering

Similarly with contrast enhancement, MATLAB provides also a detailed documentation about image filtering, available at https://www.mathworks.com/help/images/linear-filtering.html. Two popular image filtering algorithms are currently implemented in the GUI, in the 'filtering' tab. These algorithms can be CPU expensive, therefore, to avoid wasting time on tuning parameters on the whole volume, you can ask to apply filter on only one slice ('Preview' check box). If you run an algorithm slice per slice, the progress is printed within the tab.

a. Anisotropic diffusion filter

The anisotropic diffusion filter has been developed by Perona and Malik[22]. It consists in considering grey level values as concentration or temperature values, for which a diffusion calculation is performed with anisotropic, locally defined, diffusion coefficient. The latter are related to the local gradient according to a user-defined function, with different choice possible (see conduction method popup menu). The main idea being a high diffusion coefficient is applied within uniform region (i.e. within phases) while a low diffusion coefficient is applied in fast-changing regions (i.e. phases' edge) so that the filtering smooths the image within phases and enhance the contrast at the phases edge. Application of such algorithm on microstructure can be found in[26] with illustration. You can apply the filter slice per slice or the whole volume directly. Parameters (number of iteration and gradient threshold) can be estimated or manually entered. The MATLAB built-in function *imdiffusefilt* and *imdiffuseest* are used respectively to calculate the diffusion and to estimate the parameters if asked.

```
[Gradient_threshold,number_iteration] =
imdiffuseest(Microstructure,'Connectivity',Connectivity_,'ConductionMethod',Conduction_Method);
Microstructure =
imdiffusefilt(Microstructure,'Connectivity',Connectivity_,'ConductionMethod',Conduction_Method,'G
radientThreshold',Gradient_threshold,'NumberOfIterations',number_iteration);
```

b. Non-local mean filter

Applies a non-local means-based filter to the grayscale image according to the algorithm introduced by Buades et al.[23].

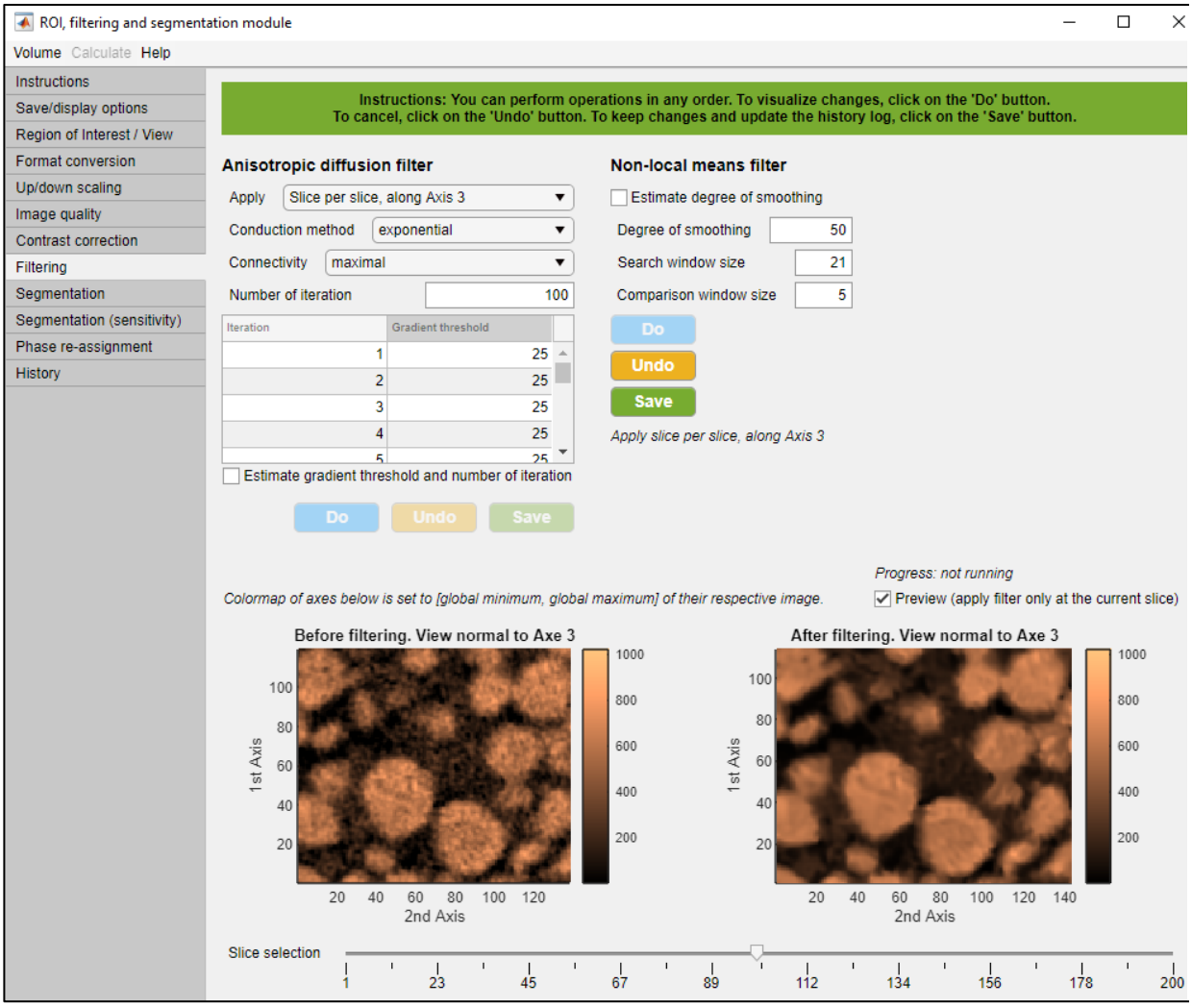


Figure V-21. Image filtering illustration with Non-local mean filter method. Additional help (tooltips) appears if you hover the mouse over parameter fields.

9. Segmentation

a. Volume segmentation

User must first enter the voxel size, the number of phases and the expected volume fraction if known. Two class of methods are available for the segmentation: global and local (slice per slice) thresholding.

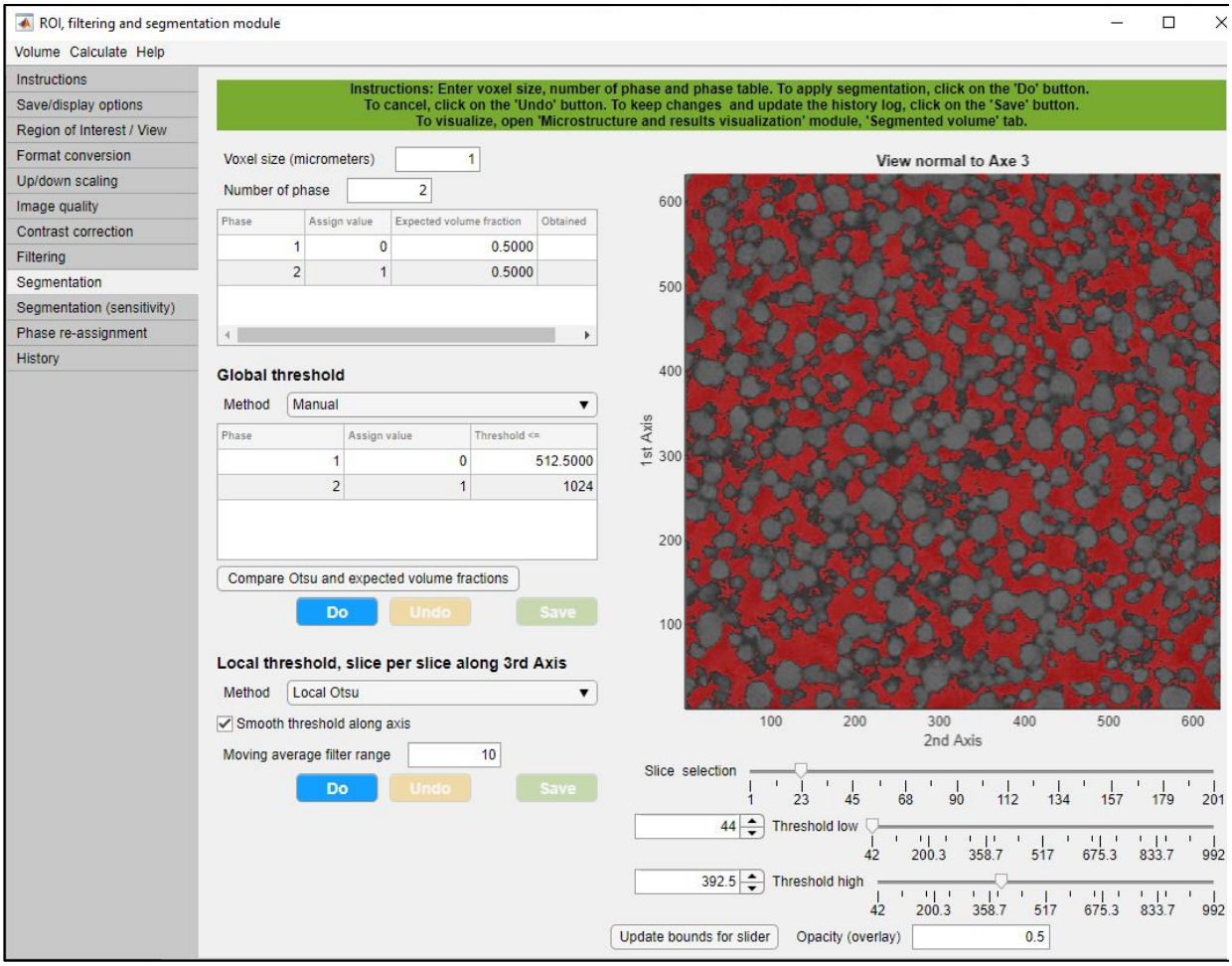


Figure V-22a. Segmentation tab.

i. Global thresholding

Thresholds are set either (i) manually, (ii) to match the expected volume fractions, or (iii) according to Otsu's algorithm (detailed in §V-6b)[27]. Thresholds from choice (ii) and (iii) can be compared clicking on the button 'Compare Otsu and expected volume fractions'. Errors of both methods are plotted together for compassion. Errors from 'expected volume fractions' choice is defined as volume fraction – expected volume fraction, with the reference value being the expected volume fraction. Error from Otsu's algorithm is defined as the separability criterion η – the maximum separability criterion η^* with the reference value being the maximum separability criterion. The graph allows user to evaluate if Otsu's algorithm is coherent with the expected volume fractions. If you cannot match the expected volume fractions using the associated option, it indicates the grey level range is not enough refined (image is too compressed for instance).

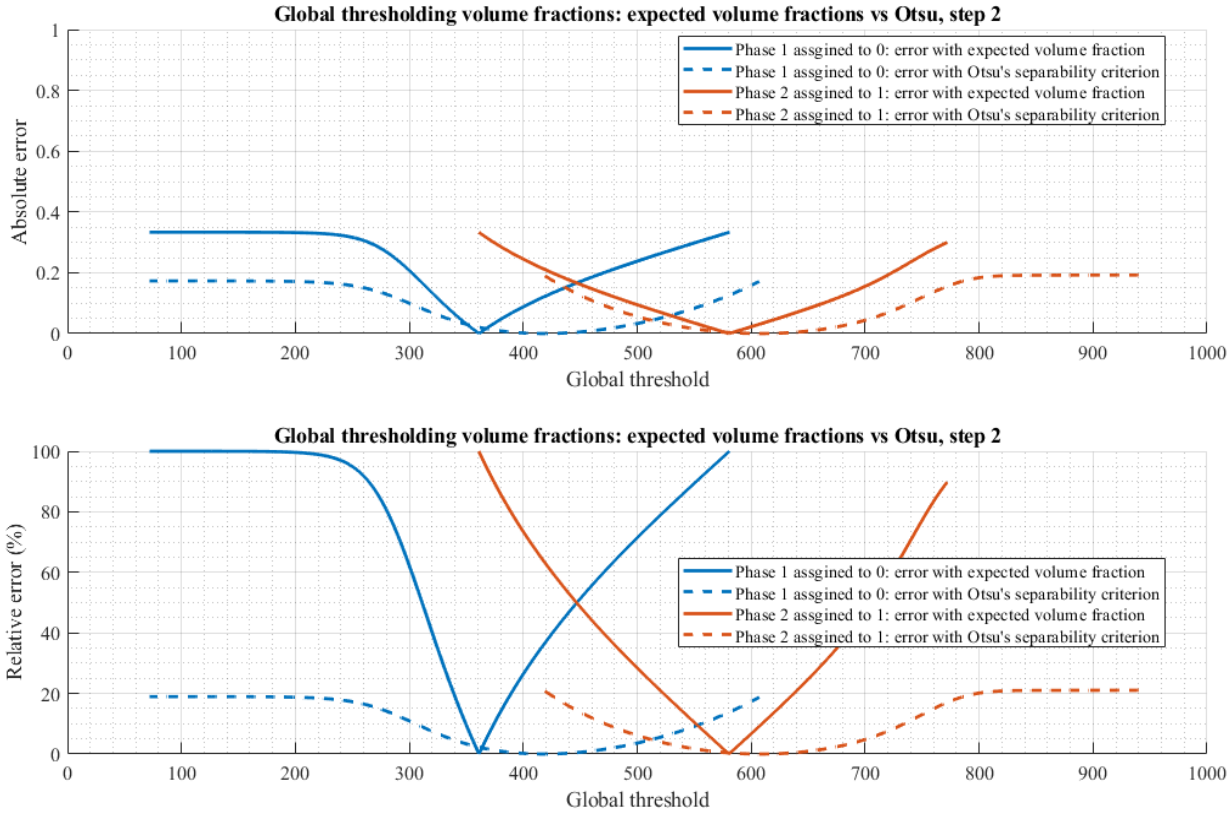


Figure V-22b. Volume fraction error compared between global thresholding methods (expected volume fraction and Otsu's algorithm) illustrated with three phases.

ii. Local thresholding slice per slice

Local threshold determined slice per slice is a relevant choice is the grey level exhibits a deviation along the third axis. If a deviation is spotted along another axis, simply swap axis. Thresholds are set slice per slice either (i) applying Otsu's algorithm slice per slice or (ii) applying Otsu's algorithm slice per slice but adding a global threshold correction so that the obtained volume fractions are those expected. To avoid a discontinuity from slice to slice, the function threshold=f(position along third axis) is smoothed using a moving average filter for which the user can choose the range.

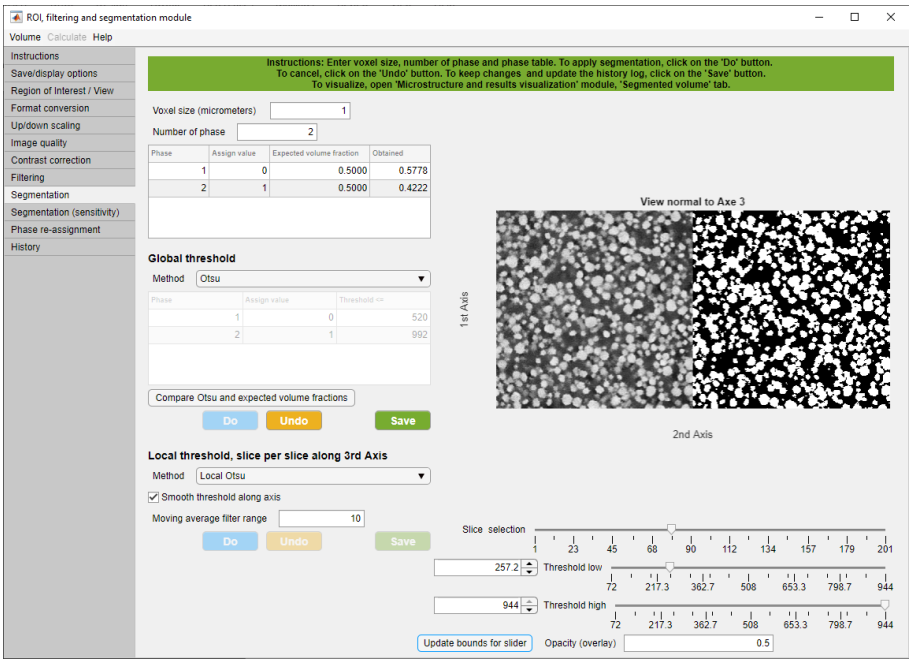


Figure V-22c. Once segmentation has been calculated, the GUI is updated to show side-by-side the grey-level and the segmented image for comparison. Obtained volume fractions are showed in the top table.

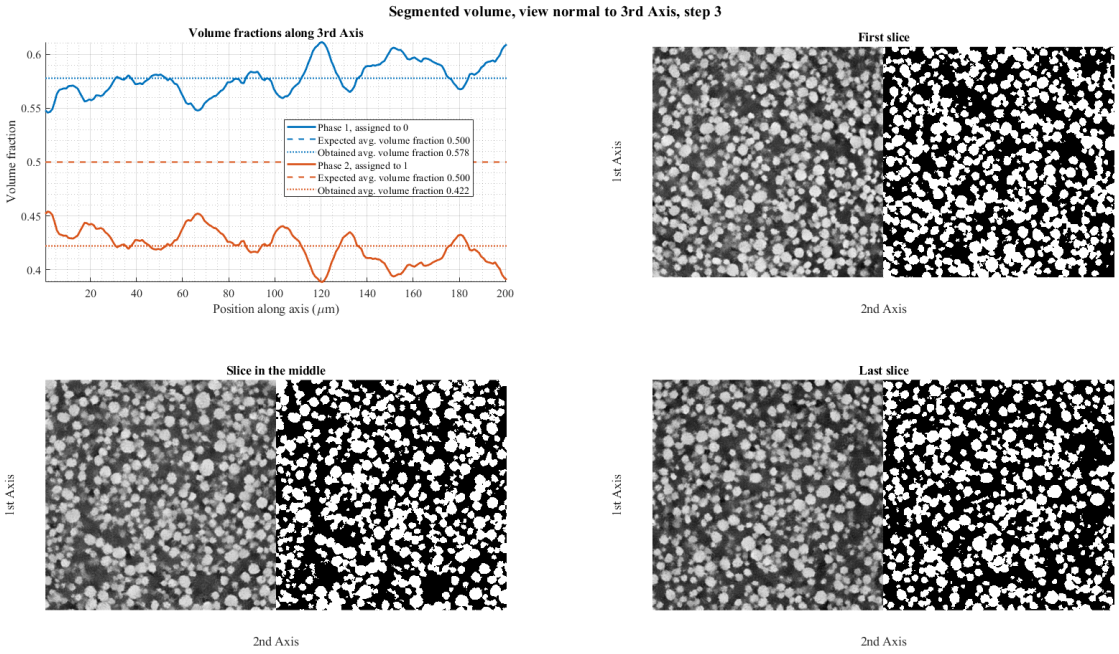


Figure V-22d. If the user clicks the 'Save' button, volume fractions are calculated slice per slice and displayed. Grey-level and segmented images are also compared side-by-side for the first, in the middle, and last slices for each direction to check segmentation quality is correct not only in the bulk but also at the domain's edge.

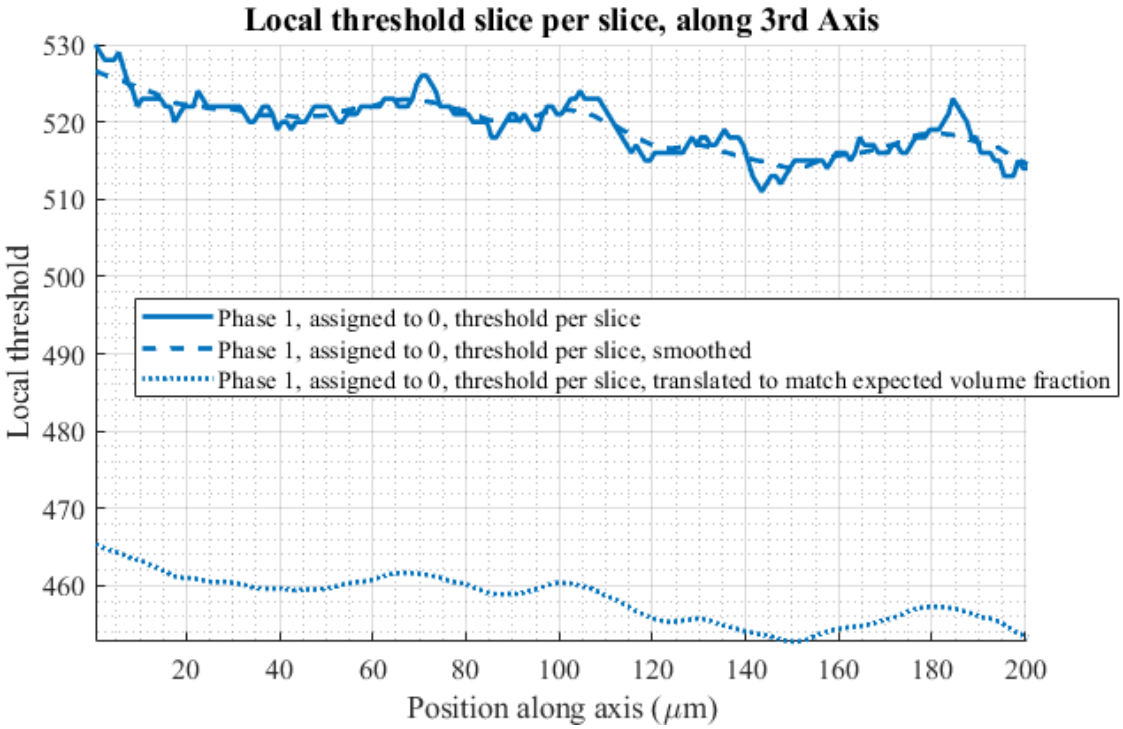


Figure V-22e. If the user clicks the 'Save' button and that local thresholding has been chose, then thresholds are also plotted along third axis position. If the choice 'Local Otsu translated to match expected volume fraction' has been chosen, then a third line is displaying showing the translated values used for the segmentation.

Note: even though the algorithm are n-phases, segmentation of more than two phases is likely to introduce a thin layer artifact at the phase interfaces. See figure 7 of reference[30] for illustration. Numerical methods to remove such artifact exist[30,31] and may be implemented in the future.

b. Microstructure properties sensitivity with thresholding

An uncertainty may exist for the threshold choice, as several values may provide visually very similar segmentation. The question arises then on the error propagation from the segmentation error to the microstructure property error. Alternatively, a user may be very confident on the choice of his threshold but still want to know how much a slight change from its segmentation will impact the microstructure properties. A small variation would strengthen the microstructure characterization results, while a significant variation will decrease the level of confidence attributed to the results. In the 'Segmentation (sensitivity)' tab, the user can set up such a segmentation sensitivity analysis for a two phase materials (pore and solid) on four microstructure parameters typically used in macroscale modeling: the porosity, the particle diameter (using a simple spherical assumption), the specific surface area, and the tortuosity factor. To avoid calculating properties on obviously incorrect segmented volumes, the user can specify the lower and upper bound of the porosity. To reduce calculation times, the user can specify the maximum

number of thresholds considered between the two bounds. Then, the user can choose if he wants to calculate the four parameters or only some of them. Lastly, the user enters the voxel size and click on the 'Calculate parameter sensitivity with threshold'. For simplicity sake, a simple global threshold swap is performed. Progress is indicated within the tab. Results are displayed as each parameter is investigated. Because a grey-level do not have the same signification for every microstructure (a unit variation is much more important for a 8-bits than for a 16-bits volume), grey levels axis are represented both with their absolute value but also as a percentage of their range as:

```
unique_values = unique(Microstructure);
range_greyscale = max(unique_values)-min(unique_values)+1;
range_greylevel=100*unique_values/range_greyscale;
```

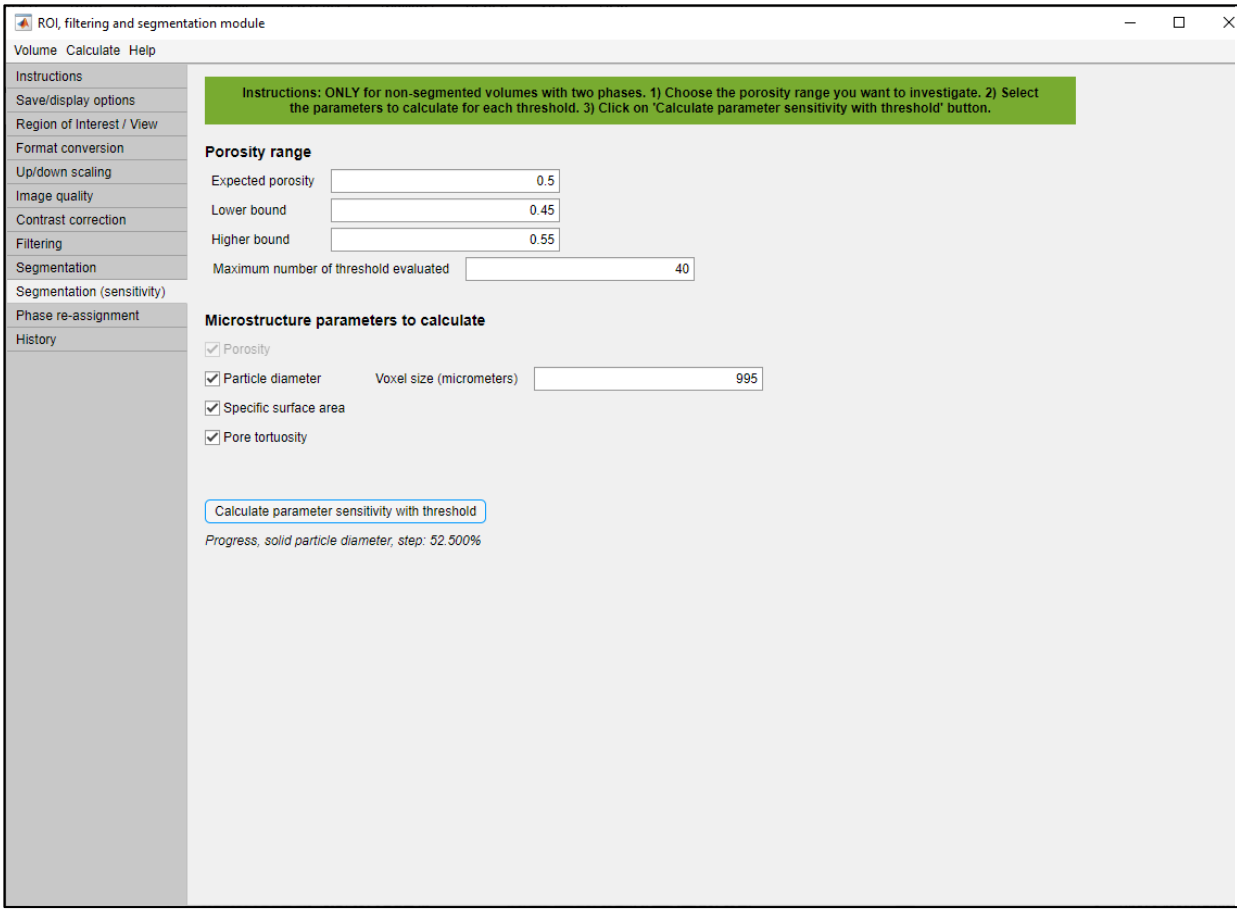


Figure V-23a. Microstructure properties sensitivity with global threshold tab.

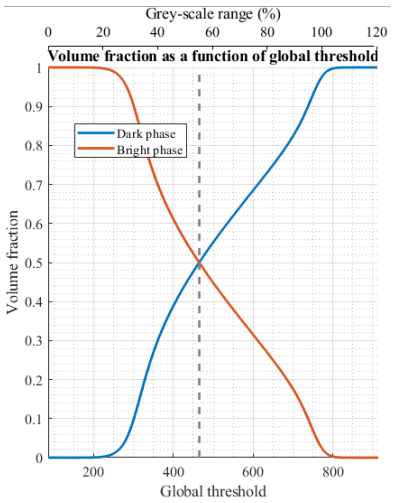


Figure V-23b. Volume fractions are by default calculated for the whole grey level space. Grey dashed line represents the value for the expected porosity indicated by the user.

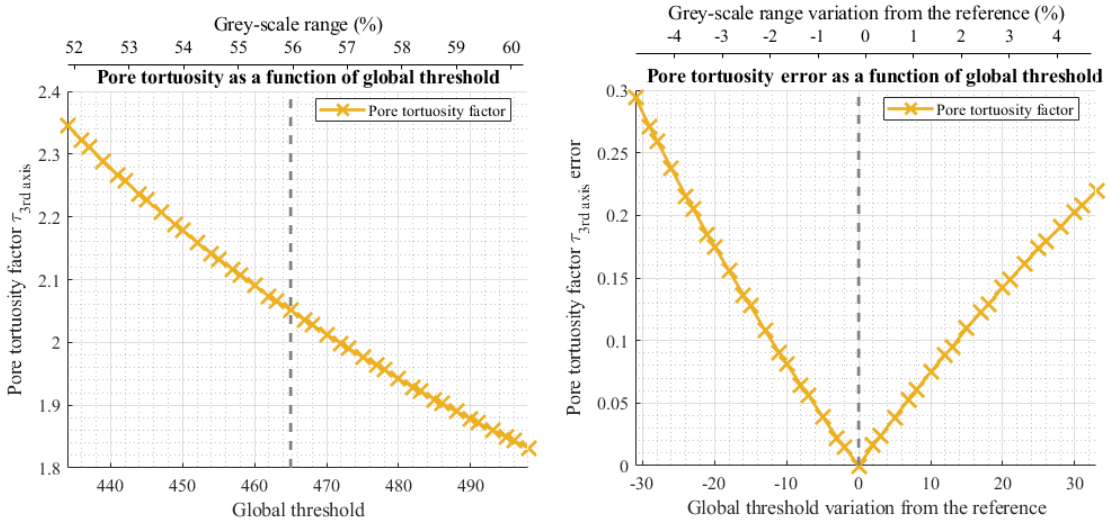


Figure V-23c. Microstructure property sensitivity with global threshold illustrated for pore tortuosity. (Left) calculated value, and (right) error.

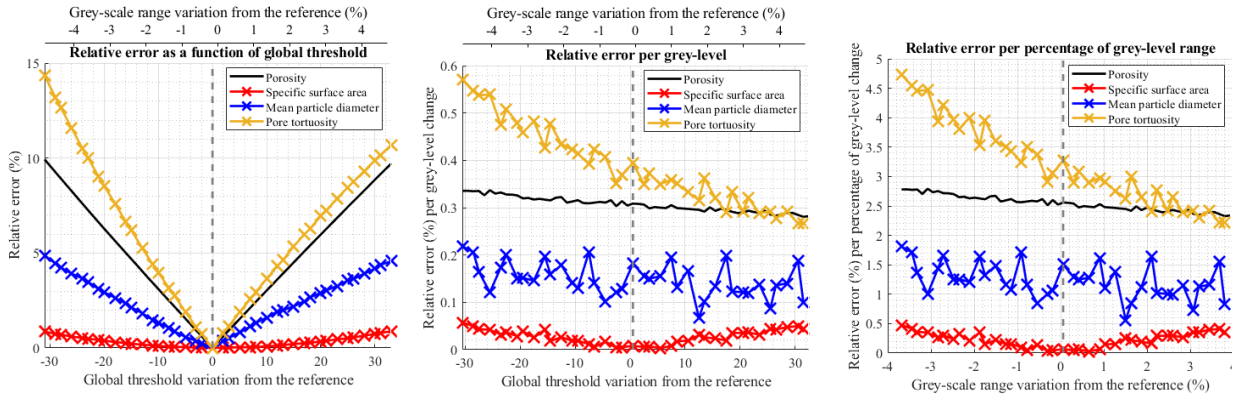


Figure V-23d. (Left) Relative error, (center) relative error change per grey-level change and (Right) relative error change per percent of the grey-level range

c. Phase reassignment

Once segmentation is completed, you can manually reorder the phases in the 'Phase re-assignment' tab.

VI. MICROSTRUCTURE CHARACTERIZATION AND HOMOGENIZATION

1. Definition, unicity, limitations, and pseudo-parameters

Microstructure characterization is applied on segmented volumes, for which each voxel identifies a phase. Microstructure characterization can be considered as a dimension reduction calculation. Microstructure volume is, from the numerical analysis point of view, a very large dimensional space: each element of volume (voxel) has a value that identifies what phase exists in this location. While this description is very accurate, at the extent of the image resolution, it is however impractical for describing in a quantitative and concise way the features or properties of a microstructure. These properties can be used for discussion and comparison ("this microstructure has a higher [property name] compared with this one") and modeling ("this [property name] is a parameter model"). Therefore, the properties to quantify from the microstructure depend on the application. For instance, in the battery field, porosity ε, specific surface area S_p, particle mean equivalent diameter D_{50}, and tortuosity factor along electrode thickness τ (definitions are provided in the section §VI-6) are the four microstructure parameters typically calculated as they are used in electrochemistry battery (macroscale) models[32,33], cf. Fig. VI-1.

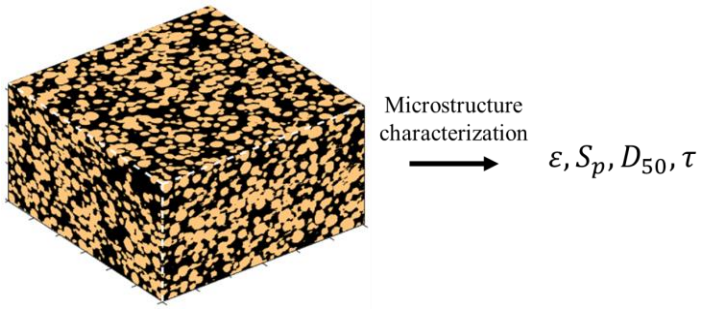


Figure VI-1. Example of a microstructure characterization for electrode battery modeling. (Left) microstructure of a positive NMC (Nickel Manganese Cobalt oxide) electrode material is represented with a 3D array with a very large number of voxels. (Right) the microstructure is abstracted with a limited number of scalars, each with a physical or geometrical definition relevant for the chosen application (electrode battery modeling).

Models are, by definition, a simplification of reality. In this frame, the microstructure volume is simplified with a limited number of properties, and microstructure characterization aim is to calculate them. This approach however is restrictive: it is not because a property is not used as a parameter for a given model that it is not important for the application. For instance, in battery electrode modeling, particles are assumed to be spherical in order to derive the model equations[32,33], while (i) this is obviously not true[34] and (ii) several authors suggest particle morphology impact electrode performances[35,36]. In this case, additional microstructure parameters worth be quantifying could be particle sphericity and elongation. Another example is that standard

models assume a uniform particle size distribution, while (i) electrode microstructures have more complex size distributions[34] and (ii) such distributions have an impact of the electrochemical response when modeled[37–39]. Restricting microstructure characterization to the sole parameters used in models may then induce the bias that all the relevant microstructure parameters are considered. It is then recommended not only to quantify the microstructure properties explicitly used as model parameters, but also additional microstructure properties that we expect impact system response too. Being able to determine these additional parameters provide an incentive for modelers to incorporate them in their model, eventually producing enhanced predictions. Furthermore, being able to finer characterize microstructures allows a more in-depth discussion, as more metrics can be used to discriminate microstructures.

It is unlikely a bijection exists between a microstructure volume m_i and a given property set p_j due to the huge discrepancy between their cardinality (the number of elements they contain): ~$10^{3\text{-}9}$ for a microstructure volume and ~10^1 for a microstructure property set. Therefore, for one or several set of microstructure properties values, multiple microstructure volumes exist as illustrated in figure VI-2. Say otherwise, values of a given set of microstructure properties do not identify uniquely a microstructure: microstructure characterization induces a loss of information that prevents recreating identically the microstructure used to calculate the properties. As the property set cardinality is increasing, the microstructure design space is more constrained and thus the number of possible microstructure volume that matches these properties is decreasing.

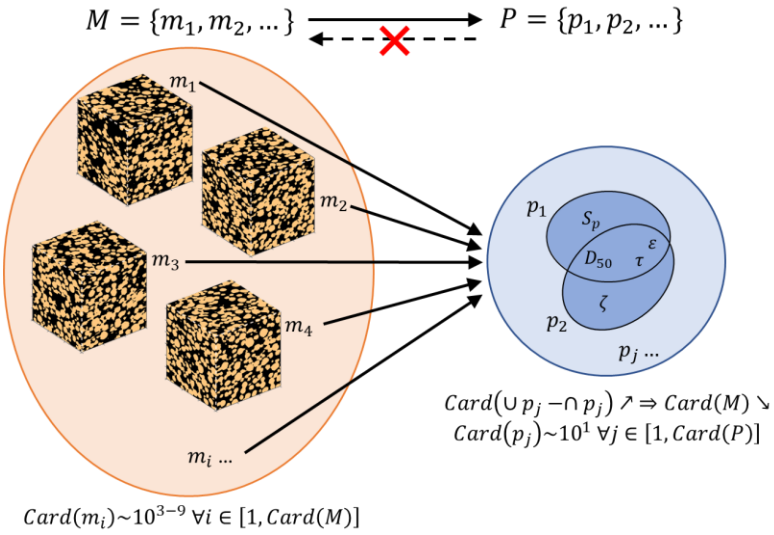


Figure VI-2. Bijection does not exist between a microstructure m_i (or a group of microstructure M) and a property set p_j (or a group of property set P). Property set are application specific. In this example p_1 is relevant for surface-type reaction (Specific surface area S_p), while p_2 is relevant for lineic-type reaction (triple phase boundary density length ζ).

Assuming the microstructure volume available for the microstructure characterization is a *valid* representation of the material, and that algorithms used to calculate the properties used as

model parameters are *adequate*, then the properties calculated are correct. A *valid representation* is a volume that satisfies various conditions: (i) investigated volume is statistically representative of the whole material volume used in the application, (ii) image resolution is high enough to capture all the relevant geometric features that can impact the system response, and (iii) segmentation is good enough to provide a close match with the actual microstructure. In practical, these three conditions are never perfectly achieved. Although they can be evaluated: (i) and (ii) are discussed in detail in §VI-2, while (iii) can be verified both globally (a correct segmentation is supposed to provide the expected phase volume fractions) and locally through visual comparison between the grey-level and the segmented images. Unfortunately, in some cases, due to the nature of the material phases and of the specifications of the experimental setup, some phases may be not visible which induces some uncertainty in the characterization results[17]. An *adequate algorithm* is a method for which (i) assumptions, if any, are relevant for the investigated microstructure (e.g. an algorithm that uses a particle spherical assumption is relevant only if the microstructure has indeed near-spherical particles) and (ii) parameters, if any, are known. Condition (i) can be checked. For instance, if the algorithm uses a spherical assumption, the particle sphericity can be calculated using another algorithm, or estimated through visual inspection (as done in[34], figure 22). As well, result sensitivity with assumption correctness can be estimated comparing characterization results obtained with numerically generated microstructures that differently satisfy the given assumption (e.g., generating ellipsoids with different sphericity and plotting algorithm result as function of the known sphericity). Condition (ii) is too algorithm-specific to have a generic discussion about it. Microstructure characterization is therefore limited by the difficulty to have a *valid representation* and *adequate algorithms*. Quality microstructure characterization work should pay special attention on evaluating both.

When microstructure properties are used as model parameters, and that model predictions are thereafter compared with experimental results, difference discussion should include the different source of errors in the analysis:

$$measured\ value = ground\ truth + experimental\ error$$
$$ground\ truth = predicted\ value + model\ assumption\ error + model\ parameter\ error$$

With 'model assumption error' being the cumulated error arising from the necessary simplifications of the actual system in order to being able to formulate model equations. These include geometry simplifications (e.g., particles are spherical), mathematical simplifications (e.g., non-linear equations are linearized), physic simplifications (e.g., mechanic is not considered) etc. Therefore, difference between measured and predicted value is equal to:

$$measured\ value - predicted\ value = experimental\ error + model\ assumption\ error + model\ parameter\ error$$

Experimental error can be estimated reading specifications of the experimental tools or measured (e.g., noise to signal ratio). Model parameter error can be also estimated, differently for each parameter, and model can be run between lower and upper bounds representing the parameter

confidence interval. On the contrary, evaluating model assumption error is more delicate, although not impossible. For instance, predictions calculated with models of different level of simplification can be compared to discriminate the precision cost of a given model simplification. The point of this paragraph is that even if you are confident with the model parameters, a discrepancy between the experimental and the model result is still possible. It is then common to fit one or more parameters to obtain a good agreement. If the fit implies that one or several microstructure parameters are assigned to values significantly outside of their respective interval of confidence, it is overfitting. This is a hint that the model assumptions are too strong, and/or you do not have a valid representation and adequate algorithms. The definition assigned to these overfitted model parameters do not match anymore with the definition of their equivalent microstructure properties. Such parameters are then called pseudo-parameters to reflect this mismatch. For instance, if you estimate with high confidence the tortuosity factor to lie between 2 and 2.5, and that your model requires a value of 3.5 to match experimental data, then the tortuosity factor used in the model is a pseudo-tortuosity factor. The out of bound pseudo-parameter values allow accommodate the model assumption.

a. Microstructure characterization and microstructure homogenization

Microstructure homogenization is a branch of microstructure characterization (as it still consists in extracting microstructure parameters from the 3D volume). The specificity of microstructure homogenization lies in how the microstructure is represented after its characterization, or simplification (cf. Fig. VI-3).

- Standard microstructure characterization, such as particle size quantification, considers the domain after simplification still as a multi-phase material, as there is no notion of particle size or specific surface area for a one-phase material. The difference with the initial 3D volume sits on the property level of complexity. In the figure below, the microstructure exhibits a wide size distribution, while its representation used for modeling only kept the mean diameter, thus neglecting the size distribution. Some models consider different particle size, so the level of simplification of the complex microstructure can vary depending on the models considered.
- Microstructure homogenization, such as effective diffusion coefficient quantification, considers the domain after simplification as a homogenous domain, i.e. a one-phase material. The idea being both the complex multi-phase microstructure and the one-phase homogenized domain responds identically at the macroscale for a given loading, e.g., the same flux for the same applied concentration or temperature gradient. Microstructure homogenization is used to determine the effective (i.e. macroscopic) material coefficient, such as diffusion coefficient[2,40] or stiffness tensor[41] (both that can be defined on a one-phase material) of an heterogenous domain, for which the coefficient value is known for the 100% dense phase (noted D_{bulk}).

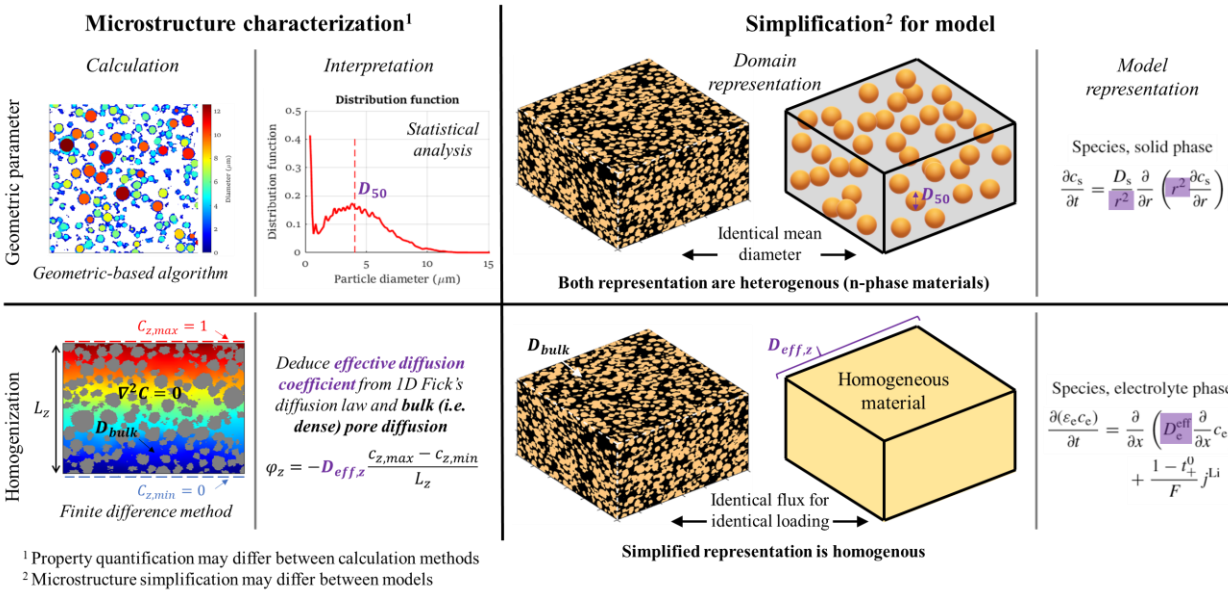


Figure VI-3. Examples of microstructure characterization for (top) particle size and (bottom) effective diffusion coefficient. Model equations from[33] *have been used.*

2. About voxel size dependence and representative volume element analysis

a. Image resolution and fractal issue

Properties exhibit different voxel size dependence, depending on they are volume properties (e.g. volume fraction, particle size), surface properties (e.g. specific surface area) or lineic properties (e.g., triple phase boundary length). We consider homogenized properties, such as effective diffusion coefficient or effective stiffness tensor to be volume properties (rule of mixture being the simplest method to determine them) even through it is an oversimplification. Several authors have investigated properties sensitivity with image resolution[17,34,40–42].

Except for very coarse image resolution that would degrade the material network, voxel size impacts only mostly the interface description. Therefore volume-based properties are converging with image resolution[17,34], and it is justifiable to extrapolate their value to an infinite image resolution to provide a better prediction. To do this, volume is downscaled iteratively in order to provide a curve property as a function of voxel size. The algorithm used to perform downscaling/upscaling is described in §V-5. Note that the dependence with voxel size of a given property may be method-specific[17]. The explanation of why a property is increasing or decreasing with voxel size depends then on (i) the particular microstructure investigated, (ii) the property, (iii) the algorithm used to downscale the microstructure, and (iv) the method used to calculate the property.

Unlike volume properties, surface and linear properties are not converging with image resolution as they exhibit a fractal behavior[42]. Such issue is briefly discussed by Usseglio et al[34] for battery microstructure, while the fractal issue has been thoroughly studied for solid oxide fuel cell anodes by Bertei et al.[42] and is related to surface roughness. As more interface details are visible at lower voxel size, surface and lengths are expecting to diverge with image resolution. Therefore, extrapolation is not relevant. Besides, an intrinsic issue lies in the fractal property's dimensionality: lineic fractals have a dimension between 1 and 2, while surface fractals have a fractal dimension between 2 and 3 (non-fractals have an integer dimension). Macroscale battery models typically use a morphology assumption, such as particles are spherical. Although, they rely on another, implicit, assumption that there is no surface roughness which ensure lineic and surface have integer dimensions. Surface and lineic properties absolute values obtained through microstructure characterization are not relevant for current macroscale models: only their dimension is converging, while their value is diverging. However, it is still worth it to calculate surface and lineic properties between different microstructures, at the condition they share the same voxel size, for comparison purpose. Calculating the fractal dimension is also valuable as it quantifies microstructure complexity as a ratio of the change in detail to the change in scale. Note that numerically generated microstructures are not adequate for such image resolution/fractal analysis study as surface is only refined but no new surface details are generated when generation is performed with smaller voxel size.

You can perform voxel size dependence analysis in the characterization module for each microstructure property. To do this, check the 'Perform voxel size dependence analysis' checkbox and choose the voxel size multiplier.

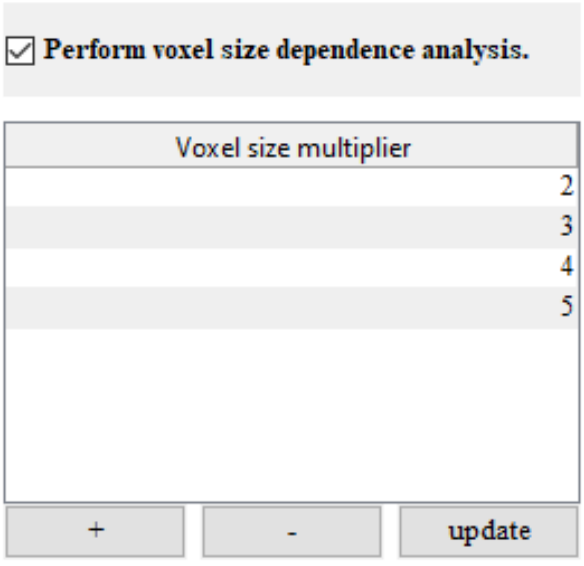


Figure VI-4. Graphic user interface for voxel size dependence analysis. In this example, property will be calculated with the original voxel of size $a \times a \times a\ \mu m^3$, but also with $2a \times 2a \times 2a\ \mu m^3$, $3a \times 3a \times 3a\ \mu m^3$, $4a \times 4a \times 4a\ \mu m^3$, and $5a \times 5a \times 5a\ \mu m^3$.

b. The representativity issue and the concept of representative volume element

i. Definition and methodology

Microstructure scale analysis are, by definition, limited to investigate only a fraction of the material volume used for the application. For electrode batteries, it corresponds to a volume < 1 mm^3, while electrode size is order of magnitude larger. Only submillimeter heterogeneities may be catch by microstructure scale analysis. While obvious, this statement is important to keep in mind as mm to cm scale local variations and defects may exist in the bulk of the electrode material, while material edge may also differ from the bulk, depending on the manufacturing process and on its quality control. Microstructure analysis can then only evaluate heterogeneities contain in the field of view and may not be representative of the whole material. Ideally, to evaluate if the measure if impacted by mesoscale heterogeneity, one would perform imaging on different sample locations.

Representative volume analysis (RVA), as defined in this toolbox, tries to answer this question: *"For a given microstructure property and a given subvolume aspect ratio, what is the minimum subvolume size that provides a relative standard deviation below a given threshold?"*. The method consists in cutting the microstructure volume into independent (i.e., non-overlapping) subvolumes of the same size and aspect ratio, and to calculate microstructure properties for each of them. The relative standard deviation is then calculated considering all the results. The process is repeated with varying subvolume size to produce the curve relative standard deviation as a function of subvolume size. When the curve crosses the relative standard deviation user-defined threshold, then the representative volume element size is found. The method is illustrated in figure VI-5. Because of the finite volume of the investigated volume, as the subvolume size is increasing, their number is decreasing. Crossing the threshold with a low number of subvolumes is less relevant from a statistical point of view, implying the confidence of the calculated RVE size is low and that a larger field of view is required. Not crossing the threshold indicates the field of view is too small to provide a RVE. Although it does not mean necessarily that the whole volume is not representative: RVA analysis can only tell if a subvolume is representative of the whole volume. This last point is important to set up the experimental imaging if determining the RVE is required. Considering a material with a particle size of ~10 µm and assuming that RVE lengths should be ~15 particle diameter (this rule of thumb is highly debatable as seen in [43], but for the sake of the demonstration we will use it), then the expected RVE size is $\sim 150 \times 150 \times 150$ μm^3. If your criterion to determine the RVE is a standard deviation of 5% obtained with *at least 9 subvolumes*, then you should consider an imaging setup that would provide a field of view of $\sim 1350 \times 1350 \times 1350$ μm^3. The relative standard deviation threshold and the minimum number of subvolumes are set by default to 5% and 4, respectively, in the module and can be modified (cf. Fig. VI-7). RVE size relation with particle size is discussed in [43].

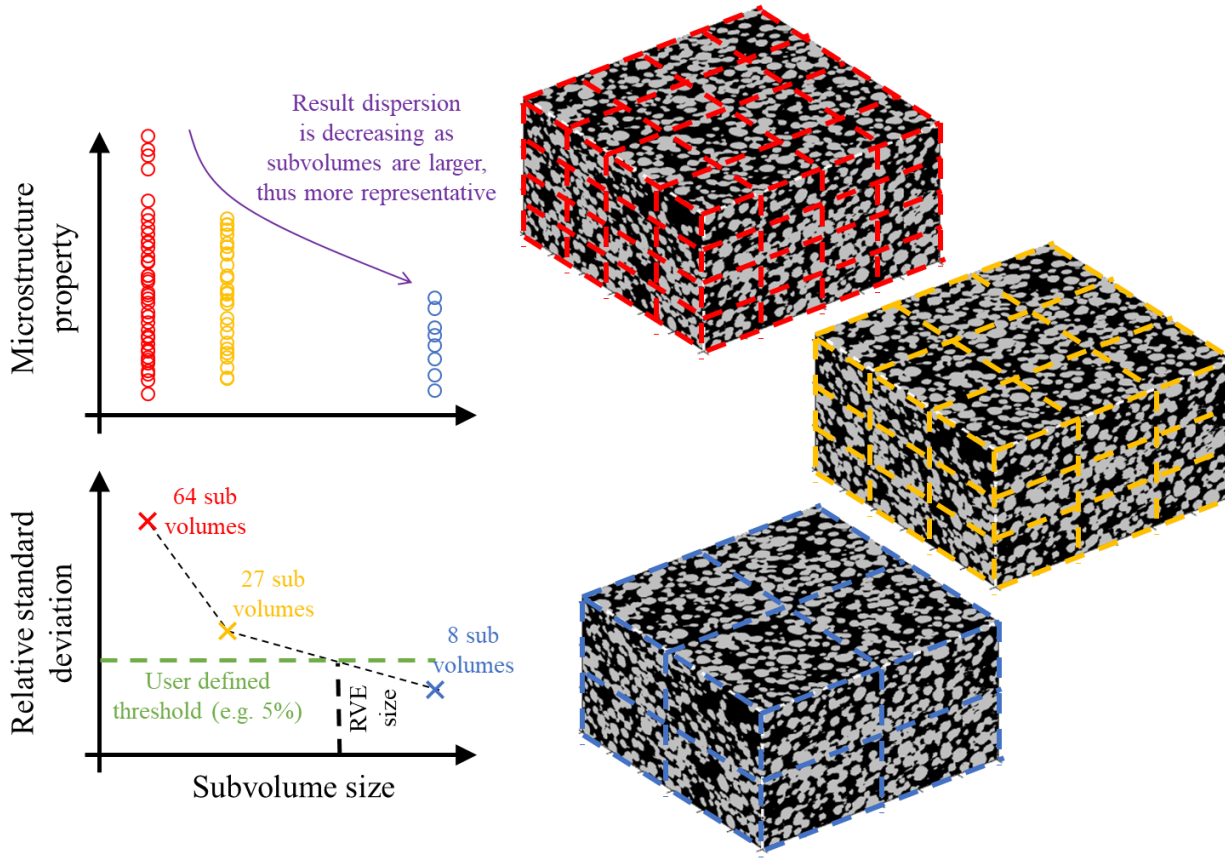


Figure VI-5. Representative volume analysis method. Subvolume aspect can vary.

ii. RVE aspect ratio

Representative volume element (RVE) size is property-dependent[43,44]: each property has its own RVE. Correlated (non-independent) properties share the same RVE[43]. Therefore, the RVE size of a microstructure is the maximum RVE size among all the properties relevant for the application. RVE size is often expressed as x µm^3, implicitly neglecting the volume aspect ratio. Instead RVE size should be expressed as $a \times b \times c$ µm^3, with a, b, and c the dimensions of the RVE volume. Percolation-related properties, such as connectivity and tortuosity factor, are strongly dependent of the volume aspect ratio. This can be illustrated considering the extreme case of a match-like volume $x \times y \times y$ µm^3 with y being the voxel length (i.e., a line with a section equal to the voxel section, $x \gg y$). Such volume has no connectivity and thus an infinite tortuosity factor. The choice of the subvolume aspect ratio then impacts not only the result of each subvolume, but also the RVE size. The toolbox offers three choice of RVE aspect ratio (cf. Fig. VI-6): (A) keeping the initial aspect ratio of the whole volume, (B) setting directly an aspect ratio for the subvolumes and (C) setting one constant dimension. These three cases are illustrated in Figure VI-7. The question is then what aspect ratio is relevant? The whole volume aspect ratio is constrained by the imaging technique and has no physical meaning. Instead, the choice of the aspect ratio for representative volume analysis is guided by application/material. For instance,

battery electrode materials are very thin in one dimension (from current collector to separator). This thickness is a characteristic of the battery as it will drive the concentration gradient along the electrode thickness, therefore modelers are interested in the ionic transport property from current collector to separator. Microstructure characterization will calculate this parameter (cf. §VI-9b), then determine the representative section area $x \times y$ µm^2 considering subvolumes of size $a \times b \times constant\ thickness$, i.e., using the Case C from the module. Note that you can stack several RVA in the module. The file Function_get_Subdomains.m in src \Microstructure_characterization determines subdomains bounds depending of the selected case (A-E).

The mean value calculated from the subvolumes is not necessarily equal to the value obtained from the whole volume. Equality is achieved only for volume averaged properties that do not suffer from any edge effects for which a simple rule of mixture approach is adequate, for instance volume fractions. Effective diffusion coefficients on the other hand can be averaged only if subvolumes are arranged in parallel, and if each of them are large enough not to be affected significantly by edge effect. Example can be found in [43].

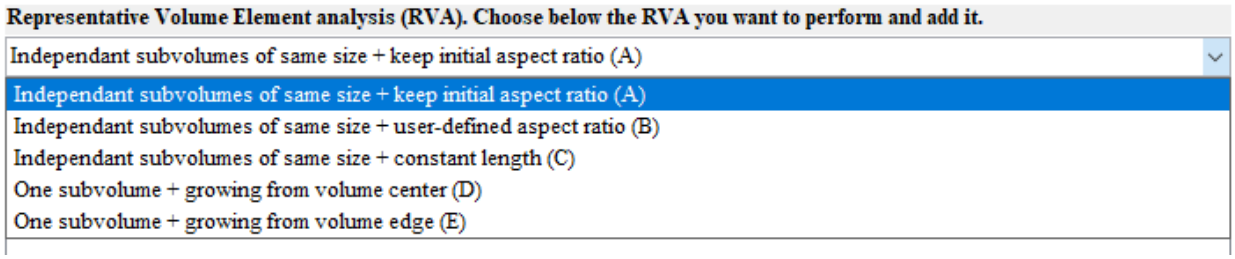


Figure VI-6. Subvolumes aspect ratio choice for representative volume analysis

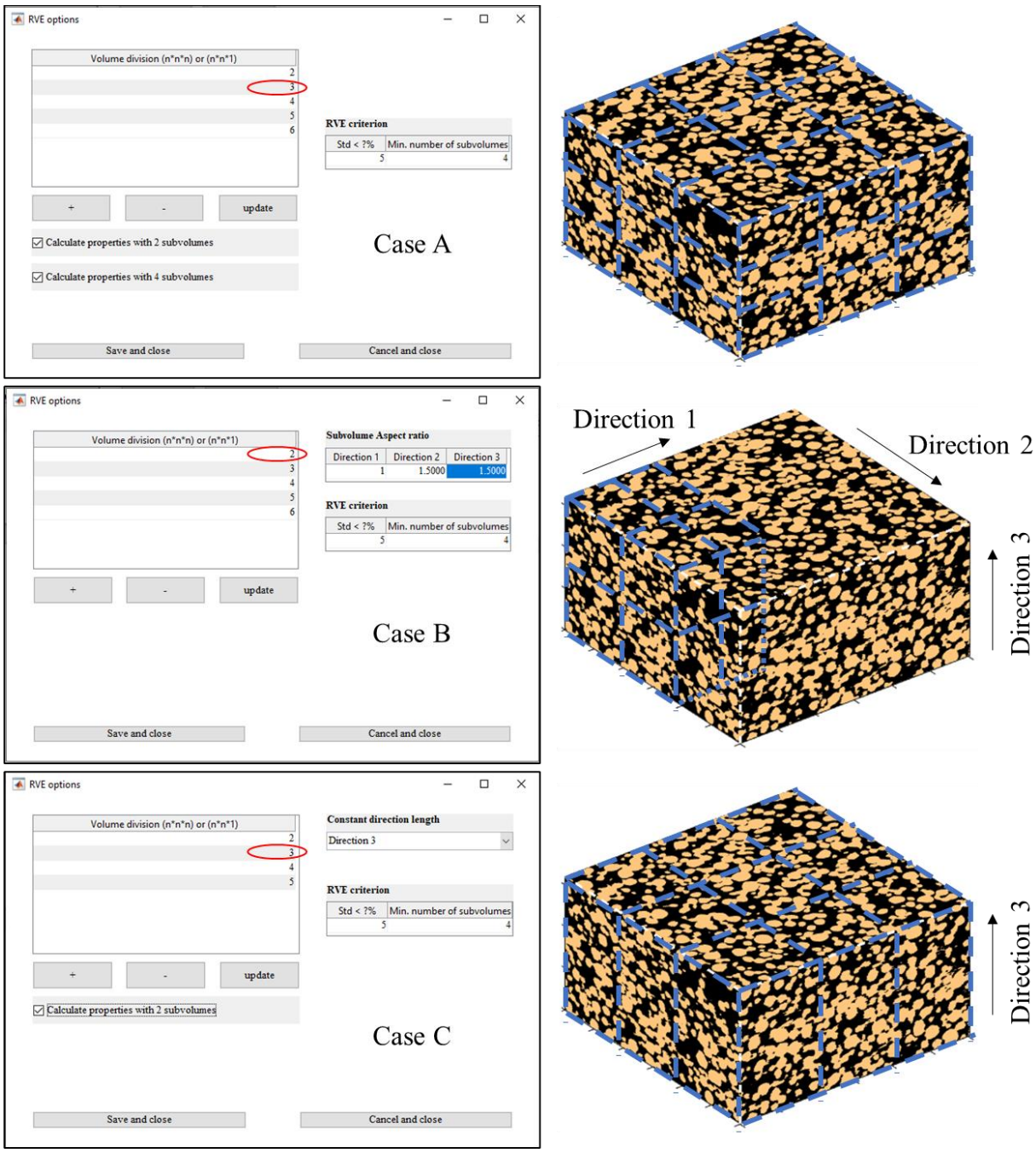


Figure VI-7. (Top) Case A: subvolumes and whole volume share identical aspect ratio. (Middle) Case B: subvolumes with custom aspect ratio 1:1.5:1.5. The largest subvolume that verifies this aspect ratio is first determined, and then cut into subvolumes. Not all the volume will be investigated. (Bottom) Case C: one direction dimension is fixed. (Right) the blue lines represent the subvolumes edges, for the volume division circled in red.

iii. RVE sensitivity with field of view

RVA result must always be considered with caution, especially if the RVE size has been found with a relatively low number of subvolumes. Being able to determine a RVE size can induced a false confidence in the results: RVA performed on the same material but with different field of view can result in different RVE sizes and properties results as illustrated in Figure VII-8 and 9. RVE size is expected to increase with the field of view, as larger field of views are more likely to include more different heterogeneities, until converging. In order to build more confidence in the RVE size determination, and thus in the result relevance, RVA can be performed on different subsets of the initial field of view, to verify if RVE size is stable with the field of view. An example of such analysis is shown in figure VI-9. Porosity RVE sizes of an NMC electrode volume from the NREL open-source library (NMC-1-CAL) have been calculated considering the whole volume, and several iterations of the same volume but cropped. As expected, the larger the volume for which RVE analysis is performed, the larger the RVE size is found (although, it should converge eventually).

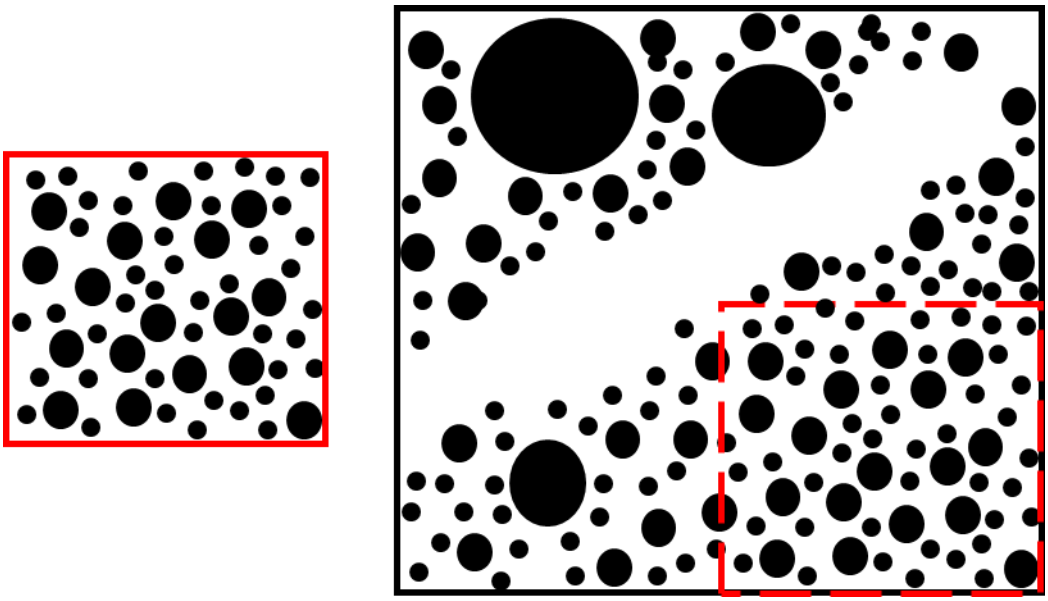


Figure VI-8. (Left) Microstructure analysis will conclude the material has a bi-modal particle size distribution and a small RVE size indicating a high uniformity (~random spatial distribution). (Right) Analysis performed on the same material but with an initial larger field of view will indicate a tri-modal particle size distribution, the presence of crack and will likely conclude the field of view is too small to determine a RVE.

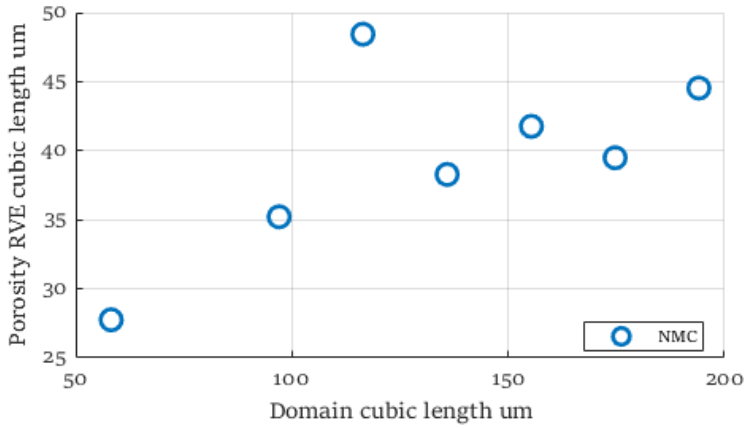


Figure VI-9. Porosity RVE size calculated on nested volumes (smaller volumes are included in the larger volumes). Figure plotted with the correlation module. Increasing trend is noticeable.

iv. Alternative approach (not recommended)

RVA is not uniquely defined in the literature. For the sake of completeness, two other approaches are included in the module, Case D and E (cf. Fig. VI-5 and 9). Case D consists in growing a unique subvolume from the field of view centroid. Case E consists in growing a unique subvolume modifying only one dimension, while the two others are fixed. The issue with these two approaches lies in the fact that they do not answer the representativity question as each subvolume has different size. Several options are available to control the volume increment at each new step, and it is recommended to use a percentage of the current subvolume so that the quantity dV/V0 = (V1-V0)/V0 (i.e., volume relative increment) is constant, with V1 the new subvolume and V0 the current subvolume. Property result is then plotted as function of the unique subvolume size. RVE could be determined setting a threshold on the slope. A constant subvolume increment dV implies that the quantity dV/V0 tends to 0 as V0 is increasing, which will *systematically* show a convergence of the property value with the subvolume size that can wrongly be interpreted as the signature of a RVE, as the slope will invariably tend to 0.

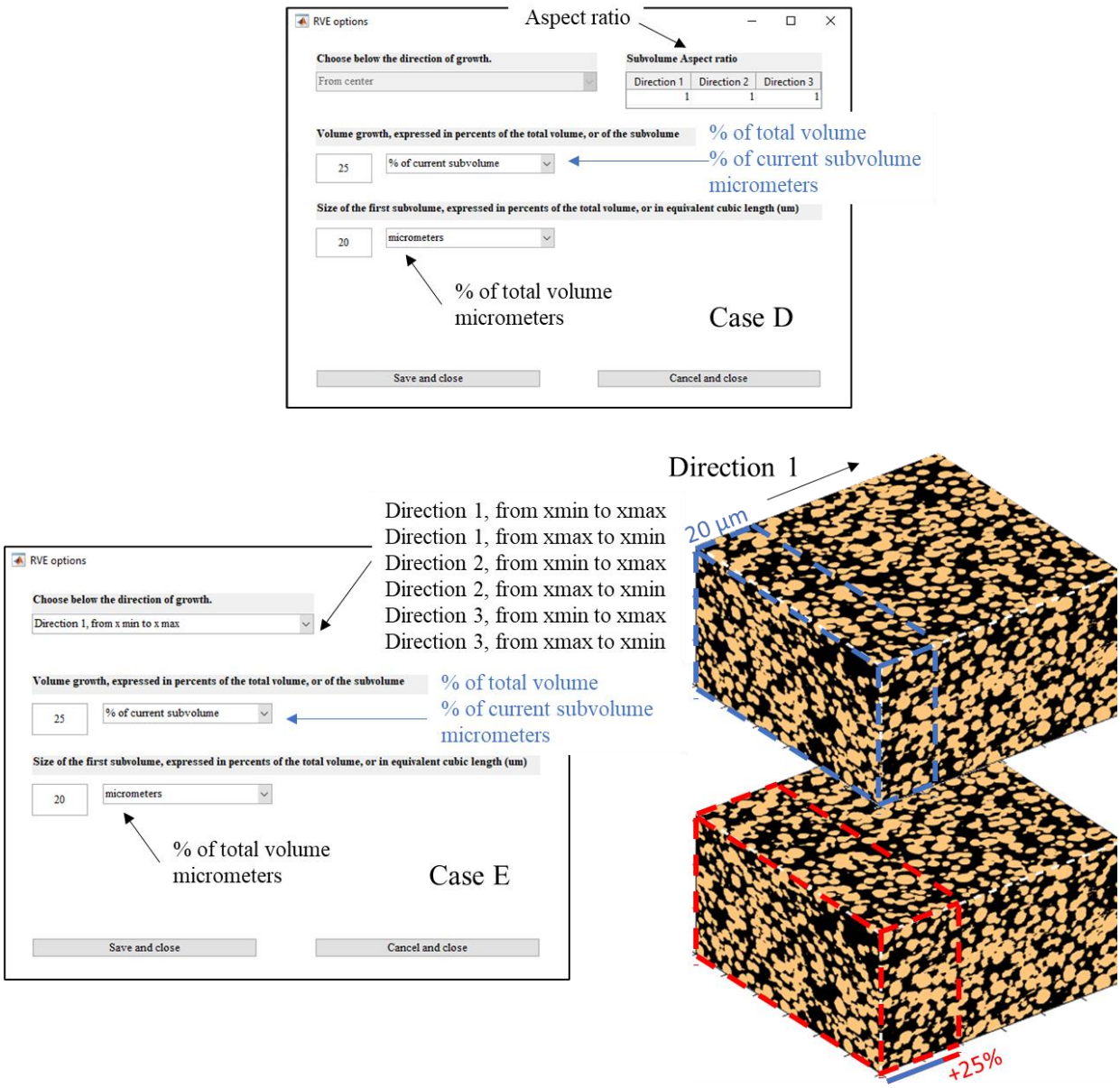


Figure VI-10. (Top) Case D: unique subvolume is growing from the field of view center. (Bottom) Case E: unique subvolume is growing from one field of view extremity. Arrows indicates the different choices offered by the popup menu.

The difficulty to evaluate with confidence the RVE size illustrates well that while it is relatively easy to calculate a microstructure property (one calculation performed on the whole field of view is enough) it is much more complicated and time consuming to estimate its error (multiple calculations are required). Although that is the price to enable quality analysis you can reasonably trust. Note that in literature, the term RVE is sometimes abusively used, confused with a control volume.

Depending on the algorithm used for their generation, numerically generated microstructures are not well suited for representative volume element analysis. For instance, the stochastic algorithm presented in §IV refreshes probabilities at each new particle generation to better match target values, thus artificially enforcing a high uniformity.

3. Module purpose and strength

The microstructure characterization module is the main one of the microstructure analysis toolbox. Toolbox development initially started with only this module and has been progressively extended towards other tasks (modules) subsequently. While other modules have limited ambitions (e.g., the filtering and segmentation module is far from exhaustive), this one aims to establish a new standard in microstructure characterization eventually. While only few microstructure characterization methods from the literature are implemented, the real added value of the module comes from its accuracy, systematic, and in-depth analysis, its user-friendliness, and its modular structure. These points are presented more in details below.

- Accuracy analysis.

Calculating a property value is only half the story. Saying that "this microstructure has a value of x for this property" is at the best incomplete and at the worst wrong. How could you be sure about the value you are giving if you do not perform a representative volume element (RVE) and a voxel size dependence analysis? Furthermore, are you certain the way you have performed RVE analysis is relevant, as different non-equivalent approaches exist (cf. §VI-2)? The method used to calculate a given property may suffer from edge effect effectively biasing the result (e.g., §VI-5b,c,f). In such case, you should provide an estimation of the edge effect to assess if its impact is marginal and can be safely ignored or if it should raise a red flag on the result. Furthermore, the property you are trying to calculate may be obtainable through different methods (cf. §VI-5f). These methods are not equivalent most of the time as they rely on different assumptions, that may be correct or incorrect depending on each microstructure. As in many other fields, it is relatively easy to provide a 'result' while estimating the level of confidence (i.e., error) of its value is much harder. This module, through its automated RVE and voxel size analysis, and thanks to its relatively large choice of algorithm to calculate a given property will help you evaluating the field of view induced error, image resolution induced error, and method related error.

- Systematic analysis.

The in-depth analysis described above can be applied to all the calculated parameters if the user chooses to do. Each parameter has their own voxel size dependence profile[17,34] (sometimes algorithms/methods that calculate the same property can even have a different voxel size dependence profile[17]) and representative volume element size[43,44]. Therefore, you should not rely on performing an RVE and voxel size dependence analysis on only one property, and then assume it determines the RVE size and voxel size for the investigated microstructure (i.e., extrapolating for all microstructure properties). Performing a manual RVE analysis is very time-consuming as it implies calculating the same property hundreds of times. Fortunately, the module automated it this task for you. Typically, such systematic analysis is recommended when you are dealing with a new type of microstructure and/or new algorithms you are not familiar with. In addition to reinforce confidence in your results, such systematic analysis allows you to find interesting findings, such as fractal behavior for a parameter with voxel size[42] and systematic hierarchy between RVE size of different properties[43].

- In-depth analysis.

Sometimes, providing a mean value with some upper and lower bounds for the confidence interval is not enough to adequately describe the complex microstructure. You may deal with a graded microstructure, for which the average result value should be complemented with a graded analysis, i.e. plotting the property value as a function of position. Some properties are better represented through a distribution function (e.g. particle size) or a standard deviation. Other properties can be derived from the initial calculation, for instance a particle identification algorithm could provide much more information than only the particle size (particle sphericity and elongation). The module automatically performs graded analysis and distribution analysis for relevant properties and calculate associated properties when adequate so that you can extract the most information from a given microstructure.

- User-friendliness.

Sometimes overlooked in open-source software development, user-friendliness is however a key aspect to hope reaching many users. Excellent software with poor interface is a recipe for a waste opportunity. The GUI should make the use of the toolbox easy. If something is not clear, please contact the author as one objective of the toolbox is to be as user-friendly as possible to streamline microstructure-related work.

- Modular structure.

The modular structure is valuable from the developer point of view as it facilitates code maintenance and future developments. Since the toolbox is open source, users also benefits from it, as they can adapt the toolbox to their needs with minimum coding, while having access to all the features of the module, such as RVE analysis (cf. §VI-7). Besides, it greatly helps code reviewing form third parties. Indeed, scientific algorithms length is kept short as all side tasks are deported in other files. The modularity structure is more detailed in §VI-6.

4. How to use

a. With the graphic user interface (standard use)

i. Import volumes

You can import a new volume in the first tab. The tab provides various information and allow you to setup some parameters (follow instructions):

- On the top left is displayed the name of the file loaded. A number is assigned ('Volume numero: X) to the current volume. The number is incremented with each new loaded volume.
- On the middle left is a slice view of the current loaded microstructure. You can change the view axis with the popup menu and explore slices with the slider.
- On the top right, you can enter the initial voxel size in nanometers, and the new voxel size in nanometers. If different, upscaling or downscaling will be applied later (note that this is not used for voxel size dependence analysis). Just below, you can choose your region of interest. If you need to perform a more complex ROI selection (e.g., rotation) you should use first the filtering/segmentation module. You can also write the direction names. These names will used to label figures and results adequatly. Below is a non-interactive text section that summarizes the size of the loaded volume and of the volume that will be actually investigated.
- On the middle right, you can enter phase information (number of phases is not limited). Volume fraction is showed for each phase. You can change the color (RGB) that will modify the representation on the left, but most importantly will be used for every figure generated by the module. You can also assign a name to the phases, that will be used also to label results. Lastly, you can assign a code to the phase. You can assign different phases with the same 'assign code' value. In such case, these phases will be merged. The newly defined phase will use the color of the first merged phase and its name will be a concatenation of all the merged phases, separated by 'and' (e.g., 'Pore', 'Additives' will give 'Pore and Additives').
- On the bottom right, you can enter optional information about the volume. Table will be saved in an excel file and is used to keep track of the different samples/calculations. Just below, you can select the save folder (until you choose it you cannot validate your options) and write the result folder for this volume (save folder/result folder). By default, the result folder is filename_Vol_Xum_Voxel_Ynm, with X and Y, respectively, the equivalent cubic domain size and Y the new voxel size.

Once you have entered all the options, you can click on the button 'Once steps 1-5 done, click to save your choices' or on 'Reset selected options and current volume' to cancel your choice and reset the tab. The green or red text above indicates if the user entered incorrect information (e.g., text instead of number, negative voxel size, out of bound ROI, etc.). You can repeat the import process (the 'Volume numero: X' will be incremented) in case you want to perform a batch analysis, for which volumes will be characterized sequentially.

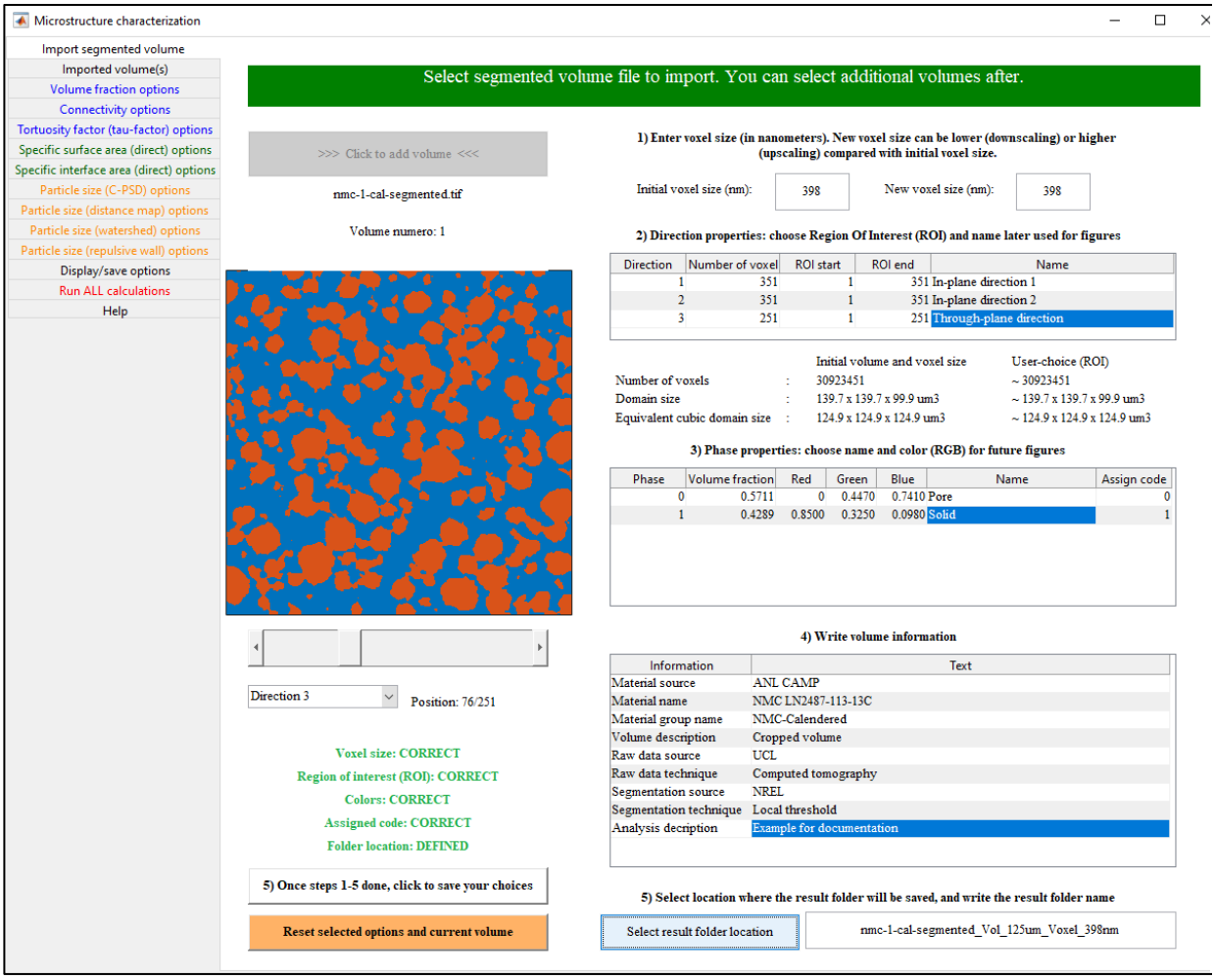


Figure VI-11. Import tab.

The second tab shows all the volume you have loaded in the first tab. You can reorganize them (modifying the order of calculation value) or remove them (setting order of calculation equal to 0). Once you have done your modifications, if any, you must click on the button at the bottom to update the order of calculation.

ii. Choose microstructure properties to calculate

The following tabs allow you to setup microstructure properties. There is one tab per method, meaning that microstructure properties for which several methods exist for their calculation (e.g. particle size) have several tabs. Choices are shared for all microstructure you have loaded. Properties are calculated for every phase.

The top half is common for each tab:

- The checkbox at the top give you the choice to calculate or not the given microstructure property with this method

- On the left, you can set up (or not) voxel size dependence analysis. See §VI-2a.
- On the right, you can set up (or not) representative volume element analysis. See §VI-2b.

The bottom half of the tab is specific for each method and enables the user to choose its parameters. Parameters are explained in §VI-5 where microstructure properties are introduced.

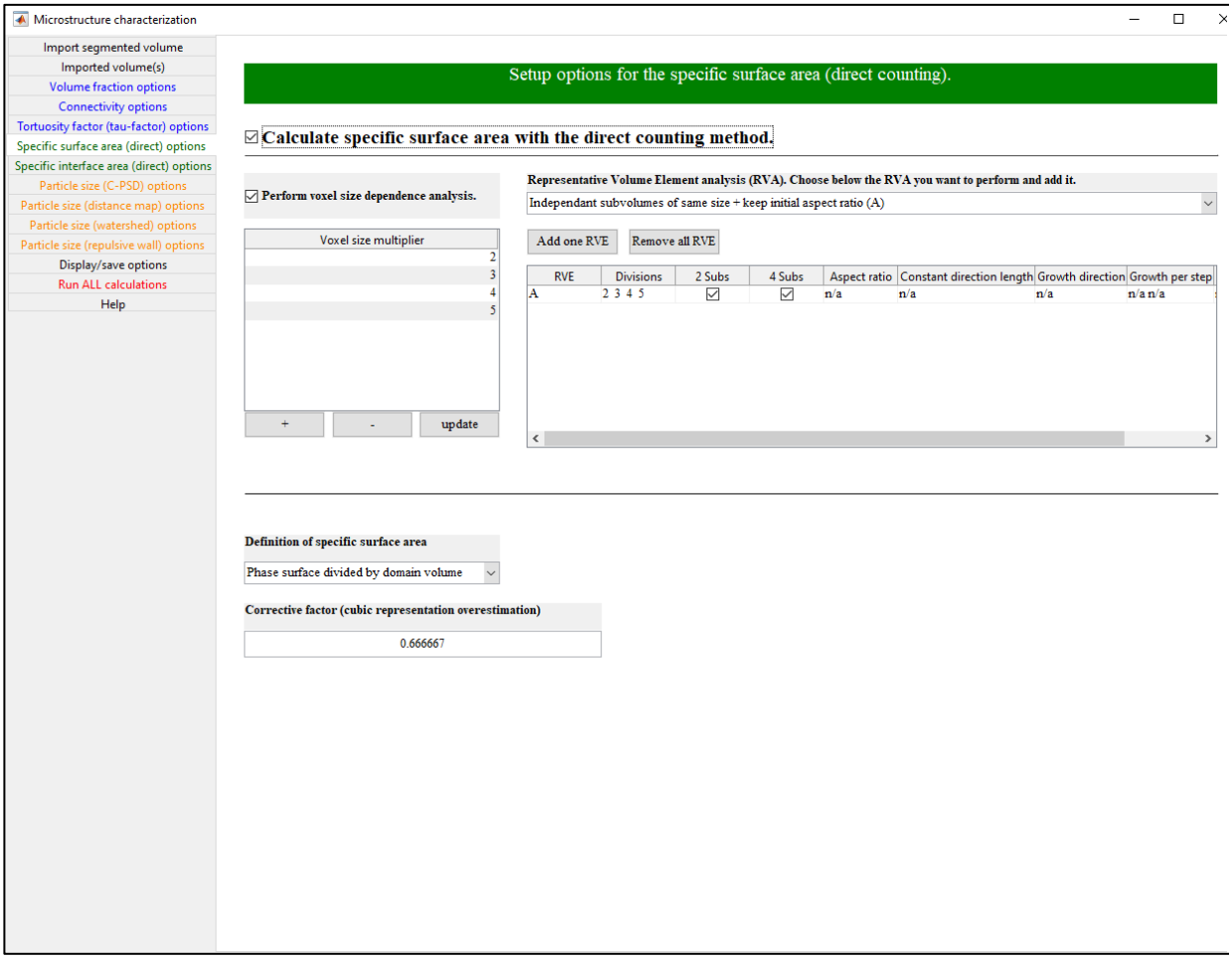


Figure VI-12. A method tab. Top half is common with others method tabs, while bottom half is specific for each method.

iii. Save options, and run calculations

In the 'Display/save options' tab, you can select various saving and format options for figures and results. By default, 'Close figure once displayed and/or saved' is checked to avoid saturating the user window with dozens, potentially hundreds of open figures. Once you have setup all your parameters, you can go to the 'Run ALL calculations' to characterize sequentially all the imported volumes.

b. Use algorithms as standalone functions from the command window

Alternatively, you can run algorithms as standalones if desired. Open function_microstructure_characterization_segmentedvolume.m (src/ Microstructure_characterization) and go to the section CALCULATIONS to see a list of all the functions used to calculate various microstructure parameters, e.g.:

```
%% PARTICLE SIZE (Continuum Particle-size distribution C-PSD)
if PROPERTY.particlesize_cpsd.todo
    Function_particle_size_CPSD(Phase_microstructure, PROPERTY, OPTIONS, INFO); % Call function
end
```

If you open one of them or read their help, you will see that two syntaxes are available:

```
function [] = Function_particle_size_CPSD(Phase_microstructure, PROPERTY, OPTIONS, INFO)
%Calculate Particle size with a spherical assumption (C-PSD)
% Function_particle_size_CPSD(array, PROPERTY, OPTIONS, INFO) - when use with the toolbox
% or
% Function_particle_size_CPSD(array, voxelsize) - when use as a standalone function
```

The first syntax, that uses structure variables PROPERTY, OPTIONS, and INFO is called when the GUI is used. The second syntax can be called directly from the command window from scratch. For example:

```
M=function_load_tif('C:\Users\fussegli\Desktop\nmc-1-cal-segmented.tif');
Function_particle_size_CPSD(M, 398); % Voxel size of 398 nm
```

Results are printed in the MATLAB command window and figure are generated, similarly when the function is used through the GUI. However, representative volume element analysis and voxel size dependence analysis are not performed when functions are used as standalone. Other calculations (e.g., distribution functions, graded analysis, etc.) are still performed although. Results are saved in a folder located in the desktop named 'Volume characterization_*' with * the current date, and organized in subfolders as done when function is called with the GUI.

c. Use raw algorithms without graphical results

Lastly, you can choose to run the method raw algorithm directly. This usage is mostly relevant for debugging/understanding. In the src/Microstructure_characterization folder, filter files that end with algorithm (i.e. *algorithm.m) to find them. For instance, Function_particle_size_CPSD_Algorithm.m

```
function [] = Function_particle_size_CPSD(Phase_microstructure, PROPERTY, OPTIONS, INFO)
[...]
for current_phase=1:1:number_phase % Loop over all phases
    code_tmp = INFO.phase(current_phase).code;
    binary_phase=zeros(Domain_size(1),Domain_size(2),Domain_size(3)); % Initialization
    binary_phase(Phase_microstructure == code_tmp) = 1;
    [Particle_size] = Function_particle_size_CPSD_Algorithm(binary_phase); % Call algorithm
```

You can run it from the command window as:

```
M=function_load_tif('C:\Users\fussegli\Desktop\nmc-1-cal-segmented.tif');
solid_phase_id = 1;
binary_phase = zeros(size(M));
binary_phase(M==solid_phase_id)=1;
[Particle_size] = Function_particle_size_CPSD_Algorithm(binary_phase);
```

The inputs/outputs can be different for other algorithm files. Although they typically all require a binary image as input (i.e., voxel=1 for investigated phase, otherwise 0) and use the least number of parameter possible to keep the file short and thus readable. In the example above, even though particle size is calculated, the voxel size is not provided as the algorithm do not require it (voxel size=1). Result are scaled (i.e., dimensionalized) after.

d. Link with other modules

i. Link with Microstructure and results visualization module

Pixel-wise defined microstructure properties, such as particle size, or cluster label, are saved in a structure named results_visualization, which is saved in the subfolder save folder/result folder /Visualization. The visualization module will automatically find all these structures (cf. §VII-3). The user can easily add new 3D array to be stored for future visualization (e.g. particle radius, or volume in addition to particle diameter) as explained in §VI-7c)

```
function [] = Function_particle_size_CPSD(Phase_microstructure, PROPERTY, OPTIONS, INFO)
[…]
for current_phase=1:1:number_phase % Loop over all phases
    results_visualization(current_phase).Particle_diameter_CPSD = Particle_size;
```

ii. Link with Properties correlation module

Dozens of microstructures, each with dozens of microstructure parameters can form a large set of parameters with hundreds of values, that will be tedious to manually copy one be one in a single file for analysis. Some results are saved in a structure named results_correlation, which is saved in the subfolder save folder/result folder/Correlation. By default, not only the property main result is saved (e.g., mean particle size) but also side results (e.g., particle side standard deviation, RVE size of particle size, etc.) allowing interesting correlation (e.g., RVE size as function of property heterogeneity, etc.). The correlation module will automatically find all these structures (cf. §VIII). The user can easily add new result to be stored for future correlation as explained in §VI-7e).

```
function [] = Function_particle_size_CPSD(Phase_microstructure, PROPERTY, OPTIONS, INFO)
[…]
for current_phase=1:1:number_phase % Loop over all phases
   results_correlation(current_phase).Particle_diameter_mean_CPSD = PSD_results(current_phase,2);
```

e. Results organization

Because numerous files are created for each method, a proper file management is required to keep things sorted and readable. Each result is usually saved in three formats when relevant: one excel .xlsx file, one MATLAB figure .fig file if reformatting if needed, and one .png file.

- **Save folder/result folder/** contains the excel file 'General_information.xls' that stores as indicated general information about the investigated volume: input/output locations, phase name and code, domain size and directions name, voxel size, module version number, user information, and start date. In addition, it contains some meta information about the volume (material name, source, etc., that correspond to the information entered in the tab import). The excel file 'Volume_setup.xls' stores parameters used to crop, up/down scale the original

volume and to re-assign phases. Both files are very useful when you have to re-open old results you may have forgotten the details.

- **Save folder/result folder/Visualization** and **save folder/result folder/Correlation** contain structure files used, respectively by the visualization and the correlation module.
- **Save folder/result folder/Volume** contains both the imported volume as it is, but also the volume that is investigated (i.e., after cropping, image resolution up/down scale, phase re-assignment). Volumes are saved in a tif format with an unsigned integer 8bits data type that used very little disk space. It means the input data are saved with the outputs to keep track of a calculation and being able to reproduce it easily.
- **Save folder/result folder/Summary** contains high-level information that summarizes essential results. Basic characterizations results are saved per phase in figures 'Results_of_phasename', and in a table (Table_summary_results.xls). Several .mat files, one per method (Results_propertyname_methodname.mat), contain more detailed result. Time.xls stored the total wall clock and CPU time used for the characterization, not discriminating tasks.

Property	Method	Mean	Min	Max	Std	Unit	Std_percents
Equivalent diameter	C-PSD	4.211	0.398	15.1	2.426	[um]	57.61
Particle level of details	C-PSD	0.9164	n/a	n/a	n/a	[]	n/a
Phase surface / phase volume	Face summation	0.5182	n/a	n/a	n/a	[um^-1]	n/a
Phase surface / domain volume	Face summation	0.296	n/a	n/a	n/a	[um^-1]	n/a
Tortuosity factor 1/2/3	Tau factor	1.459 / 1.444 / 1.57	n/a	n/a	n/a	[]	n/a
Bruggeman exponent 1/2/3	Tau factor	1.675 / 1.656 / 1.805	n/a	n/a	n/a	[]	n/a
Normalized Eff. Diff. Coeff. 1/2/3	Tau factor	0.3914 / 0.3954 / 0.3638	n/a	n/a	n/a	[]	n/a
Volume fraction	Voxel summation	0.5711	n/a	n/a	n/a	[]	n/a

Figure VI-13. Example of a summary figure for a phase.

- **Save folder/result folder/Propertyname_methodname** contains results specific for this microstructure property, calculated with this particular method. See §VI-5 for information specific for each method. Representative volume element analyses are saved in a subfolder (one per case).

5. Microstructure properties

The module has been built with battery electrode material characterization in mind, and therefore is focusing on quantifying microstructure properties used in battery modeling, namely porosity, connectivity, tortuosity factor, specific surface and interface area, and particle size. Additional properties, interesting to evaluate image quality (level of detail) and refine description of the microstructure (particle shape, domain topology) can also be calculated. Results are illustrated with microstructures from the NREL open-source microstructure library[13],or numerically generated with the microstructure generation module of this toolbox.

a. Volume fraction

Results are saved in **Save folder/result folder/Volume_fraction.** Calculation is realized with Function_Volume_fractions.m.

Volume fraction ε_k of phase k is defined as the number of voxels v that belong to the phase k divided by the total number of voxels N of the domain.

$$\varepsilon_k = \frac{1}{N}\sum_{i=1}^{N} v(i) \quad with \quad v(i) = \begin{cases} 1 \; if \; v(i) \in phase \; k \\ 0 \; otherwise \end{cases} \qquad \text{[VI-v1]}$$

```
voxel_number = numel(Phase_microstructure);
for current_phase=1:1:number_phase % Loop over all phases
    code_tmp = INFO.phase(current_phase).code;
    Numbervoxel_phase(current_phase,1)= sum(sum(sum(Phase_microstructure==code_tmp)));
    Volumefraction_phase(current_phase,1) = Numbervoxel_phase(current_phase,1)/voxel_number;
    results_correlation(current_phase).volume_fraction = Volumefraction_phase(current_phase,1);
end
```

Results are organized as:

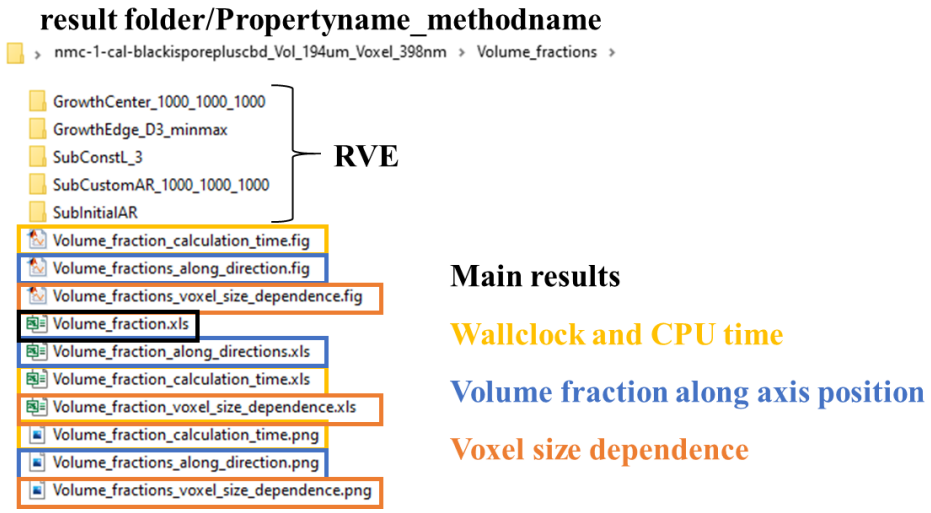


Figure VI-14a. Volume fractions result files organization. RVE subfolders are named accordingly to their case (cf. §VI-2b), from top to bottom: one subvolume growth from center with an aspect ratio of 1:1:1 (case D), one subvolume growth from edge along direction 3, from minimum to maximum (case E), independent subvolumes of same size with constant length for 3rd axis (case C), independent subvolumes of same size with a custom aspect ratio of 1:1:1 (case B), and independent subvolumes of same size with initial aspect ratio (case A).

Volume fractions slice per slice are also calculated slice per slice and plotted.

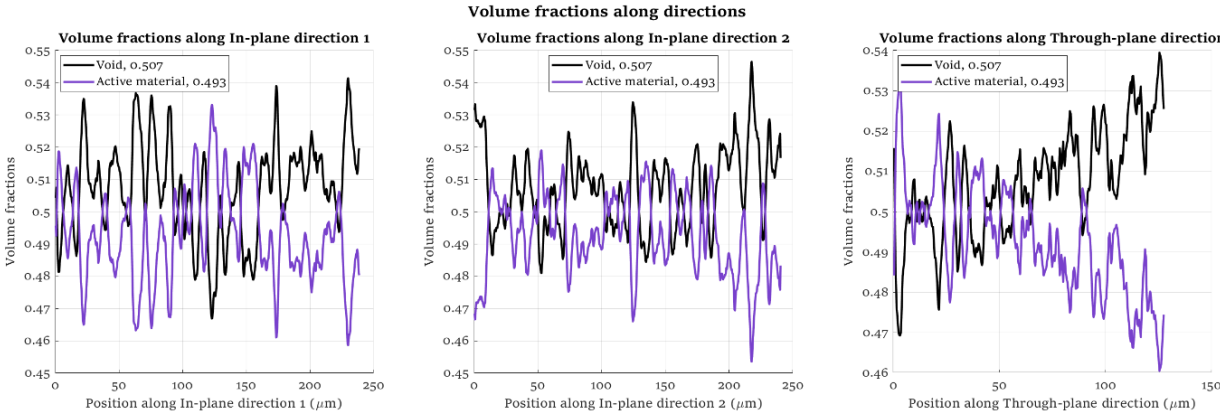


Figure VI-14b. Example of a volume fractions plotted as a function of positions. Legend, line color, titles, and x-labels are those defined by the user in the import tab.

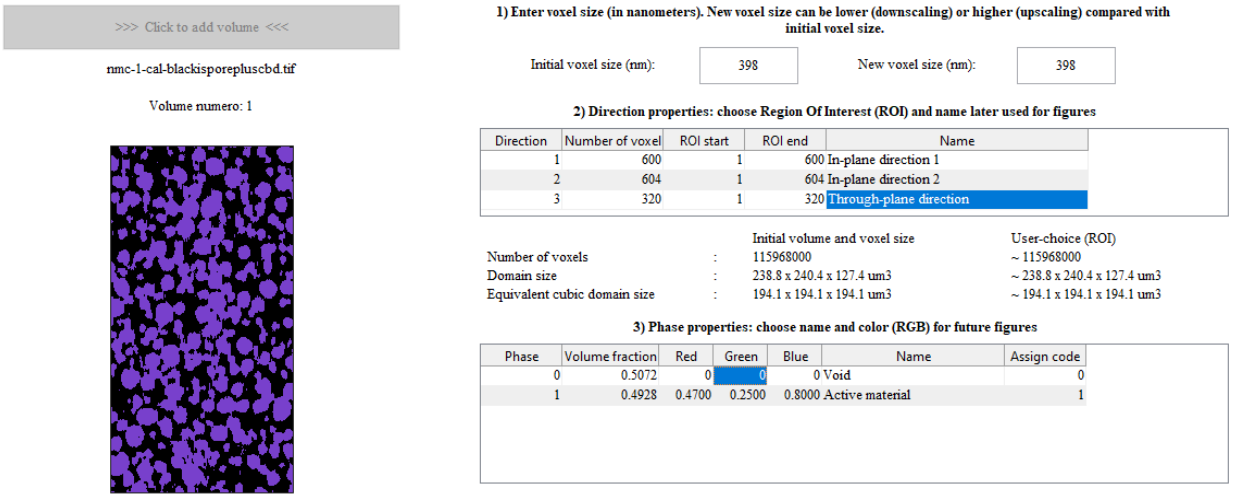


Figure VI-14c. Legend, line color, titles, and x-labels are those defined by the user in the import tab.

If a voxel size dependence analysis is asked by the user, then result is plotted:

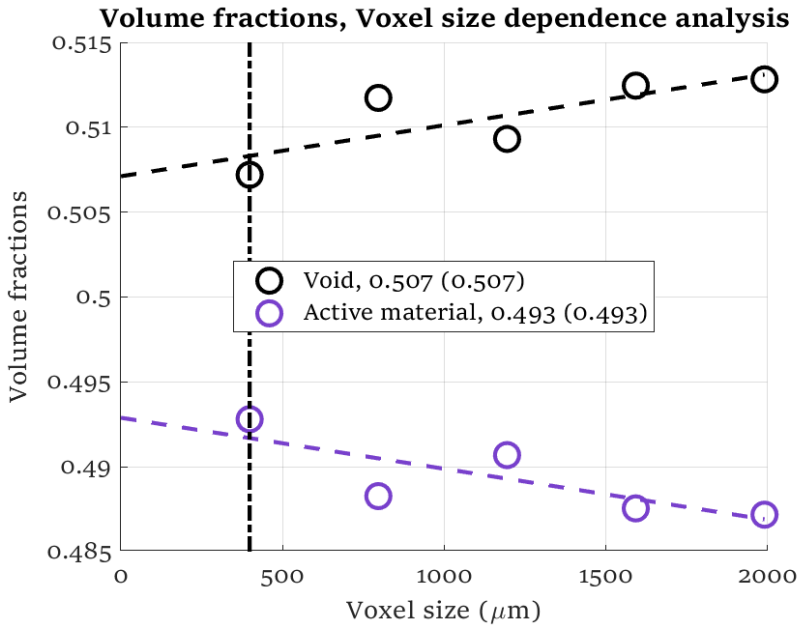


Figure VI-14d. Voxel size dependence calculated for volume fraction. Vertical dashed black line represents the initial image resolution. Dashed lines are (by default) linear interpolation. Values between parenthesis is the value extrapolated for a resolution of 0 nm. Because volume fraction is a volume averaged property, voxel size has a very limited impact on the result.

If one or several representative volume element analyses are asked by the user, results are plotted:

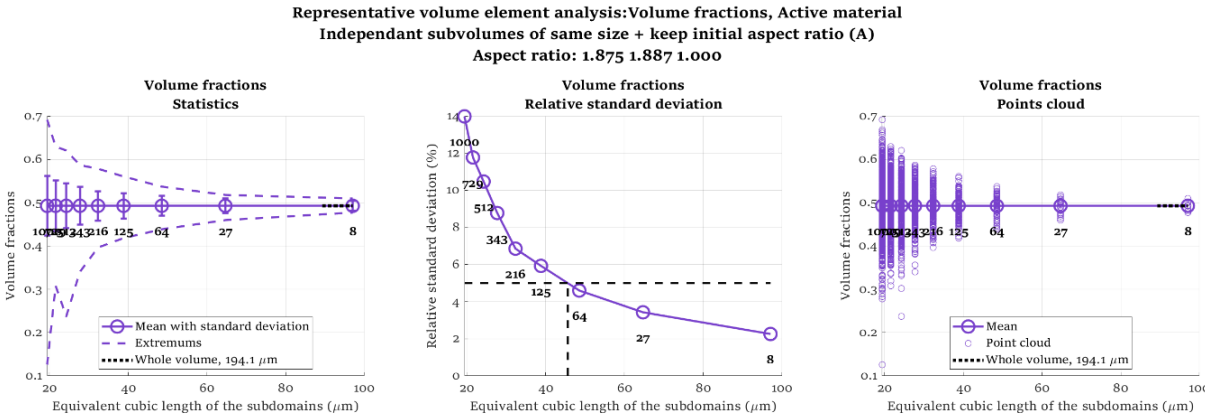


Figure VI-14e. Representative volume element analysis performed for volume fraction of the active material (same analysis is performed for the other phases). (Left) means, extremums, and standard deviations. The short-dashed line at the right indicates the value calculated with the whole volume (also indicated in the legend). Numbers next to values indicate the number of subvolumes. Note that mean values are equal with the one calculated with the whole volume. (Center) relative standard deviation. Vertical dashed line indicates the RVE size for the user-defined threshold (by default set to 5%). (Right) results for each subvolumes.

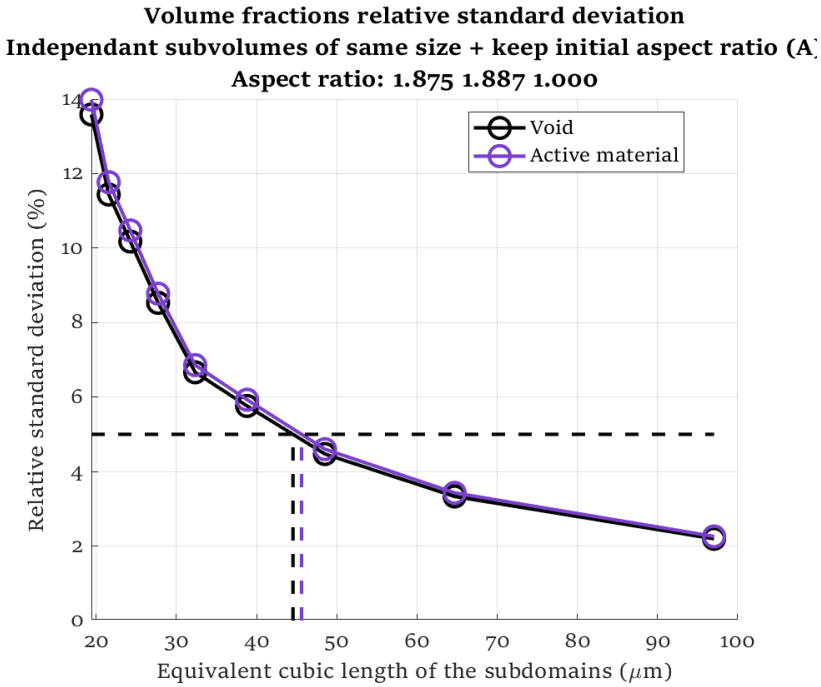


Figure VI-14f. Relative standard deviations are plotted for each phase. If RVE is case C (one dimension is set constant) then figure VI-13e and f are reproduced with both the subdomain cubic root and the quadratic root of the subdomain 2D field of view for the x-axis.

RVE analyses performed with Cases D or E (one unique subvolume) are different:

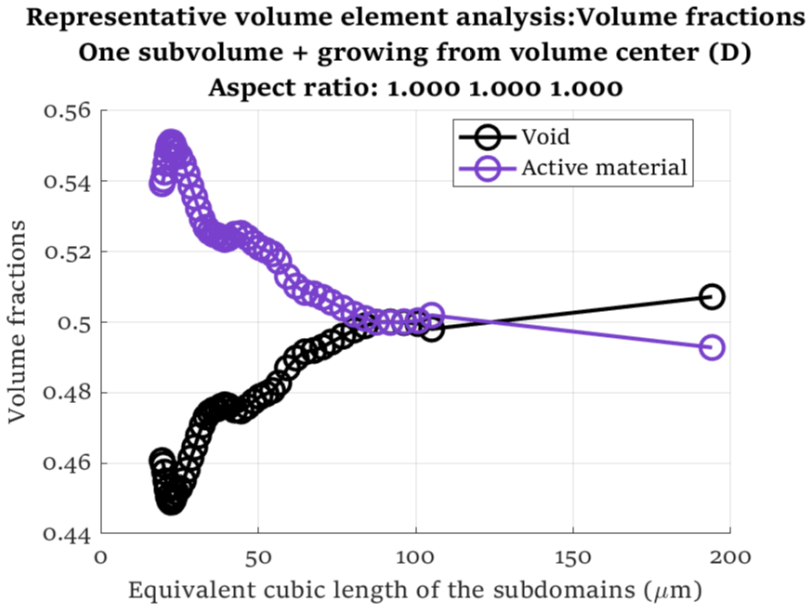


Figure VI-14g. Representative volume element analysis with one subvolume growth from center with an aspect ratio of 1:1:1 (case D). Because the whole volume has a different aspect

ratio with the one choose for the subvolume, a gap exists between the whole volume size and the largest subvolume with the custom aspect ratio.

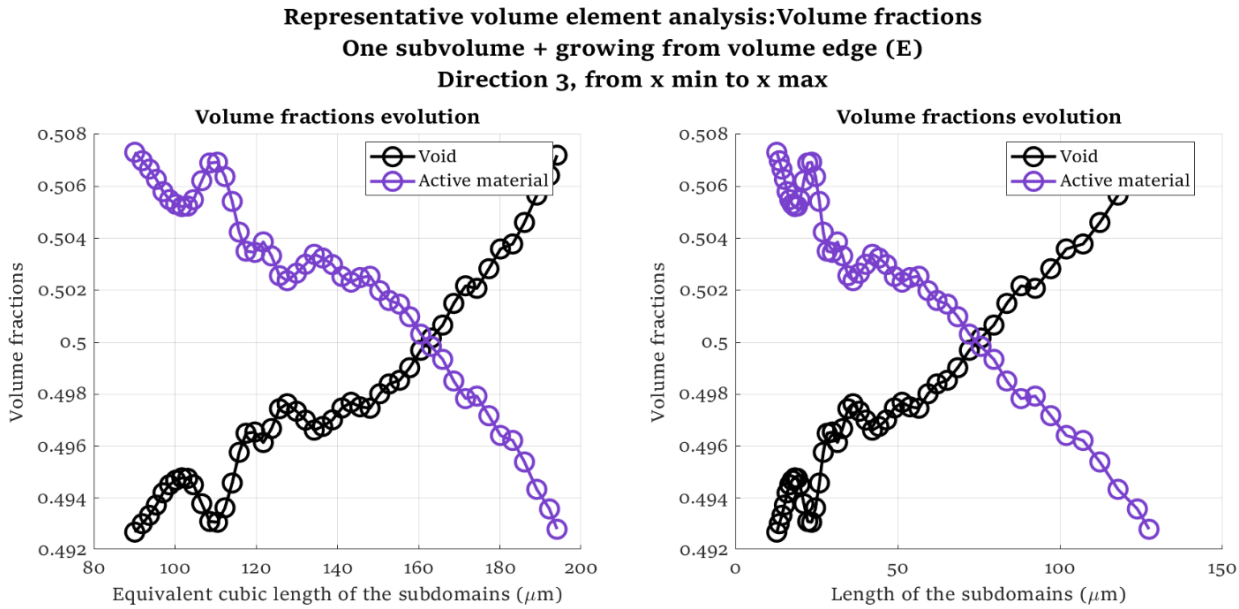


Figure VI-14h. Representative volume element analysis with one subvolume growth from volume edge along direction 3 constant (case E). (Left) x-axis is the cubic root of the subvolume size, (right) x-axis is direction 3 growing length.

b. Connectivity

Results are saved in **Save folder/result folder/Connectivity**. Connectivities are handled with the files Function_connectivity.m and Function_Connectivity_Algorithm.m.

Two voxels are considered connected if they share one common face (edges and vertices connections are ignored). A group of connected voxels is named a cluster. Cluster volumes are expressed in percents of the investigated phase. Connectivity (or percolation) has different definition, illustrated in figure VI-15a, and detailed below:

- An **isotropic definition**[30,43], for which three connectivities are determined per phase: (i) the **largest connected cluster**, (ii) the sum of all **isolated clusters**, and (iii) the sum of all **unknown clusters**. Isolated clusters are not connected with the largest connected cluster and are not located at the domain's edge. Unknown clusters are not connected with the largest connected cluster and are located at the domain's edge so that it is unknow if they are actually connected with the largest connected cluster or not (a larger field of view would have been required). Unknow cluster is an edge effect. A large volume of unknow cluster indicates the field of view is too small for the analysis.
- A **directional definition**, for which, for each direction three connectivities are determined per phase: (i) the sum of all clusters that are connected **from the first slice to the last slice normal with the investigated direction,** (ii) the sum of all isolated clusters that are not connected with

such connected clusters and that are not located at the domain's colinear edges, and (iii) the sum of all unknown clusters that are not connected with such connected clusters and that are located at the domain's colinear edges. The oriented definition suits better connectivity analysis of microstructure for which species transport are expected to diffuse through a particular direction – along the whole material thickness (such as the pore domain of battery electrodes).

- A **directional and oriented definition**, for which, for each direction six connectivities are determined per phase: (i) the sum of all clusters that are connected **to the first (respectively, last) slice normal with the investigated direction,** (ii) the sum of all isolated clusters that are not connected with such connected clusters and that are not located at the domain's colinear edges, and (iii) the sum of all unknown clusters that are not connected with such connected clusters and that are located at the domain's colinear edges. This connectivity is relevant for electronic percolation for instance, for which particles to be connected only need to be in contact with the current collector (and not necessarily with the opposite interface, i.e., the separator).
- Clusters that contain only one voxel are considered Ill-defined. Ill-defined clusters indicate either or both (i) segmentation produced noise and (ii) image resolution is too low.

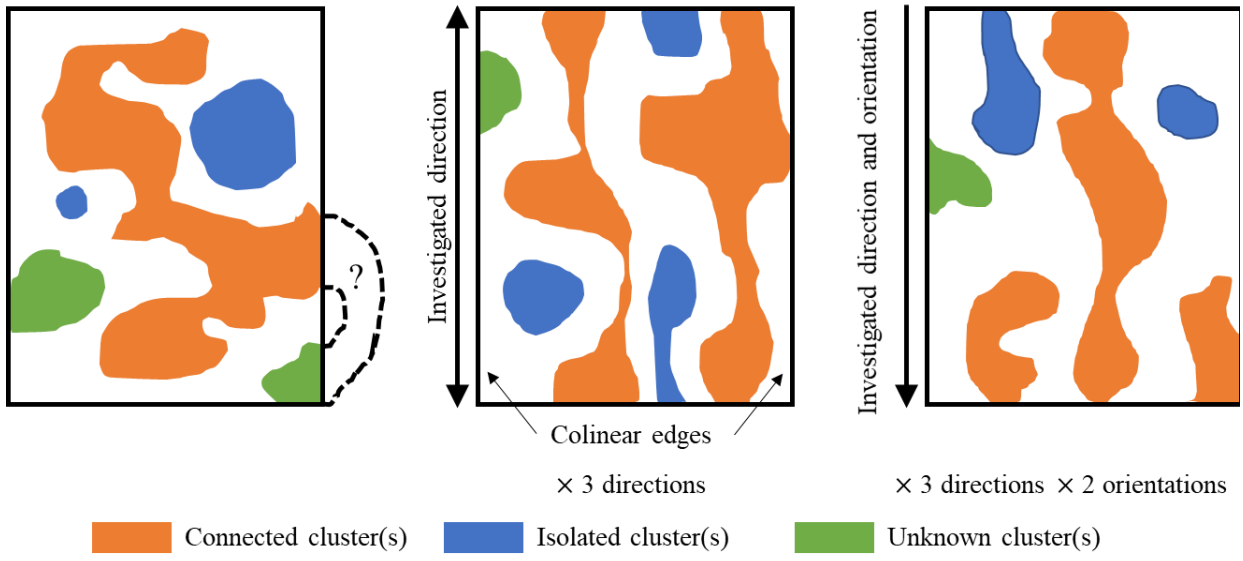


Figure VI-15a. (Left) Isotropic connectivity. There is only one connected cluster (the largest, whatever it is in contact with domain's edge). (Center) Directional connectivity, illustrated along the vertical direction. (Right) Directional and oriented connectivity, illustrated along the vertical direction, for the bottom edge.

The built-in MATLAB functions *bwlabeln* is used to calculate connectivity with a face-to-face only connectivity (edges and vertices are ignored to establish connectivity).

```
Clusters = bwlabeln(binary_array,6); % connectivty face-2-face
```

User can, in addition to calculate connectivity of the whole volume, calculate connectivity section per section (independently) using bottom half part of the 'Connectivity options' tab. Such section per section analysis is valuable, for instance, to investigate bilayer or multilayer microstructures, for which each layer may have different properties.

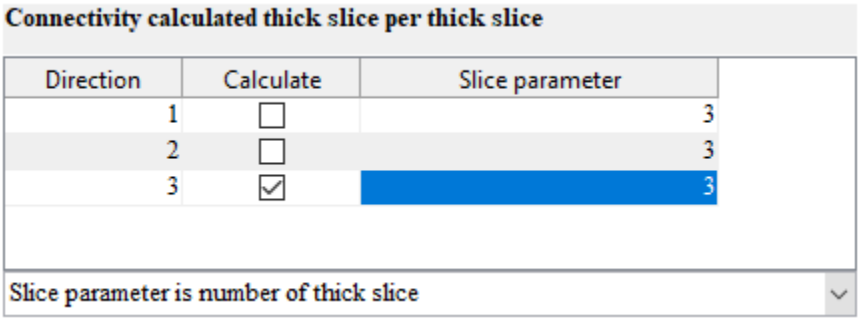


Figure VI-15a. Options specific to connectivity. In this example, volume is cut in three independent subvolumes along direction 3 and connectivity is calculated for each of them separately.

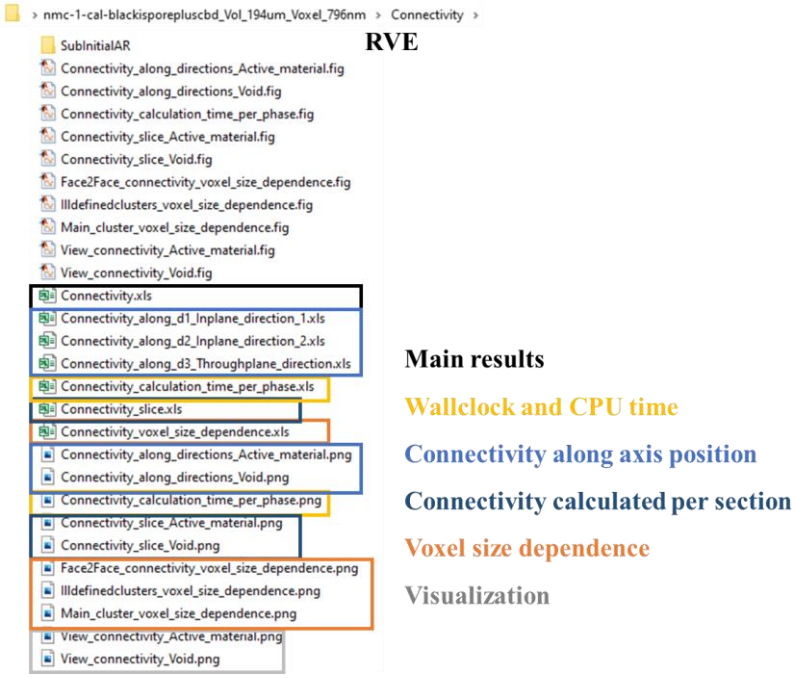


Figure VI-15b. Connectivity result files organization.

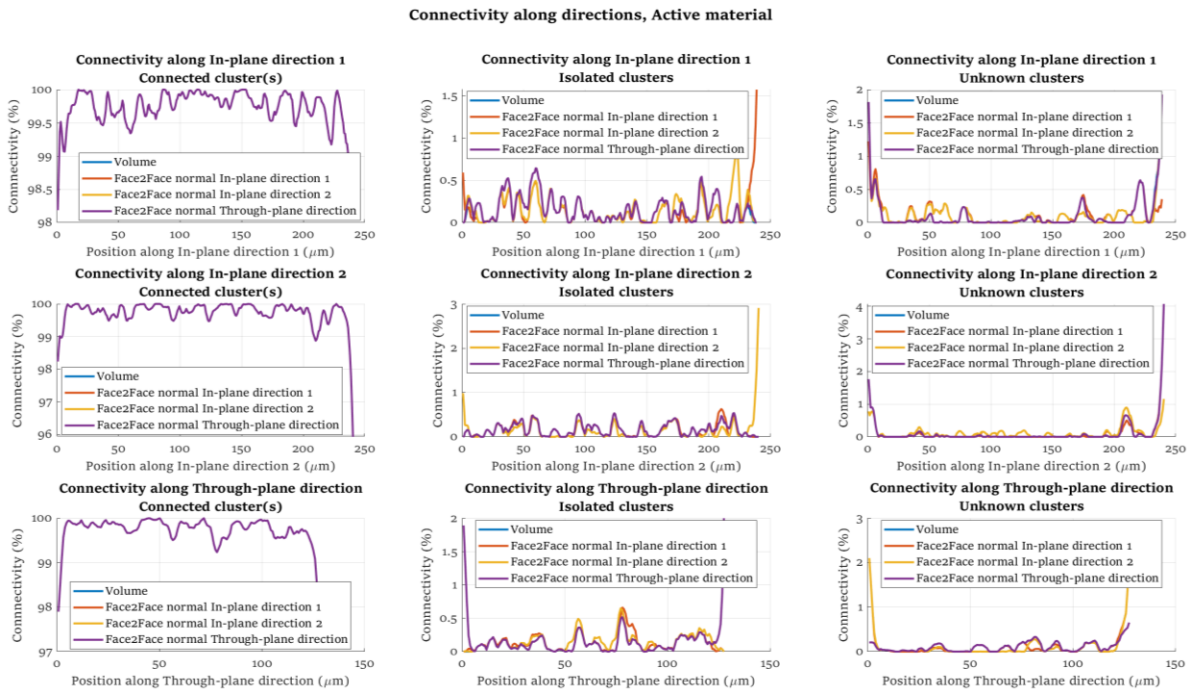


Figure VI-15c. Connectivity along directions (slice per slice). Blue lines correspond to the isotropic definition, while red, orange, and magenta lines correspond to oriented definition.

Clusters are labeled according to their connectivity and are plotted. Cluster connectivity can be explored slice per slice more in details with the visualization module.

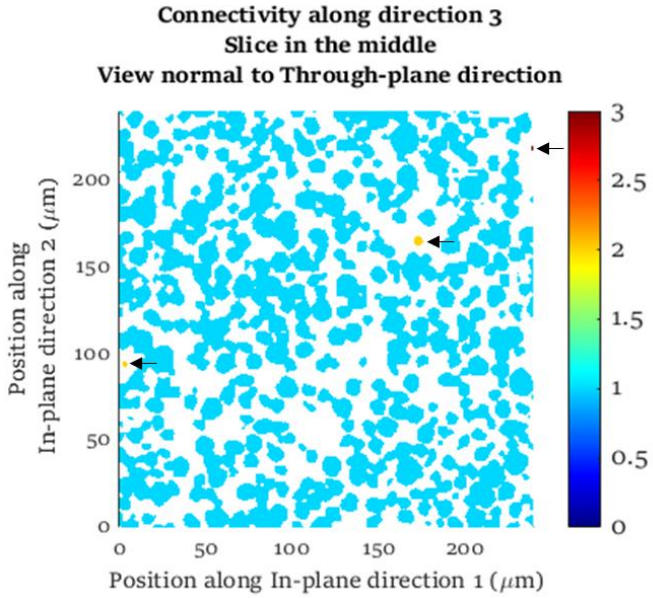


Figure VI-15d. Cluster oriented connectivity (0=background, 1=connected clusters along direction 3 2=isolated clusters, 3=unknown clusters). Arrows have been added to the graph to

pinpoint the very few isolated and unknown clusters. There are figures for each different connectivity.

If the user asks for section per section connectivity:

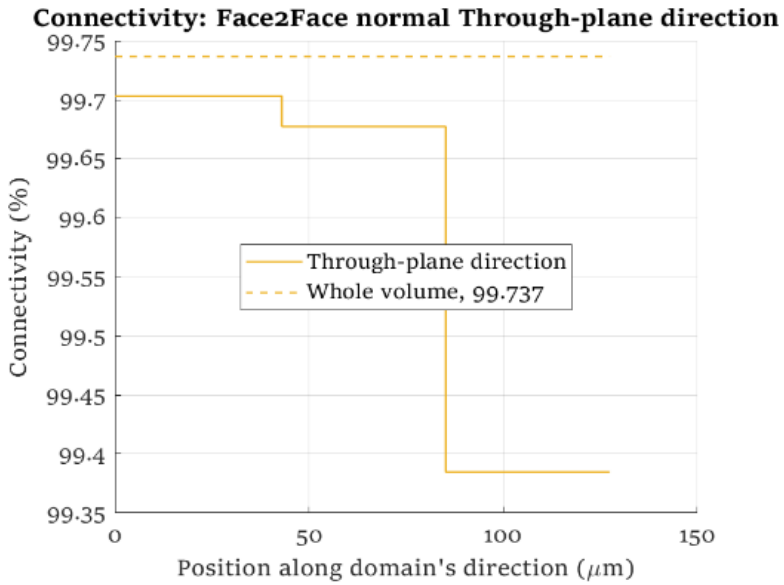


Figure VI-15e. Connectivity calculated per section (here 3). There are figures for each different connectivity.

In Figure VI-15c, connectivity is calculated within the whole volume, and voxel connectivity is plotted as function of its location. In Figure VI-15e, connectivity is calculated for several subvolumes independently (thick sections along a direction), and the overall connectivity is displayed for each of them.

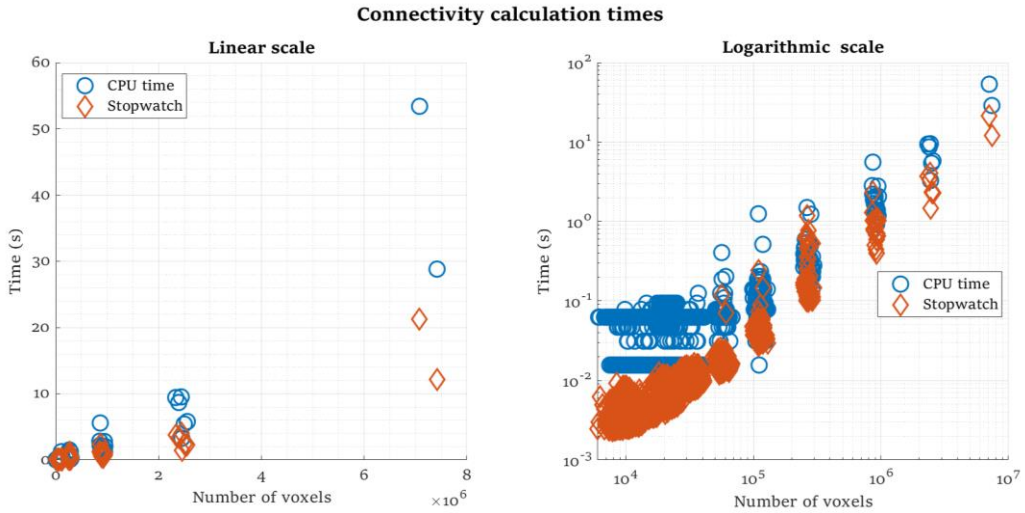


Figure VI-15f. Calculation times are also plotted, to evaluate scalability and future calculation times. Because RVA and voxel size dependence analysis were performed, there are many different points.

If the user requested a voxel size dependence analysis, then connected clusters connectivity (all definitions) are plotted as a function of voxel size. In addition, ill-defined cluster volume fraction, relative to the investigated phase, is also calculated, as it provides information about the quality of the image resolution.

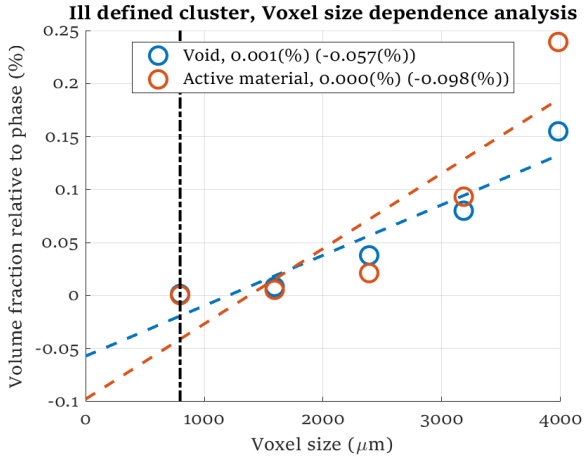


Figure VI-15g. Voxel size dependence analysis performed for ill-defined cluster. Note the increasing trend indicates more ill-defined (i.e., one-voxel size) clusters appear with coarser image resolution, as expected.

If one or several representative volume element analyses were requested, results are plotted for each connectivity.

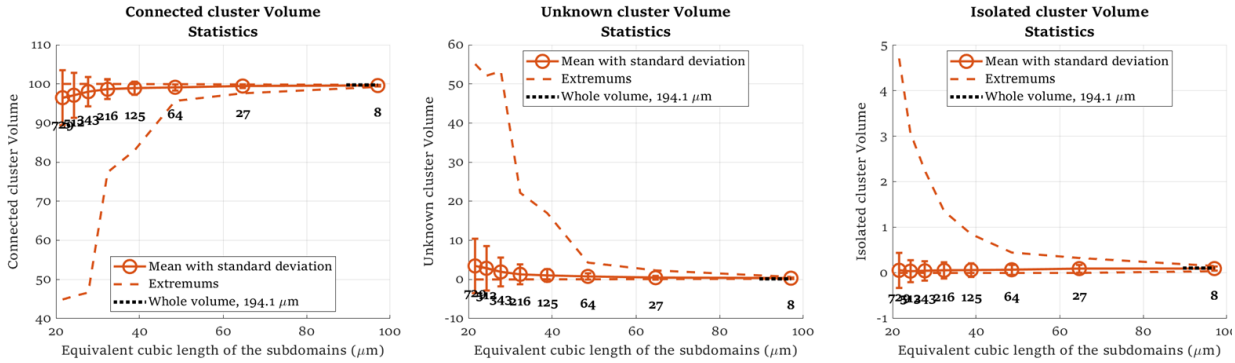


Figure VI-15h. Representative volume element analysis performed for (left) largest connected cluster, (center) unknown clusters and (right) isolated clusters. Note that mean values are not equal with the one calculated with the whole volume. This is due to the increasing edge effects (quantified through the unknown cluster) that decrease cluster connectivity, as seen in [43].

c. Tortuosity factor

Results are saved in **Save folder/result folder/ Tortuosity_factor_taufactor**. Tortuosity factors are handled with the file Function_Tortuosity_factor_taufactor.m.

The tortuosity factor τ_i (adimensional) denotes the effect of the convoluted, tortuous, path of the phase that hinders the species diffusion or conduction along the direction i. In battery modeling, tortuosity almost always refers to the through-plane (i.e., along electrode thickness) pore tortuosity to represent the lithium ion diffusion within the electrolyte (i.e. within the pore domain) from the electrode-current collector interface to the electrode-separator interface. The two other diffusion directions, along the in-plane dimensions, are not considered in 1D and pseudo 2D battery model as electrolyte ionic transport is 1D. However, for mesoscale 3D models all tortuosity factors (in-planes and through-plane) are considered.

Tortuosity factor is deduced from the normalized effective diffusion coefficient (defined as the ratio between the effective diffusion coefficient $D_{eff,i}$ along the direction i and the bulk, dense, diffusion coefficient D_{bulk}) and the porosity ε (cf. eq. VI-t1). Its value can be obtained through a homogenization calculation performed on a three-dimensional volume reconstruction of the microstructure[2,17,40,45]. The method consists in solving the Laplace equation ($\nabla^2 C = 0$, i.e., stationary heat equation with a unit diffusion coefficient) within the connected network, either with finite element method[40,43] of finite difference method[2], with a particular set of boundaries condition applied on the two domain's extremity faces normal with the investigated direction i. Then, the effective diffusion coefficient is deduced from the analysis of the calculated concentration/temperature field using the 1D Fick's first law as detailed in[40]. The method suffers from edge effect due to loss of percolation at the domain's edges (quantified through the number of unknown voxels, cf. §VI-5b), and exhibits a sensitivity with the choice of the boundary conditions[40,43,44]. Both effects are vanishing with larger volumes.

The homogenization calculation is an indirect determination of the tortuosity factor, as it computes first the normalized effective diffusion coefficient and then only deduces the tortuosity factor using equation VI-t1. More direct, geometric, determinations exist but rely on morphology and topology metrics more difficult to characterize[34,46], some of them presented in §VI-5g,h.

Tortuosity factor is usually related with the porosity through the Bruggeman's relationship[47,48] with p the Bruggeman exponent (cf. eq. VI-t2) and with its empirical counterpart, the generalized Archie's relationship[49,50] (cf. eq. VI-t3). The Bruggeman exponent p value is analytically known for ideal particle shape, under certain conditions that limits its application range (cf. eq. VI-t2a), otherwise its value is deduced from the tortuosity factor (cf. eq. VI-t2b). The analytical Bruggeman's law (cf. eq. VI-t2a) has been extensively used in the past, although with the combined advance of 3D microstructure-scale imaging and CPU, numerical determination of the tortuosity factor has revealed it provides a strong underestimation of the actual tortuosity factor for low porosity lithium ion battery electrodes[17,51], explained by its limited range of validity hardly applicable to real microstructures[48]. Nowadays, the most standard relationship used, in the

battery community, is the one proposed by Thorat[50] (cf. eq. VII-t3). To establish it, at least two doublets $\{\tau_i, \varepsilon\}$ are required. The interest of such microstructure-tortuosity relationships is to (i) tune microstructure to lower tortuosity factor and thus improve diffusion and (ii) predict tortuosity factors for a wide range of porosity which is particularly valuable for modeling. Note that many other microstructure-tortuosity relationships, that consider additional microstructure properties such as constriction and sinuosity, exist in the litterature[34].

$$\frac{D_{eff,i}}{D_{bulk}} = \frac{\varepsilon}{\tau_i} \quad \text{[VI-t1]}$$

$$\tau = \begin{cases} \varepsilon^{-0.5} \; if \; particles \; are \; spherical \\ \varepsilon^{-1} \; if \; particles \; are \; cylindrical \end{cases} \quad \text{[VI-t2a]}$$

$$\tau = \varepsilon^{1-p} \quad \text{[VI-t2b]}$$

$$\tau_i = \gamma_i \varepsilon^{1-\alpha_i} \quad \text{[VI-t3]}$$

Tortuosity factor is calculated using Taufactor[2], from Dr. Samuel Cooper, with a finite-difference based approach, using fixed (i.e., Dirichlet) boundary conditions. Previous work has shown that tortuosity factors calculated with Taufactor are converging with values obtained with a finite element approach[17]. Calculation is done with the Taufactor *TauFactor* function.

```
for current_phase=1:1:number_phase % Loop on every phase
    code_tmp = INFO.phase(current_phase).code; % The code of the phase
    binary_phase = zeros(Domain_size(1),Domain_size(2),Domain_size(3));
    binary_phase( Phase_microstructure==code_tmp )=1;
    voxel_number = sum(sum(sum(binary_phase==1)));
    % Volume fraction
    volume_fraction_total(current_phase,:)=voxel_number/prod(Domain_size);
    for current_direction=1:1:number_dimension % Loop on every analysed direction
        % Call tortuosity algorithm (cf. Tau factor manual)
        if current_direction==1
            Taufactor_result = TauFactor('InLine',1,0,0,binary_phase,[0 0 0;0 0 0;1 0 0],[1 1
1]);
            if Tau_factor_result.Tau_W1.Tau == Inf % No percolation path. 65535 is default value
for inf tortuosity factor
                Tau_factor_result.Tau_W1.Tau = NaN;
            end
            Tortuosity_factor(current_phase,current_direction)=Tau_factor_result.Tau_W1.Tau;
        elseif current_direction==2
            Tau_factor_result = TauFactor('InLine',1,0,0,binary_phase,[0 0 0;0 0 0;0 1 0],[1 1
1]);
            if Tau_factor_result.Tau_W2.Tau == Inf
                Tau_factor_result.Tau_W2.Tau = NaN;
            end
            Tortuosity_factor(current_phase,current_direction)=Tau_factor_result.Tau_W2.Tau;
        elseif current_direction==3
            Tau_factor_result = TauFactor('InLine',1,0,0,binary_phase,[0 0 0;0 0 0;0 0 1],[1 1
1]);
            if Tau_factor_result.Tau_W3.Tau == Inf
                Tau_factor_result.Tau_W3.Tau = NaN;
            end
            Tortuosity_factor(current_phase,current_direction)=Tau_factor_result.Tau_W3.Tau;
        end
        % Effective diffusion coefficient
        Effective_diffusion_coefficient(current_phase,current_direction) =
1*volume_fraction_total(current_phase,1)/Tortuosity_factor(current_phase,current_direction);
        tau = Tortuosity_factor(current_phase,current_direction);
```

```
        eps = volume_fraction_total(current_phase,1);
        Bruggeman_exponent(current_phase,current_direction) = 1 - log(tau)/log(eps);
    end
end
```

User can, in addition to calculate tortuosity factor of the whole volume, calculate tortuosity factor section per section (independently) using bottom half part of the 'Tortuosity factor (tau factor) options' tab – similarly with connectivity.

If a voxel size dependence analysis is asked by the user, both tortuosity factor, Bruggeman exponent, and normalized diffusion coefficient are plotted as function of voxel size. As well, if RVA is asked, RVE size are plotted for tortuosity factor, Bruggeman exponent, and normalized diffusion coefficient.

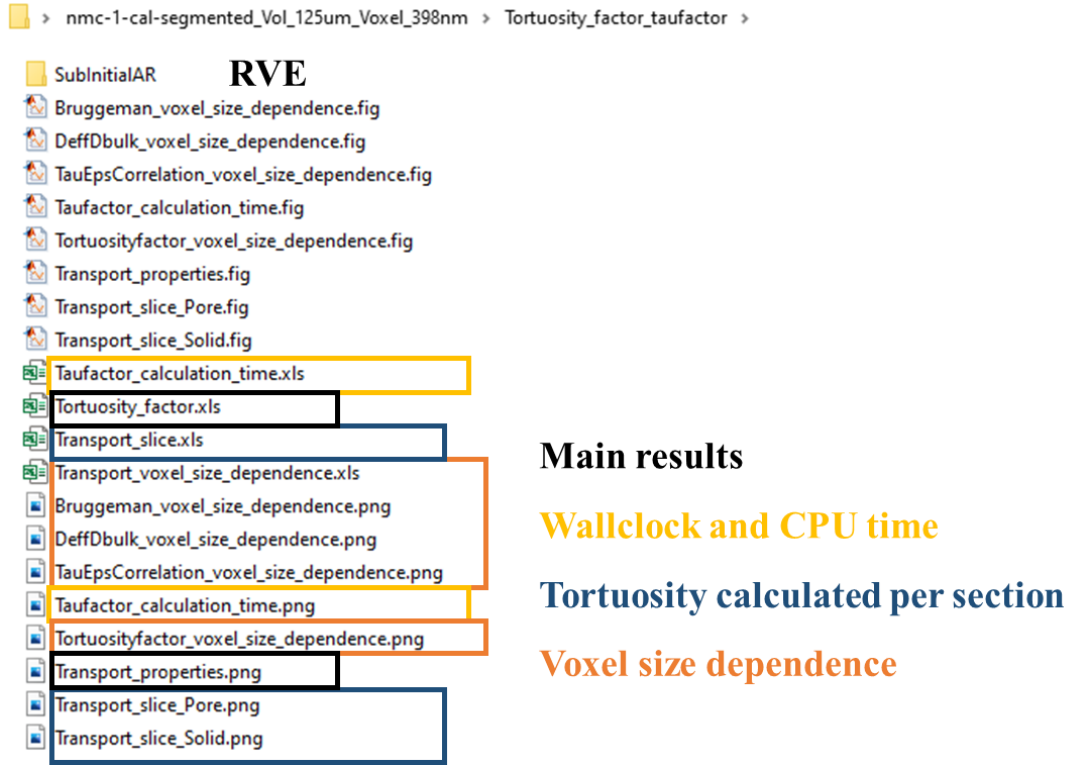


Figure VI-16a. Tortuosity factor result files organization.

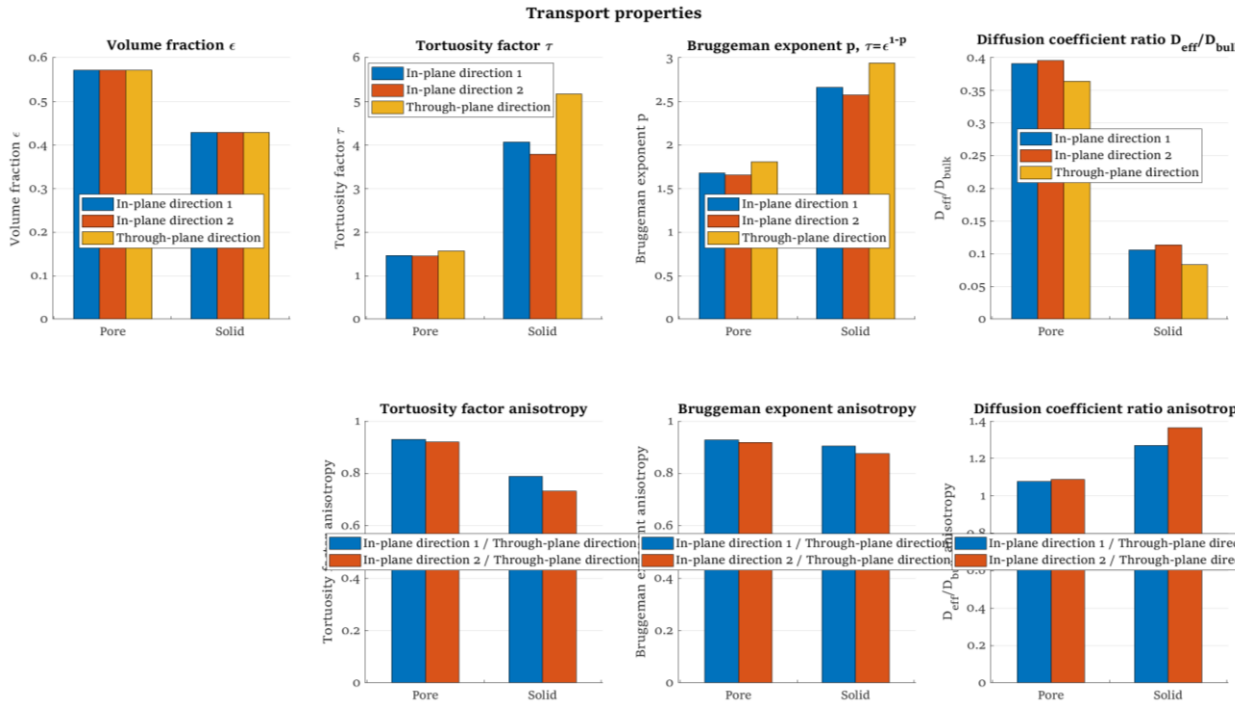


Figure VI-16b. (Top) Volume fraction, tortuosity factor, Bruggeman exponent, and normalized effective diffusion coefficient. (Bottom) Coefficients anisotropy.

d. Specific surface area

Specific surface area $S_{p,i}$ of phase *i* (m^{-1}) is the phase surface ∂V_i normalized by the phase volume V_i or the domain volume (both definitions are used). It is commonly used as a scaling factor for the electrochemical reaction that occurs at the solid electrolyte active interface in macroscopic models. Specific surface area obtained from microstructure characterization should be considered with great caution, due to a fractal issue discussed in §VI-2a.

$$S_{p,i} = \frac{\partial V_i}{V_i} \quad or \quad S_{p,i} = \frac{\partial V_i}{\sum_{i=1}^{N} V_i} \; with\, N \; the\; number\; of\; phases \qquad [VI\text{-}s1]$$

i. Direct method

Results are saved in **Save folder/result folder/Specific_surface_direct**. Specific surface area are handled with the file Function_Specificinterface_direct.m, and calculated with Function_Specificsurface_direct_Algorithm.m. Surface area can be simply calculated by adding all the voxels faces located at the boundary between the investigated phase and the complementary phase[40,43,52] (direct method). It is well known the discretization of the medium in cuboids induces an overestimation of the surface area calculated with this method[43,52–54]. For instance, for ideal non-overlapping spheres, a corrective factor of 2/3 can be used to correct the overestimation. Note that this corrective factor is aimed only to correct the overestimation induced by the voxel discretization and not the unknown surface roughness. Particles of NMC and graphite electrodes

are typically oblate spheroid (i.e., ellipsoid of revolution with two identical diameters larger than the third diameter: $a = b, a > c$)[34] for which a corrective factor can be deduced as described here. The *voxel-based* numerical surface area of an oblate spheroid is given by eq. VI-s2a. Note that it corresponds to the surface projection of the ellipsoid along each axis. The theoretical surface area of an oblate spheroid can be approximated using eq. VI-s2b[55].

$$S_{numerical} = 2\pi ab + 2\pi ac + 2\pi bc = 2\pi(a^2 + 2ac) \quad with \quad a = b \qquad \text{[VI-s2a]}$$

$$S_{theoritical} = 2\pi\left(a^2 + \frac{c^2}{e}atanh(e)\right) \quad with \quad e^2 = 1 - \frac{c^2}{a^2} \qquad \text{[VI-s2b]}$$

We can then deduce the expression of the corrective factor *C* as:

$$C(r) = \frac{S_{theoritical}}{S_{numerical}} = \frac{1 + \frac{r^2}{\sqrt{1-r^2}}athanh\left(\sqrt{1-r^2}\right)}{1 + 2r} \quad with \quad r = \frac{c}{a} \in]0,1[\qquad \text{[VI-s2c]}$$

The expression of the corrective factor is plotted in figure VI-17a. For the extreme case short diameter over long diameter equals 0 (i.e., a=b, c=0), the geometry is no more a volume but a two-dimensional ellipsoid. Since the voxel-based numerical value correspond to the sum of the 2D projections of the volume, it matches with the theoretical value for this extreme case.

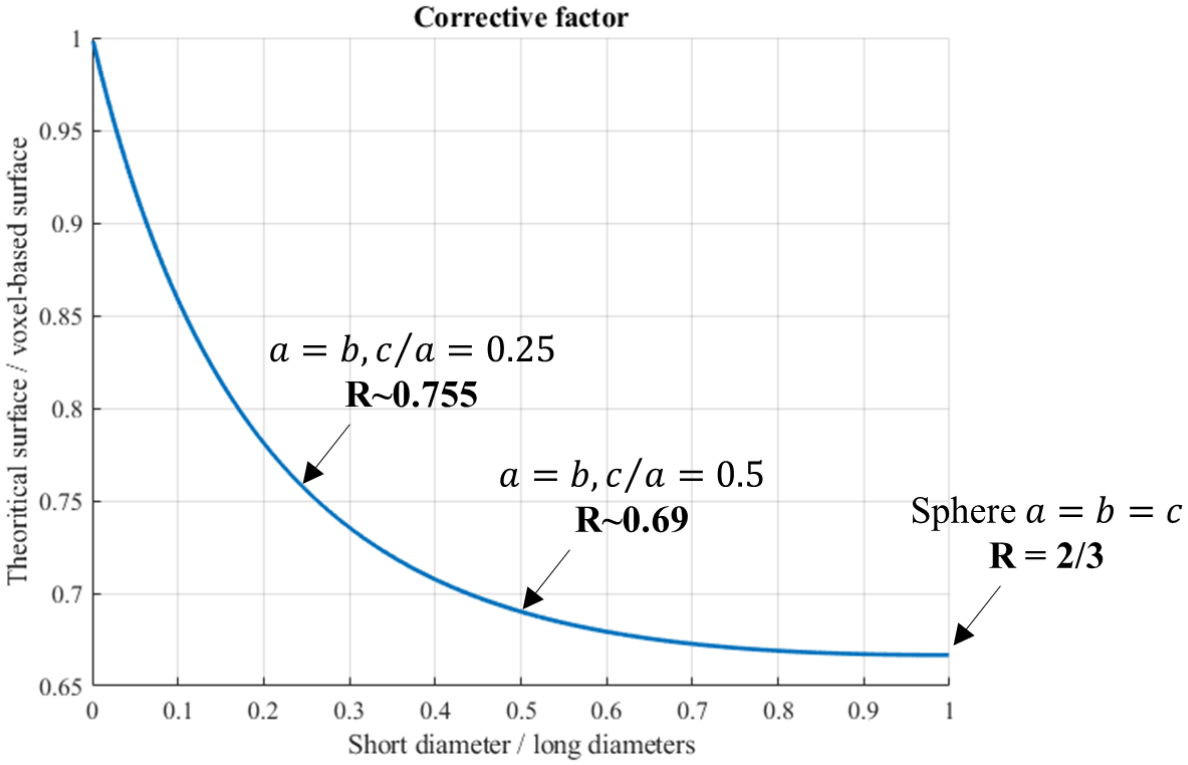


Figure VI-17a. Corrective factor as a function of ellipsoid short diameter over long diameters ratio.

You can choose the value of the corrective factor in the 'specific surface area (direct) options' tab, by default set to 2/3 (which corresponds to non-overlapping spherical particles). Surface area can be normalized with the phase volume or with the domain volume.

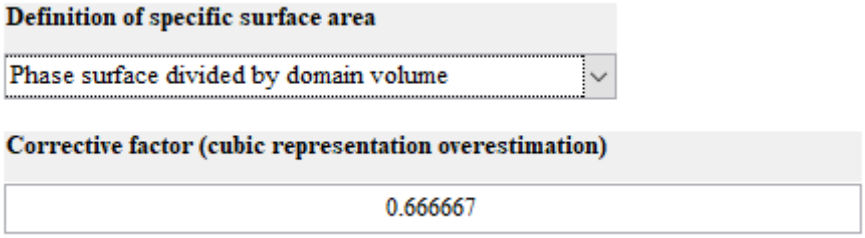


Figure VI-17b. Options specific to specific surface area.

Calculation is performed on a binary matrix (i.e., voxel=1 if it belongs to the investigated phase, 0 otherwise) where the logical exclusive-OR (xor in MATLAB) is detect adjacent voxels for which one belongs to the phase and the other not.

```
% Detection of the faces normal to the direction 1
Face_normal_1 = xor( binary_phase(1:Current_Domain_size(1)-1,:,:) ,
binary_phase(2:Current_Domain_size(1),:,:));
% Summation of all these faces
sum1 = sum(sum(sum( Face_normal_1 )));
```

File organization is as below:

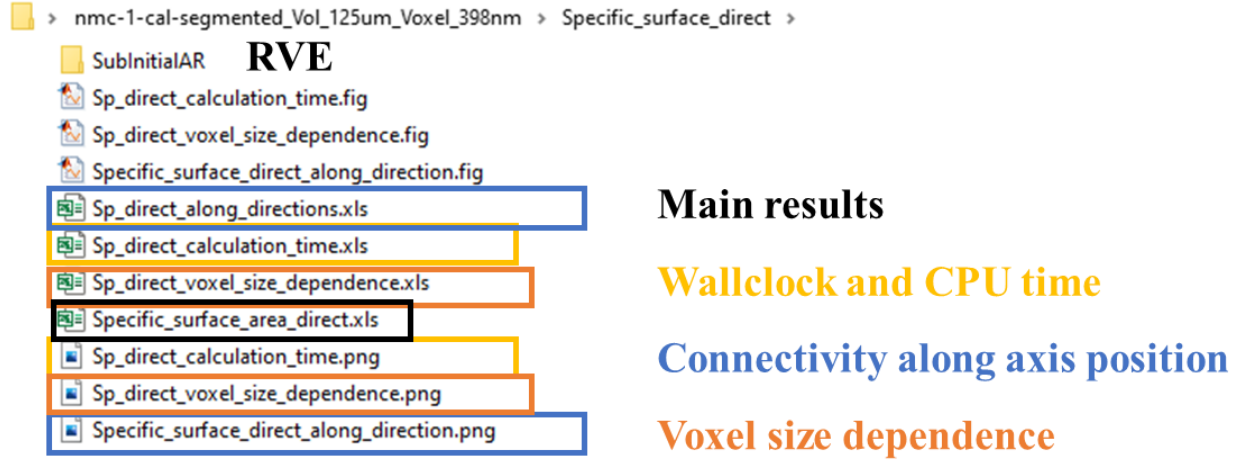


Figure VI-17c. Specific surface area result files organization.

Specific surface area S_p are also plotted along directions. Following a similar mathematical approach used to calculate distribution functions, the curve f(x) is calculated and plotted such as:

$S_p = \frac{1}{L}\int_{x=0}^{L} f(x)\,dx$ with L the domain length. Although f(x) is not a probability density function as its dimension is µm^{-1}.

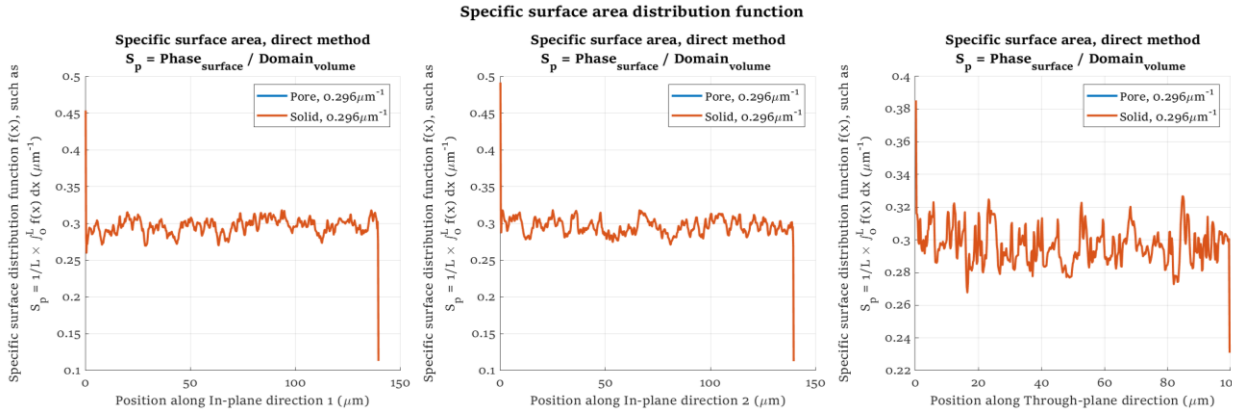


Figure VI-17d. Specific surface area plotted along one axis. Because the volume has only two phases and that we normalized with domain volume, both phase has the same specific surface area. Values noted in the legend are the specific surface calculated on the whole volume. Because we are plotting the function f(x), both have similar values.

e. Specific interface area

Specific interface area (m^{-1}) of phase *i* and *j* is the interfacial area between these two phases S_{i-j} normalized by the domain volume. Specific interface area is especially relevant for microstructures with more than two phases. For instance, in lithium ion battery it can be used to quantify the active interface area between pore and active material particles, while considering a third solid phase (additives). Specific interface area obtained from microstructure characterization should be considered with great caution, due to a fractal issue discussed in §VI-2a.

$$S_{p,i-j} = \frac{S_{i-j}}{\sum_{i=1}^{N} V_i} \quad with\ N\ the\ number\ of\ phases \qquad [VI\text{-}i1]$$

Results are saved in **Save folder/result folder/ Specific_interface_direct**. Specific surface area are handled with the file Function_Specificinterface_direct.m. and calculated with Function_Specificinterface_direct_Algorithm.m. Interface area is calculated for each phase-phase interface permutation. Interface area are calculating by adding all the voxels faces located at the boundary between the two investigated phases. Similarly with the specific surface area direct method, a corrective factor can be applied to correct the interface area overestimation induced by the cuboid representation. Note that interfacial area is not calculated if the number of phase (including pore) is not superior to 2. For bi-phase microstructure, please calculate the specific surface area instead. File organization is as below:

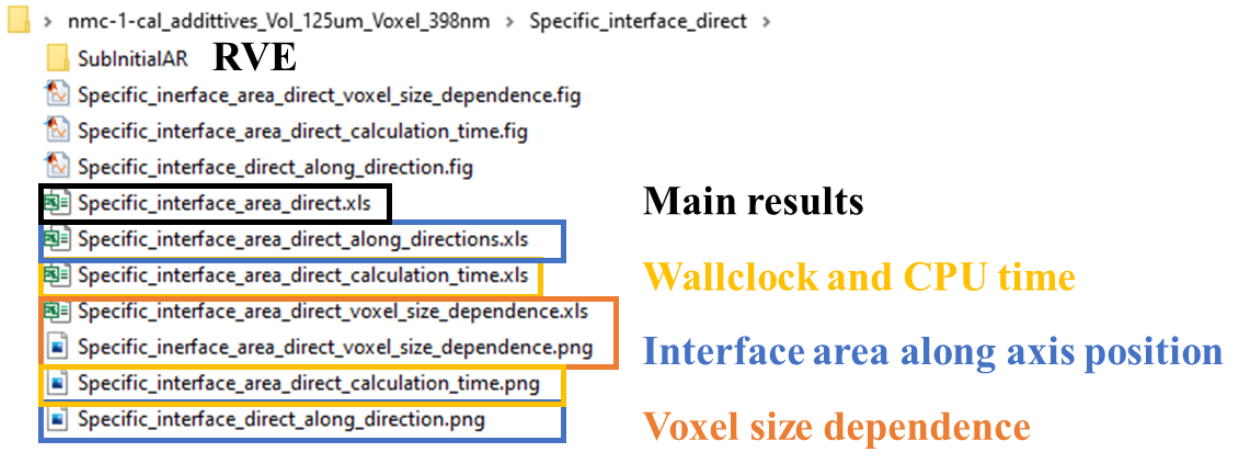


Figure VI-18a. Specific interfacial area result files organization

Similarly with specific surface area, specific interface area $S_{p,i-j}$ are also plotted along directions. Following a similar mathematical approach used to calculate distribution functions, the curve g(x) is calculated and plotted such as: $S_{p,i-j} = \frac{1}{L}\int_{x=0}^{L} g(x)\,dx$ with L the domain length. Although g(x) is not a probability density function as its dimension is μm^{-1}. A three-phase domain is used for illustration.

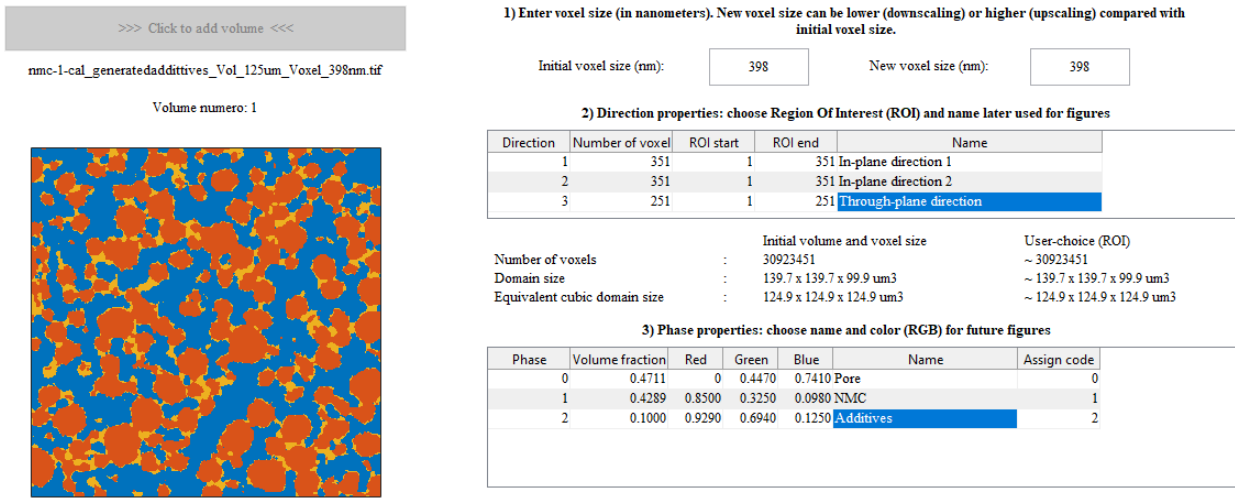


Figure VI-18b. Three phases material (import tab). Additive phase has been numerically generated using the generation module.

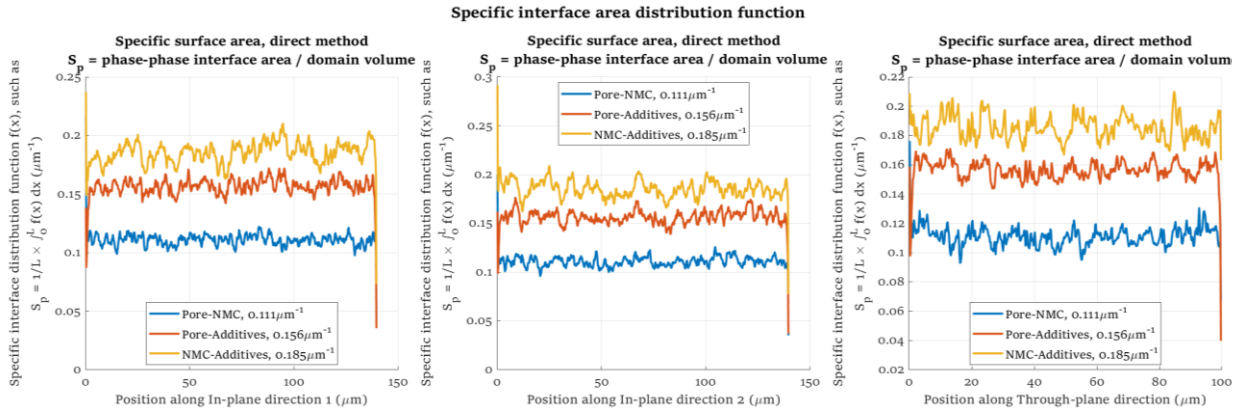


Figure VI-18c. Specific interface area plotted along one axis. Values noted in the legend are the specific interface area calculated on the whole volume. Because we are plotting the function g(x), both have similar values.

f. Particle size

Three-dimensional volume reconstructions of lithium ion battery electrodes obtained with computed tomography usually do not show distinct particles, but rather a unique connected cluster that do not allow for a straightforward determination of the particle size[34,43]. Indeed, there is simply not a unique definition of what is a particle within a fully connected cluster. Particle size calculation is therefore a somewhat ill-defined problem as the object to quantify, 'particle', is not defined. Because of this definition issue, a large variety of numerical methods has been developed in the literature. Short review of methods is available in[34] along with a comparison of diameter predictions between them.

i. Spherical assumption, continuum Particle Size Distribution (c-PSD) and particle level description

Results are saved in **Save folder/result folder/Particle_size_Cpsd.** Particle sizes are handled with the file Function_particle_size_CPSD.m. and calculated with Function_particle_size_CPSD_Algorithm.m.

- The spherical assumption approach consists in attributing for each voxel of the investigated phase, the diameter of the largest sphere that (i) contains it and (ii) does not overlap with the complementary phase[43,56,57]. Spheres are allowed to overlap between each other (thus the adjective continuum). Because the c-PSD algorithm relies on a spherical assumption, it provides a strong underestimation of the actual diameter[34].
- The particle size field c-PSD is used to deduce another metric, noted $Detail_{particle}(V)$, that quantifies the particle level of description of the phase *V*. The particle level of description indicates if the image resolution is high enough to describe accurately the particle of the

investigated phase *V*. To calculate it, the volume sum $V_{one-voxel}(V)$ of all the particles with a one-voxel-size is normalized with the phase volume:

$$V_{one-voxel}(V) = \frac{\sum_{i=1}^{N} k_{one-voxel}(v_i)}{\sum_{i=1}^{N} k(v_i)}$$

$$with \begin{cases} k_{one-voxel}(v_i) = \begin{cases} 1 \; if \; v_i \in \partial V \subset V \; and \; c - PSD(v_i) = voxel \; length \\ 0 \; otherwise \end{cases} \\ k(v_i) = \begin{cases} 1 \; if \; v_i \in V \\ 0 \; otherwise \end{cases} \end{cases} \qquad \text{[VI-p1]}$$

The particle level description criterion $Detail_{particle}$ is then deduced as done in[43]:

$$Detail_{particle}(V) = 1 - V_{one-voxel}(V) \qquad \text{[VI-p2]}$$

A value near 1 indicates that almost every particles are described with more than one voxel, while a low value indicates a significant ratio of particles are ill-described suggesting the image resolution is too coarse. Voxel size dependence analysis can be used to identify adequate image resolution.

File organization is as below:

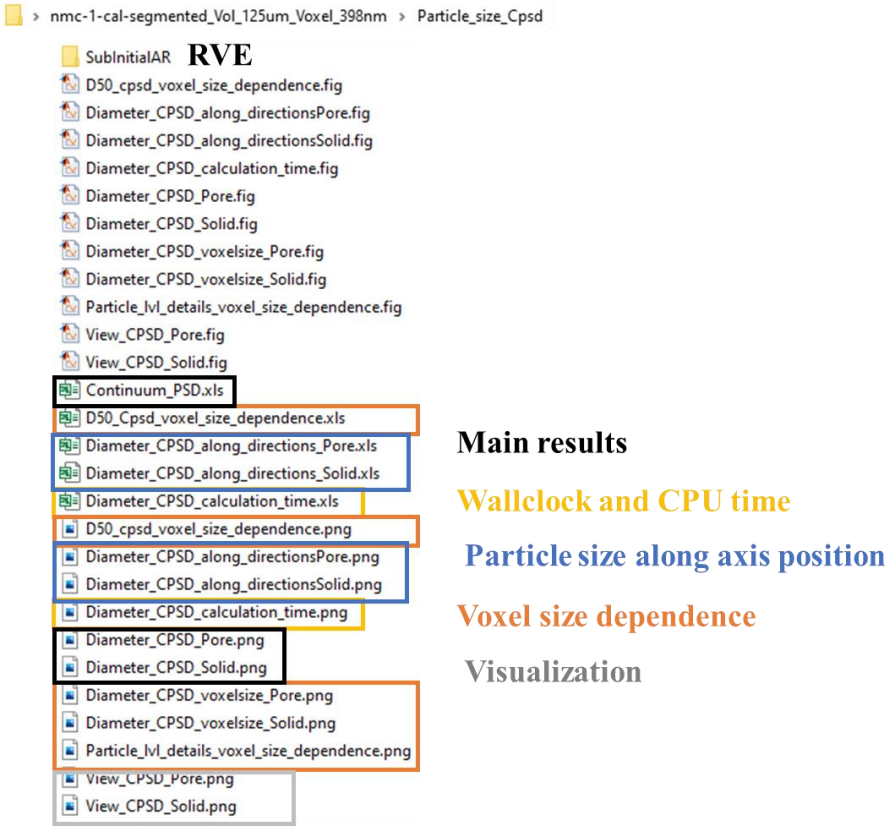


Figure VI-19a. Particle size, spherical assumption, result files organization

The particle size distribution is calculated with the file Function_probability_density.m (src\Miscellaneous), using a moving average filter on the cumulative function to smooth its derivate and avoid a noisy distribution function.

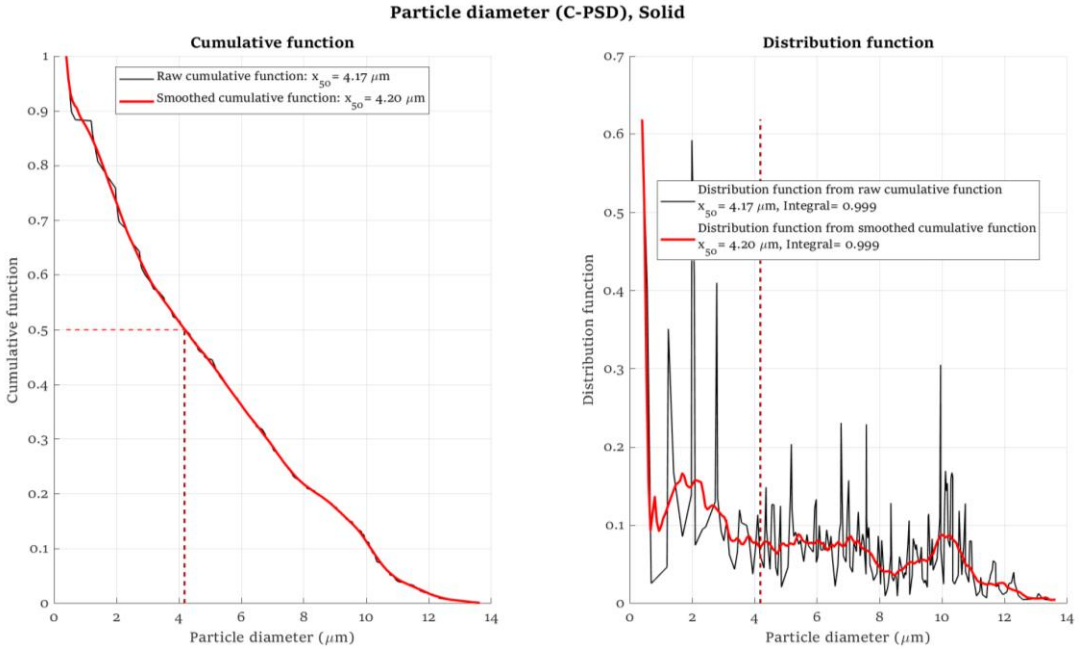


Figure VI-19b. Particle size distribution of the solid phase. Black line corresponds to the raw cumulative function, while the red line corresponds to the smoothed cumulative function. Peak near the small particle diameter is an artifact of the method (many tiny irrelevant particles are located at the phase edge) and should not be considered.

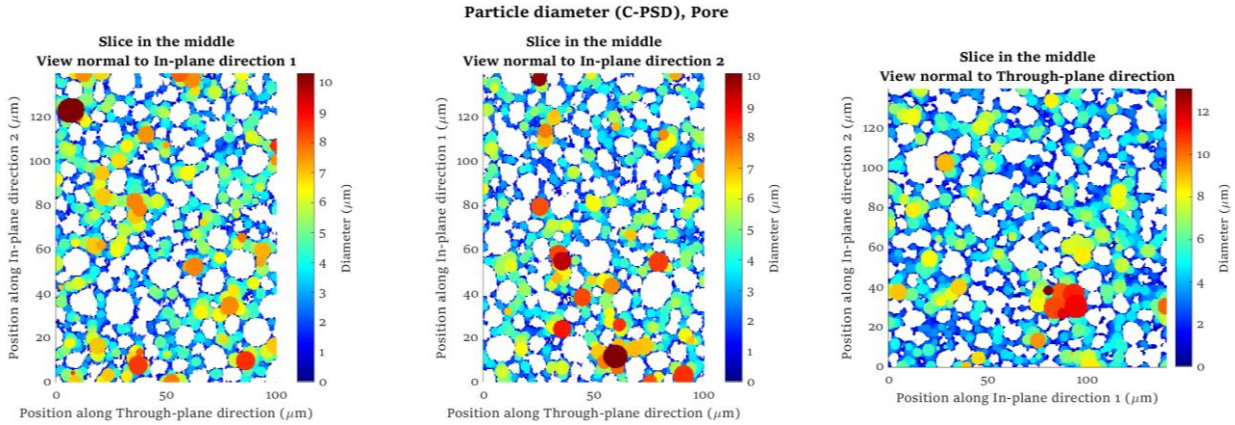


Figure VI-19c. Pore size visualized on three slices. Notice the field c-PSD is continuous (particle are not identified distinctly).

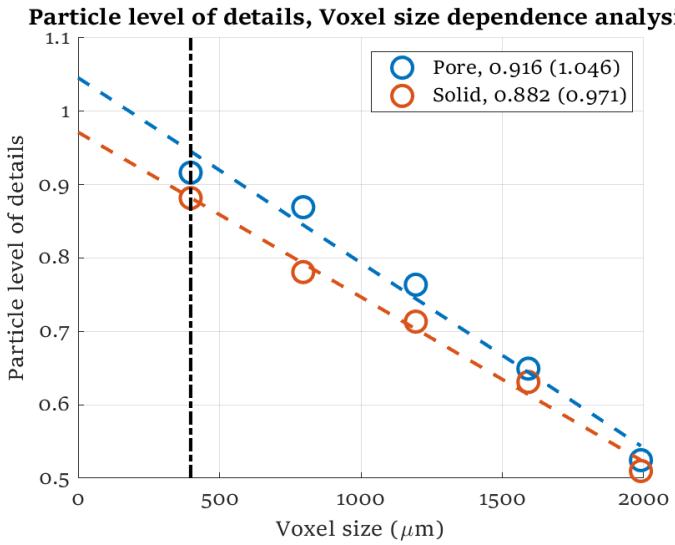


Figure VI-19d. Particle level of detail as a function of voxel size. At the initial image resolution, around 10% of the phase volume is described with one-voxel size particle.

ii. Euclidean distance map fitting method (EDMF).

Results are saved in **Save folder/result folder/Particle_size_dmap.** Particle sizes are handled with the file Function_particle_size_distancemap.m. and calculated with Function_particle_size_distancemap_Algorithm.m.

The method introduced in[34] consists in fitting the diameter of a sphere so that its Euclidean distance map cumulative function matches the one calculated on the complex microstructure. Note that the distribution function of the Euclidean distance map is not the distribution of the particle size. This method only provides the mean particle diameter, and not its distribution. Similarly, the graph plotting distance along axis position are the distance to surface distance, and not the particle diameter.

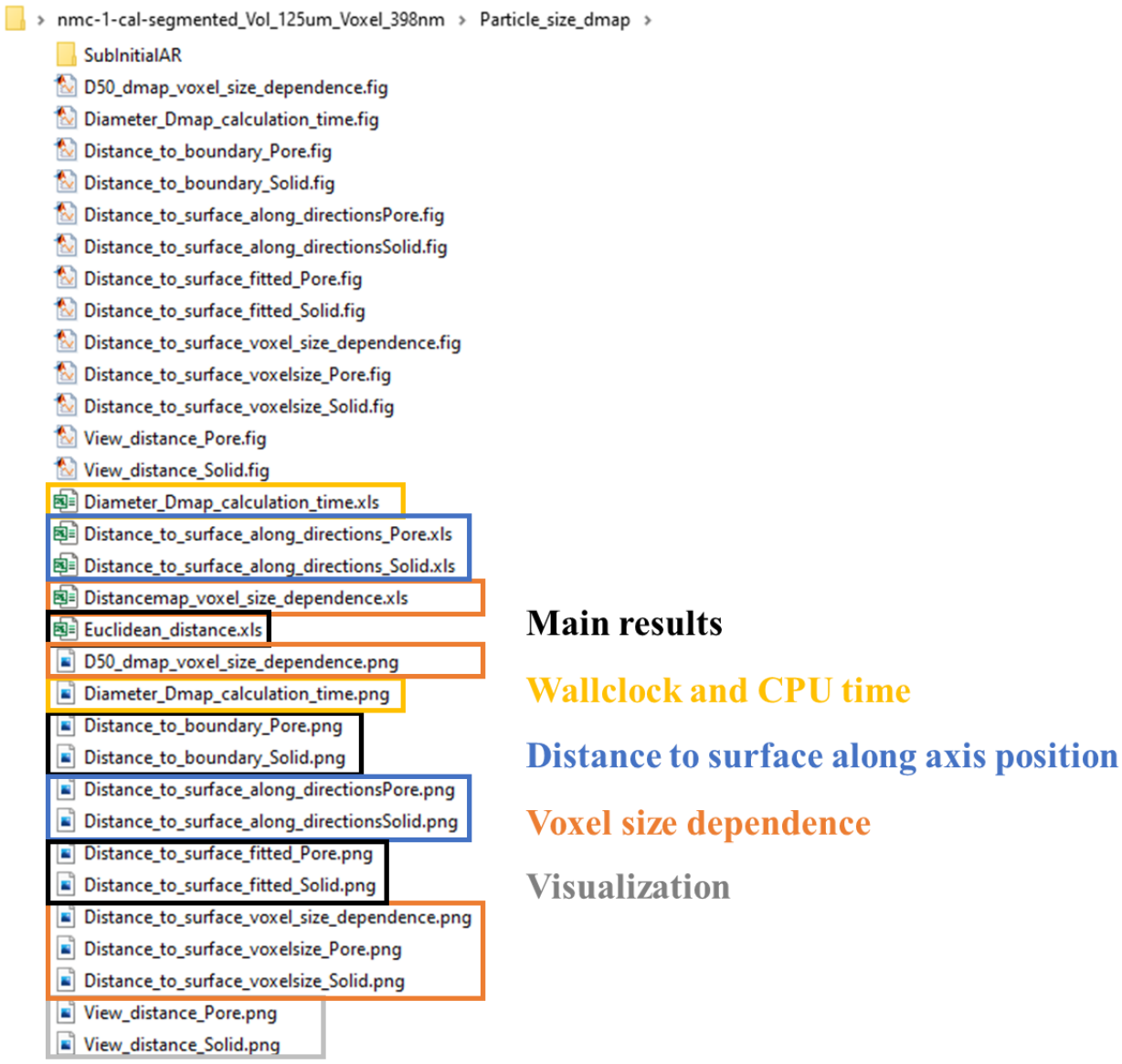


Figure VI-20a. Particle size, EDMF, result files organization

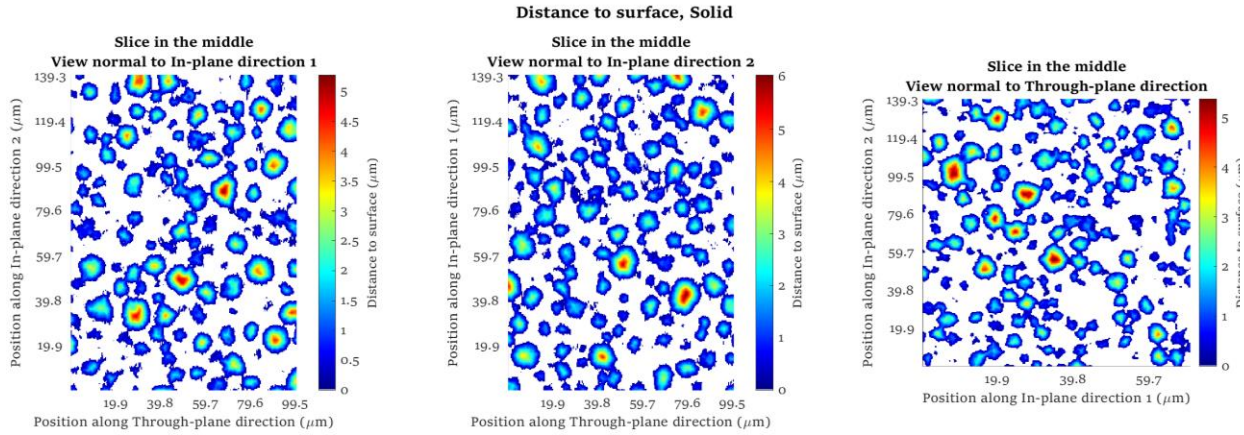


Figure VI-20b. Euclidean distance map illustrated on slices of the solid phase

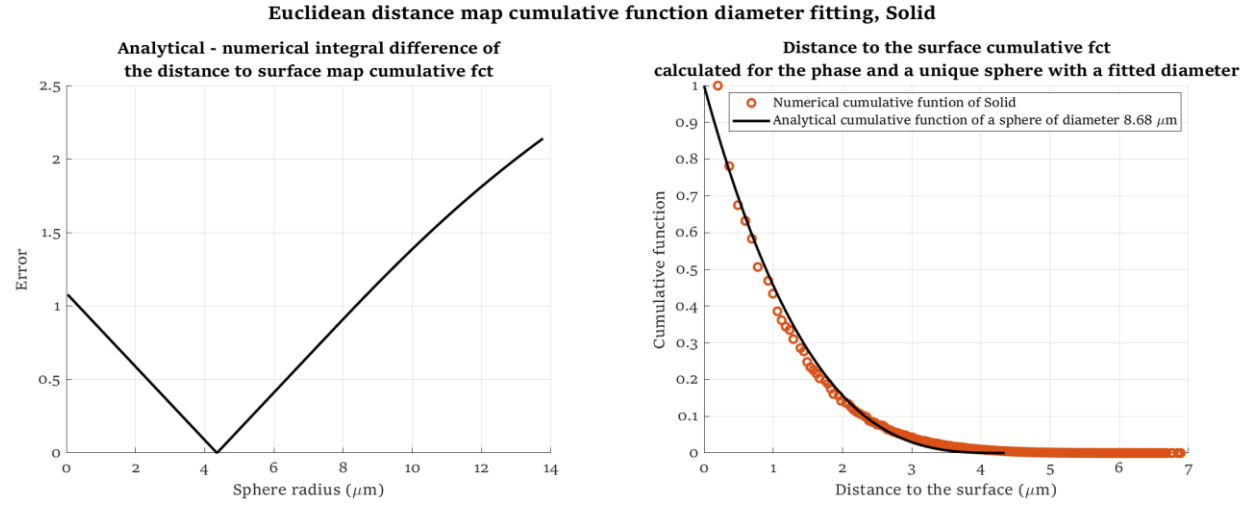


Figure VI-20c. Euclidean distance map cumulative function of a sphere is fitted to match the Euclidean distance map cumulative function calculated on the solid domain to determine the particle mean diameter.

iii. Watershed method, discrete Particle Size Distribution (d-PSD)

BETA: algorithm is bug-free to author's knowledge but is very slow. It is recommended to use it with small domain or to downscale the image resolution. Nevertheless, it was fast enough to characterize 3D electrode volumes as done in[34]. Voxel size dependence and RVE analysis not yet implemented. Method usable through the GUI for whole volume analysis only.

Watershed[58] is a standard method for particle identification/segmentation. Method is illustrated on lithium ion battery electrode in[34,43].

You can visualize the watershed method by calling directly the algorithm function using a 2D array as described below (visualization will slow down the calculation and is not representative of the true algorithm speed):

```
M=function_load_tif('C:\Users\fussegli\Desktop\nmc-1-cal-segmented.tif');
solid_phase_id = 1;
binary_phase = zeros(size(M));
binary_phase(M==solid_phase_id)=1;
binary_phase_2D = binary_phase(:,:,50); % Choose any slice
customfunction = true;
cpsd_refining = true; % Correct oversegmentation
details_convergence = true; % Write algorithm progression in the MATLAB command window
visualize_2D = true; % Create figure and video illustrating the algorithm progression and result
[D_PSD_particle_size,Label_lake] =
Function_Discrete_particle_size_watershed_immersion_algorithm(binary_phase_2D,cpsd_refining,custo
mfunction,details_convergence,visualize_2D);
```

The code above will generate figures *VI-21a-c*.

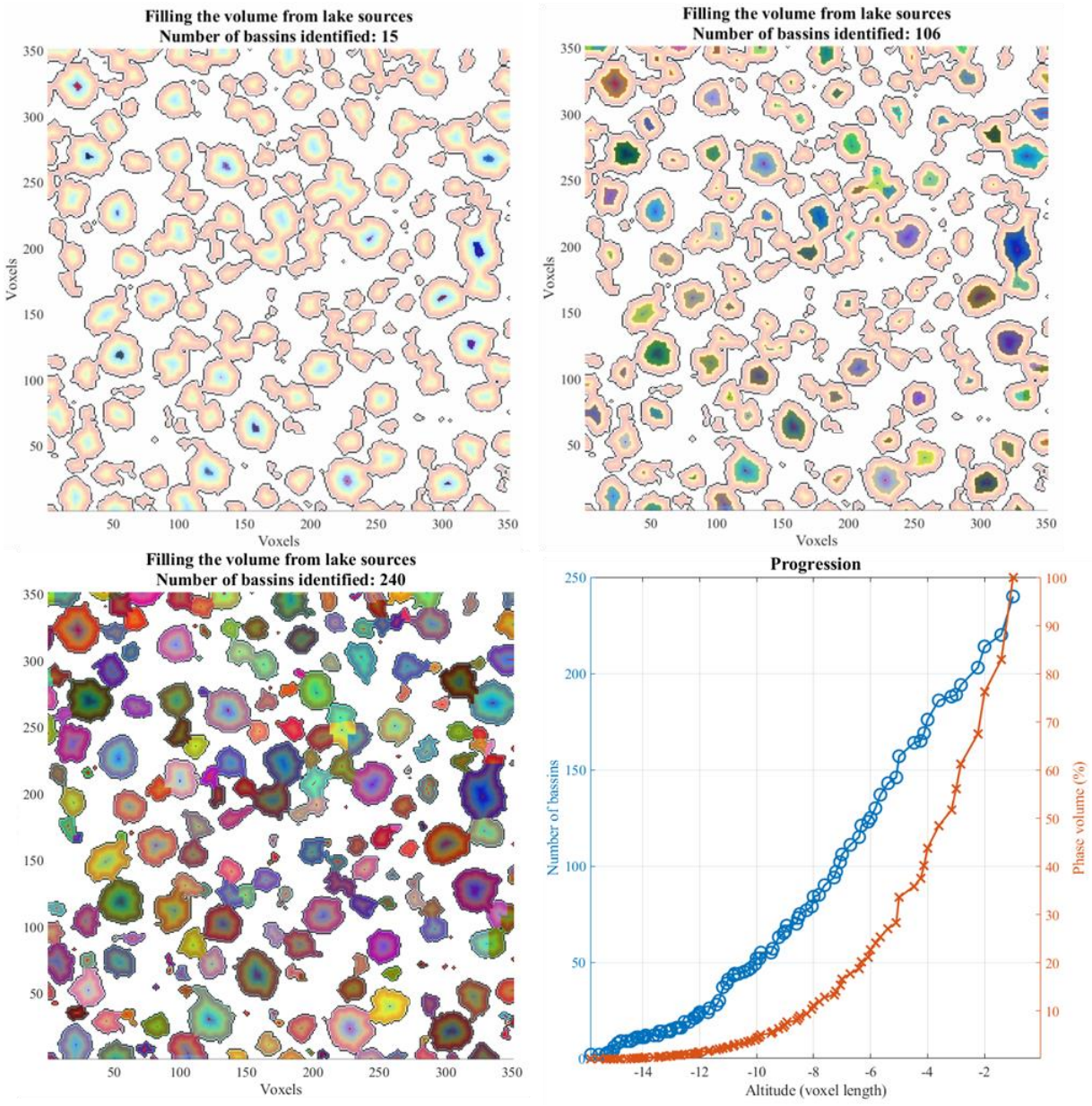


Figure VI-21a. Flooding progression illustrated on a 2D slice of an NMC electrode.

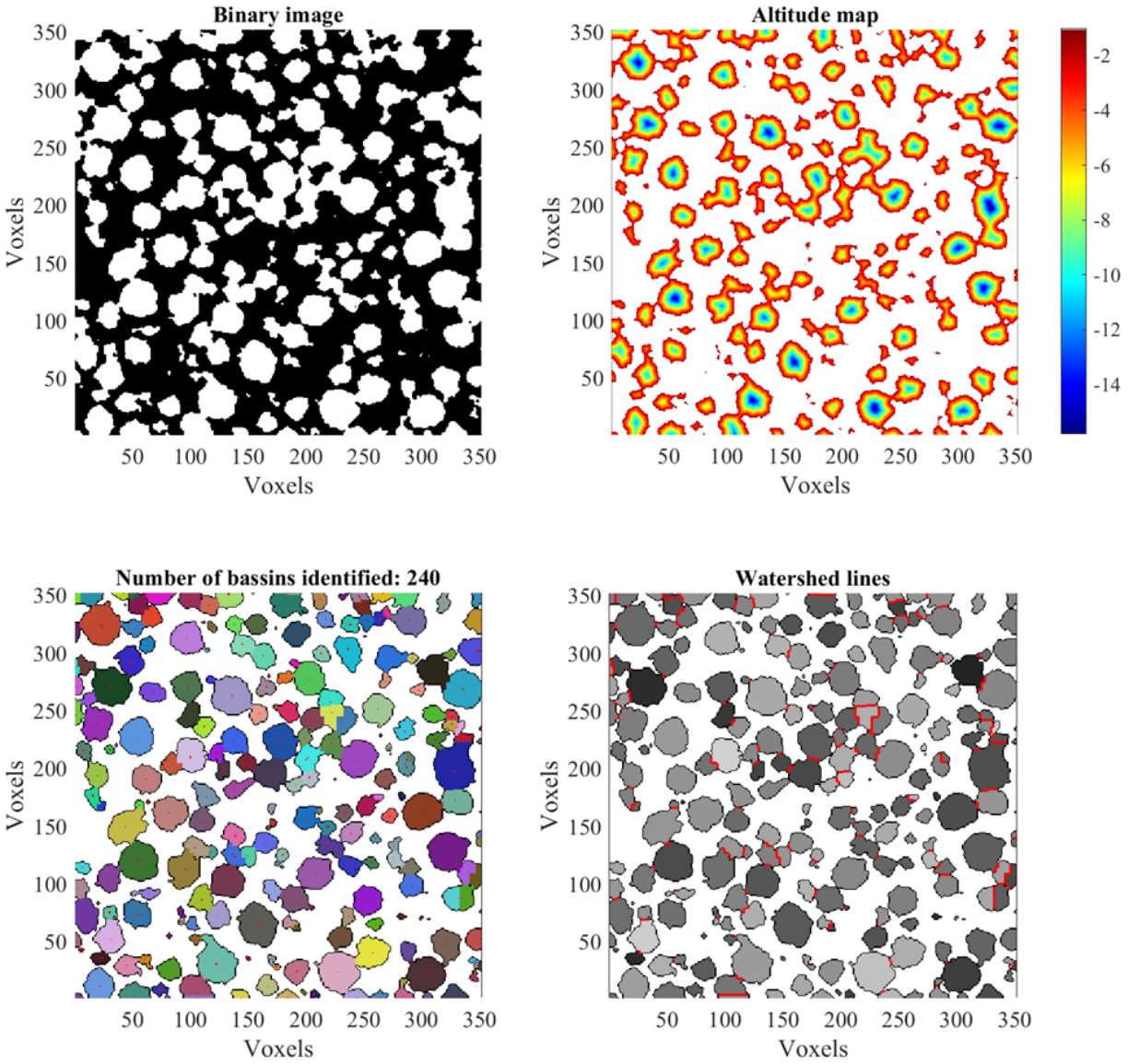


Figure VI-21b. Particle identification on a 2D slice using the watershed algorithm, after over segmentation correction.

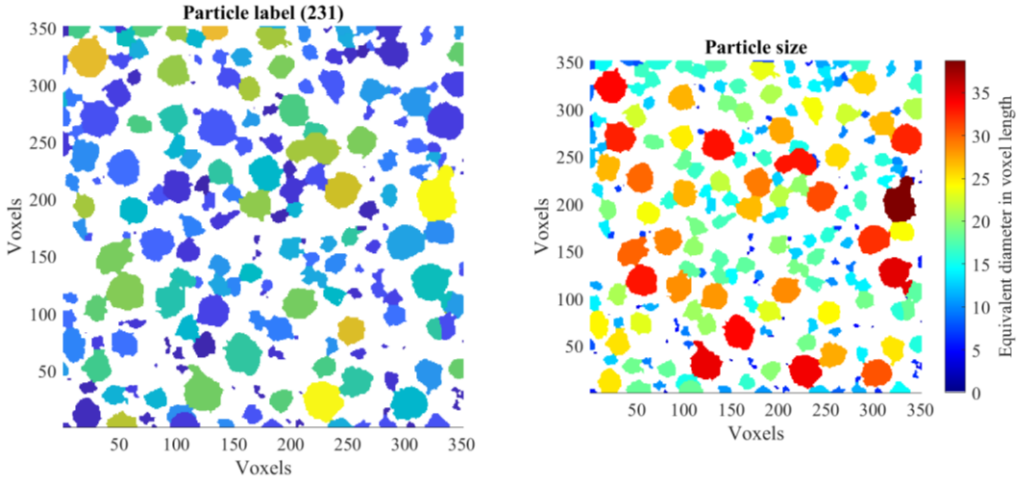


Figure VI-21c. Particle labeling and size (equivalent diameter) on a 2D slice using the watershed algorithm, after over segmentation correction.

- Because watershed method is gradient based, it usually suffer from over-segmentation or over-identification due to the presence of concavities on the particle surfaces (e.g. non-smooth interfaces), which induce a noisy gradient image and generate a lot of irrelevant, oddly shaped small particles. Correct watersheds lines can be then lost in a mass of irrelevant ones, even if the gradient image has been previously filtered. Over-segmentation is discussed in length in the literature[34]. Over segmentation is corrected using the method introduced in[43] that discards particles assigned with a diameter lower than the one obtained with a c-PSD algorithm and reassign their voxels to adjacent particles, assuming that the smallest possible geometric feature that can be considered as a particle is the largest sphere and everything below is an artifact (i.e., assuming c-PSD gives the particle size lower bound)[34,43]. The method employed here is also used for the other d-PSD method, Pseudo-Coulomb Repulsive field (PCRF). Over segmentation correction is illustrated on the figure below:

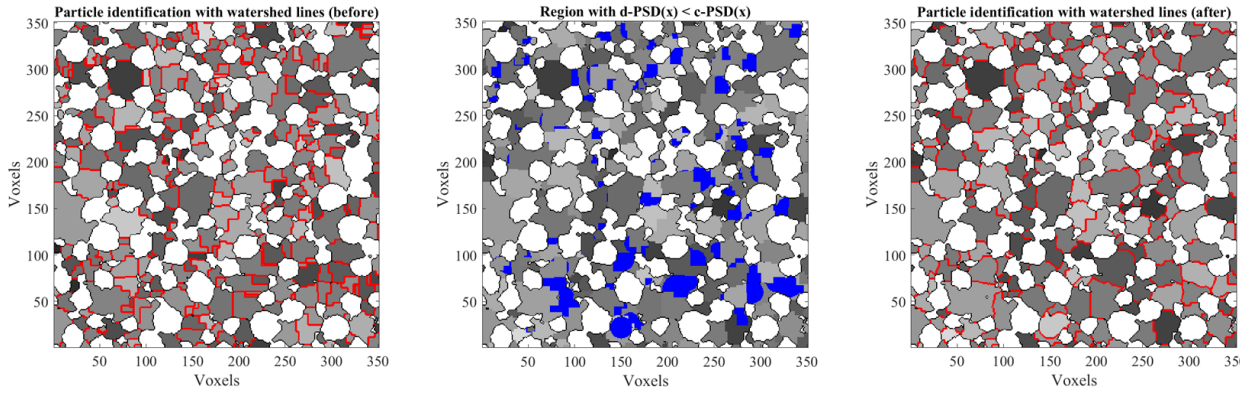


Figure VI-21d. Watershed lines before and after over segmentation correction illustrated for the pore domain of an NMC electrode

iv. Pseudo-Coulomb Repulsive field (PCRF) method, discrete Particle Size Distribution (d-PSD)

BETA: algorithm is bug-free to author's knowledge but is very slow. It is recommended to use it with small domain or to downscale the image resolution. Nevertheless, it was fast enough to characterize 3D electrode volumes as done in [34]. Voxel size dependence and RVE analysis not yet implemented. Method usable through the GUI for whole volume analysis only.

Method is illustrated on lithium ion battery electrode in [34].

You can visualize the PCRF method by calling directly the algorithm function using a 2D array as described below (visualization will slow down the calculation and is not representative of the true algorithm speed):

```
M=function_load_tif('C:\Users\fussegli\Desktop\nmc-1-cal-segmented.tif');
solid_phase_id = 1;
binary_phase = zeros(size(M));
binary_phase(M==solid_phase_id)=1;
binary_phase_2D = binary_phase(:,:,50); % Choose any slice
cpsd_refining = true; % Correct oversegmentation
exponent_law = 3; % Coefficient k of the pseudo Coulomb law
max_dist = 40; % PCRF local value is calculated on a restricted field of view
2*max_dist;2*max_dist;2*max_dist centered on the local position to reduce CPU time
details_convergence = true; % Write algorithm progression in the MATLAB command window
visualize_2D = true; % Create figure and video illustrating the algorithm progression and result
[D_PSD_particle_size, Label_lake, Repulsive_matrix, Repulsive_matrix_sign] =
Function_Discrete_particle_size_PCRF_algorithm
(binary_phase_2D,cpsd_refining,details_convergence,visualize_2D,exponent_law,max_dist);
```

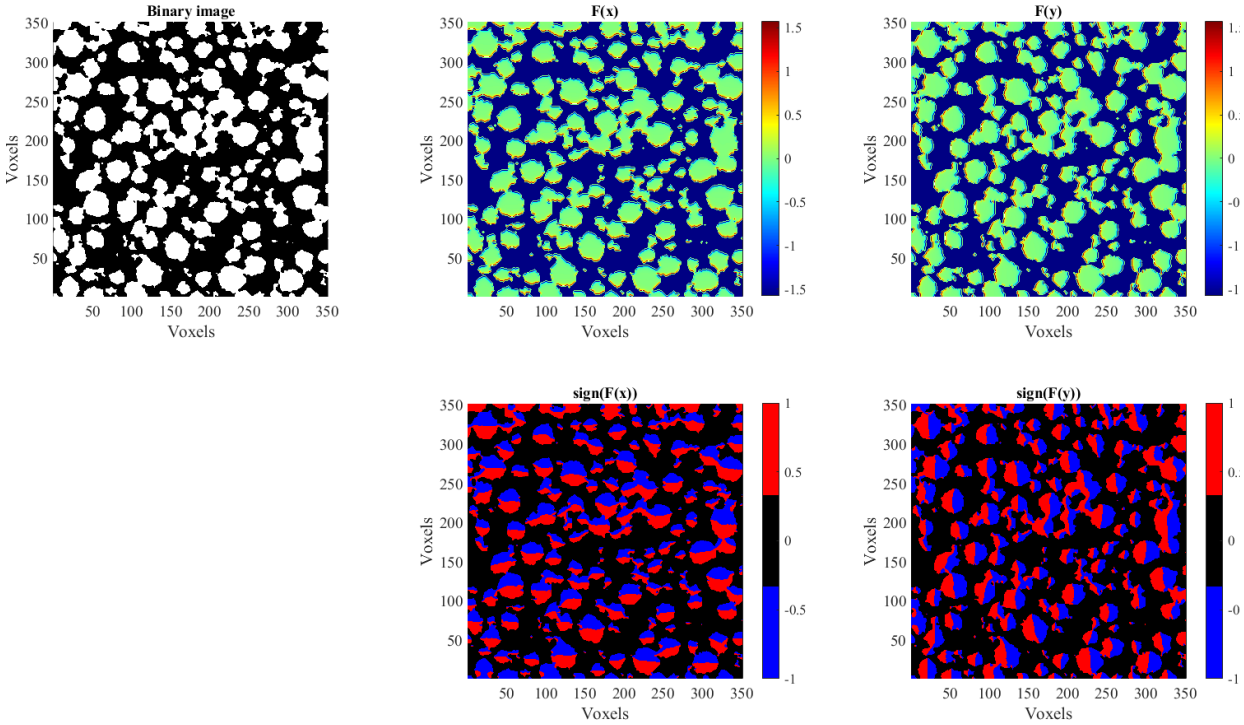


Figure VI-22a. (top left) Binary image, (center) Pseudo-Coulomb repulsive field along vertical direction and (right) Pseudo-Coulomb repulsive field along horizontal direction. The Pseudo-

Coulomb repulsive field push dropped imaginary charged points far from the phase edges and particle bottlenecks, eventually enabling particle identification after analyzing all charged points trajectory.

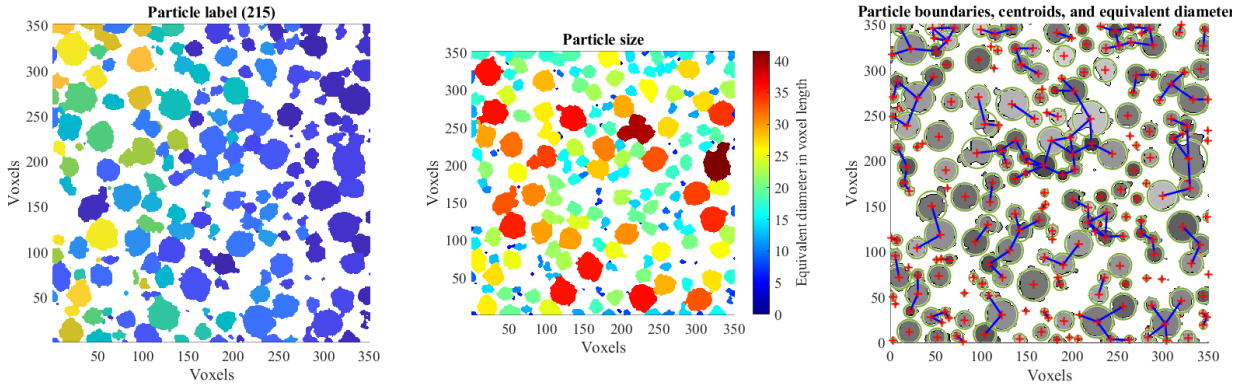


Figure VI-22b. Particle labeling and size (equivalent diameter) on a 2D slice using the PCRF algorithm, after over segmentation correction.

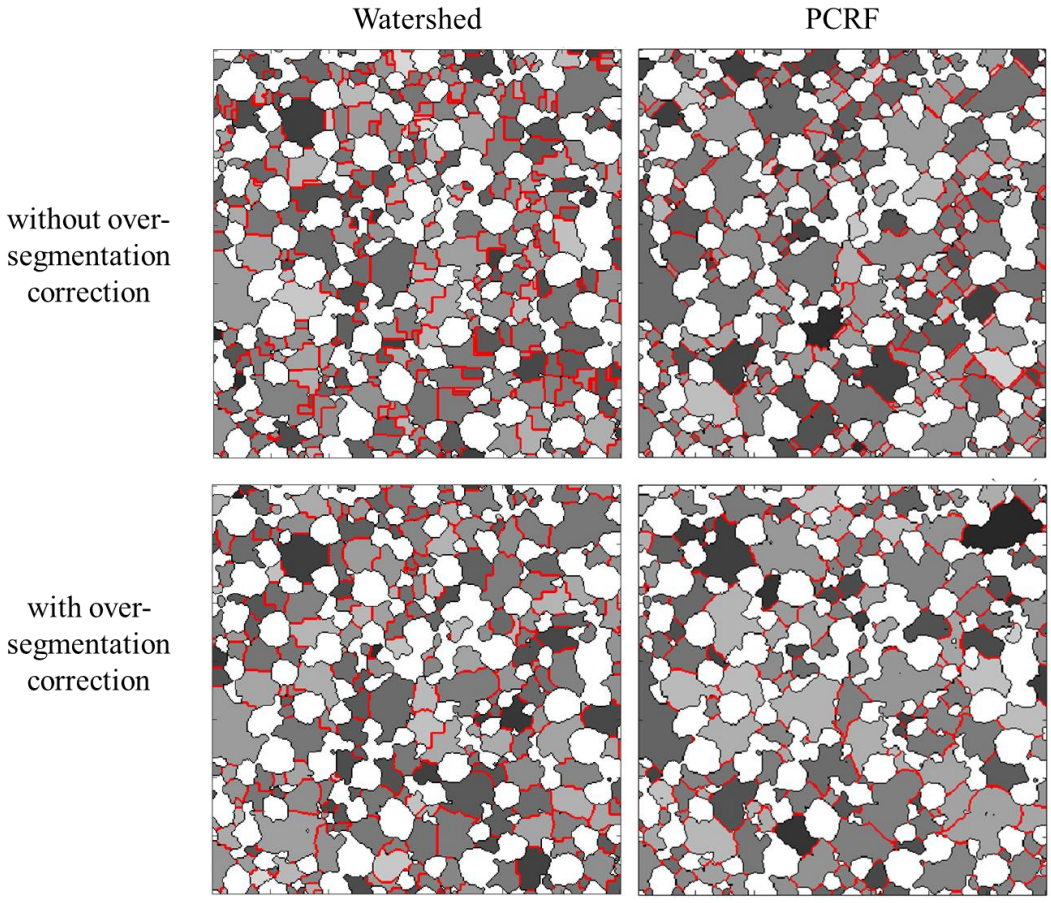


Figure VI-22c. PCRF method generates less over segmentation compared with watershed for reasons explained in[34] and is then a more reliable method.

g. Particle morphology deduced from particle identification

Both Watershed and PCRF method calculate particle aspect ratio and particle sphericity as explained in[34].

h. Domain topology deduced from particle identification

Both Watershed and PCRF method calculate geometric tortuosity and constriction factor as explained[34]. Constriction factor can then be deduced as detailed in[34].

6. Module organization

a. File hierarchy

This section is mostly valuable if you plan to modify the module. From a purely end-user point of view, you can skip it. The module organization is illustrated with the figure below.

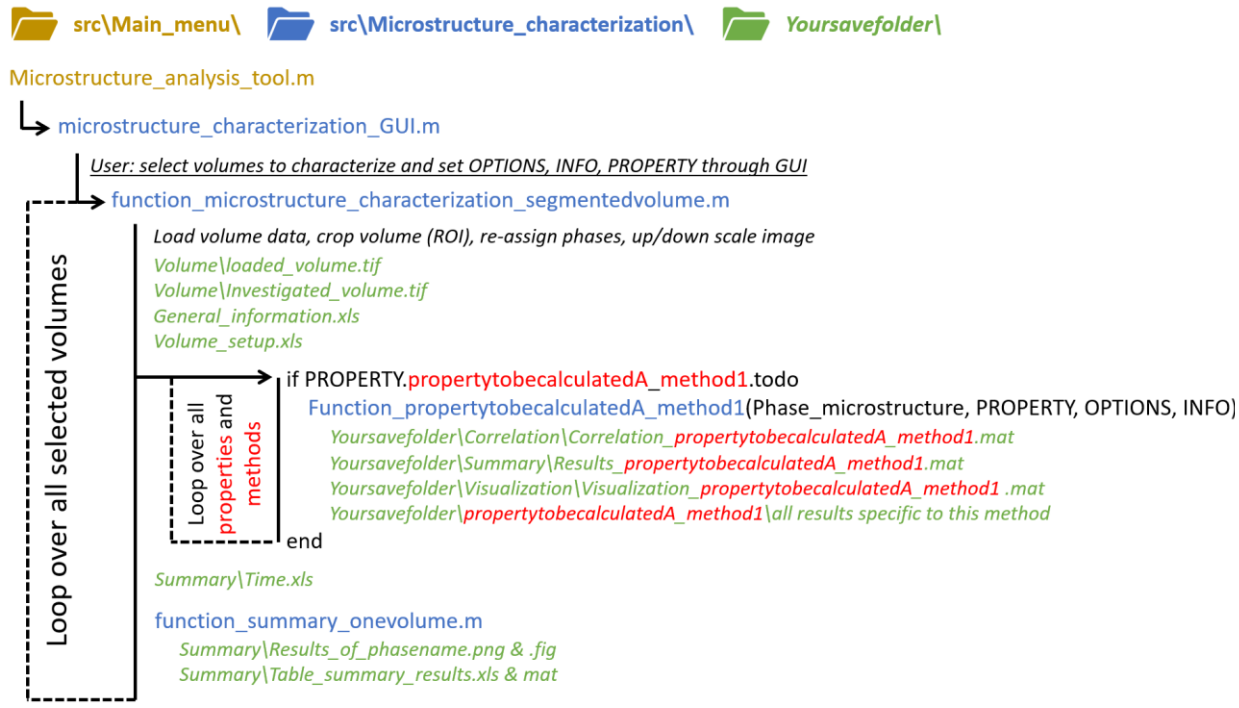


Figure VI-3. High-level organization chart of the characterization module. Green files are outputs. In addition to the MATLAB m files in src\Main_menu and src\Microstructure characterization, other files common with other modules are located in src\Miscellaneous.

The fields of the MATLAB structures OPTIONS, INFO, and PROPERTY are determined in microstructure_characterization_GUI.m, by the user interactions with the module GUI. OPTIONS controls save, format and figure options. Most OPTIONS fields are common for all volumes, except those for loading and saving locations. INFO contains information about phases name and code, directions name, voxel size and region of interest, in addition to general comments about the investigated volume. INFO fields are different for each volume. Lastly, PROPERTY

controls all property parameters, RVE analysis parameters, and voxel size dependence analysis parameters. PROPERTY fields are common for all volumes. Essentially, the GUI is used to fulfill these three structures. The file Function_propertytobecalculatedA_method1 is detailed in the next section.

b. Typical file organization

To explain the organization of a Function_propertytobecalculatedA_method1.m, we will use as template the file Function_particle_size_CPSD.m used to calculate particle size through a continuum approach and using a spherical assumption. In this section we only focus on the file organization, not the algorithms (explained in §VI-6).

- Comments lines immediately after the first one indicate the two possible syntax to call the function: through the toolbox, or as a standalone. The variable 'Phase_microstructure' is the 2D or 3D array representing the segmented volume to be quantified.

- %% DEFAULT VALUES section will set the structures OPTIONS, INFO, and PROPERTY with default values in the case the function is called as a standalone.

- %% DATE section is self-explanatory.

- %% FOLDER section creates sub folders, named with the convention propertytobecalculatedA_method1. Folder separator "\" and "/" are selected depending on the code is running on Windows or not.

- %% INFO section gets the domain size (in voxels) of the microstructure. If the microstructure data is an image (2D array), we enforce the 3rd dimension size to be 1, as code often asks for this value. Number of phases, of voxels and voxel size is also extracted from the INFO structure.

- %% INITIALIZE RESULTS (USE FOR CORRELATION and VISUALIZATION) section initializes the structure results_correlation and results_visualization that are used in the correlation and visualization module, respectively. The structure results_correlation stores scalar values (e.g., mean particle size) while the structure results_visualization stores 2D or 3D array (e.g. local particle size). While results_correlation is used by every Function_propertytobecalculatedA_method1.m files, results_visualization is only used when results can be visualized voxel-wise. Some file, such as Function_Volume_fractions.m, does not have the results_visualization structure.

- %% DISPLAY section prints in the MATLAB command window what property is calculated, and with what method.

- %% ALGORITHM ON WHOLE VOLUME section is specific to each file, although it follows the same organization for every file, as described below with a pseudo code.

```
%% CALCULATION
Get sarting times, both wall clock and CPU.
Initialize one or several arrays that will store main results.
Set parameters for smoothing distribution functions, if any.
```

Distributions functions are calculated in Function_probability_density.m, while smoothing is performed in function_smooth_movingaveragefilter.m, both files located in the src/ Miscellaneous folder. Complete lists of arguments are detailed in the file comments.

```
Loop over each phase.
   Select code of the current phase.
   Create a binary array, =1 for voxel that belong to the current phase, 0 otherwise.
   Call method algorithm on the binary array.
```

Method algorithm are named according to the convention: Function_propertytobecalculatedA_method1_Algorithm.m

```
   Perform side calculations, if any (in this case the particle level description metric).
```

Side calculations are performed outside of the algorithm file to keep the algorithm file short. Besides, since the algorithm is called several times for RVE and voxel size dependence analysis and that you may not need the side calculations for these tasks, it will save time.

```
   Perform basic statistics analysis (minimum, maximum, mean, standard deviation and relative
   standard deviation) if relevant, and store the results in the previously initialized arrays.
   Perform advanced statistics analysis (distribution functions), if relevant, and store
   the results in the previously initialized arrays.
   Store whatever scalar result values you would like to correlate later in the structure
   results_correlation. Store whatever 2D or 3D voxel-wise results, if any, you would like to
   visalualize later in results_visualization.
End loop
Measure elapsed time.
```

Typically, the average and standard deviation are stored in the correlation structure, but you can add other values.

```
%% TABLES
Create table that contains number of voxels, CPU and wall clock time.
```

The time structrure, named Table_time, will be updated with new triplets (voxel number, CPU time, wall clock time) if RVE and/or voxel size dependence analysis will be performed.

```
Create one or several tables for main results.
Create one or several tables for advanced statistics analysis (distribution functions), if any.
```

Tables are stored in new structure named Results_propertyormethodname (here Results_cpsd). New table will be stored in this structure as the code progresses.

```
%% SAVE TABLES
Tables are saved in a excel file (one table per sheet) in the save folder.
%% DISPLAY
Tables are printed in the MATLAB command windows
```

- %% ADDITIONAL RESULTS ON THE WHOLE VOLUME section is specific to each file, although files are sharing same pieces of codes:

```
%% CUMULATIVE AND DISTRIBUTION FUNCTIONS PLOT
```

Is used for plotting distribution functions (see function_probability_distribution_figure.m in src/ Miscellaneous). Minimum adjustments are required for each file. Figures are saved in the save folder.

```
%% ALONG DIRECTIONS
```

Is used for plotting properties results as a function of location (see Function_along_direction.m in src/ Miscellaneous). Figure are saved in the save folder, while results are stored in a table, within the structure Results_propertyormethodname. Minimum adjustments are required for each file.

```
%% PARTICLE DIAMETER MAP
```

This section is specific to this file. It will create 2D slices of the particle size and save the figure.

- %% IMAGE RESOLUTION SENSITIVITY ANALYSIS section automates the voxel size dependence analysis: microstructure is up/down scale several times according to the user parameters selected in the GUI (stored in the structure PROPERTY), and properties are plotted as a function of the voxel size. The array property_voxelsizedependence is a 2D or 3D array, with rows being the results for each voxel size (1st column is the voxel size, subsequent columns are phase results). The third axis is used to store additional results. Image scaling is performed with function_scale_array.m in src\Microstructure_characterization.

```
%% EXTRAPOLATION TO 0 nm
```

Property is extrapolated to a 0 nm voxel resolution, using a polynomial of order 'interpolation_voxelsize_order'. The extrapolated value is stored in the results_correlation structure so that user can choose later to perform correlation both with non-extrapolated and extrapolated values.

```
%% MANAGING RESULTS
```

Results are organized in table, one table per result.

```
%% DISPLAY TEXT RESULTS
```

Tables are printed in the command windows.

```
%% SAVE RESULTS
```

Tables are stored within the structure Results_propertyormethodname. Results are saved in an excel file Propertyname_methodname_voxel_size_dependence.xls (here, D50_Cpsd_voxel_size_dependence.xls).

```
%% FIGURES
```

The function Function_create_figure_voxelsizedependence is called for each parameter to be plotted as a function of voxel size.

```
%% FIGURES 2 (DISTRIBUTION)
```

The function_probability_distribution_size_figure is called for each parameter distribution function to be plotted for different voxel size. Not every method uses it.

- %% REPRESENTATITVE VOLUME ELEMENT (RVE) ANALYSIS section performs representative volume element analysis. The whole section is repeated for each RVE case (A-E) selected by the user. Despite its complexity, this part is very uniform among characterization files, most changes consist in changing structure field to match with the current property/method.

The structure PROPERTY contains the RVA parameters. The first subsection fulfills the structure RVEparameters from PROPERTY, in a similar way for all files: simple replace the field with the current property/method in PROPERTY.particlesize_cpsd.RVE. Results are stored within a subfolder, one for each RVE case. Function_get_Subdomains determines bounds of all subdomains for the RVE case considered (A-E), and the subvolume information is stored in the structure RVE.info.

```
%%  ALGORITHM
```

Method algorithm is called for each subvolume. Results are stored within the variable Property_eachsubdomain. When several results are saved, additional dimension are added to the variable (see file function_connectivity.m for example)

```
%%  STATISTICAL ANALYSIS and RVE SIZE
```

The result analysis is perfomed in the file Function_subdomains_statistical_analysis.m (src\Microstructure_characterization). Property_subdomains_statistics contains for each RVE size, the number of subvolume and basic statistics (mean, extremums, standard deviation). RVE size are stored withing the variable Size_RVE. The RVE size can be found inferior, equal, or superior to a domain size, respectively if subdomains sizes are, too large, adequate, or too small. Two size are considered, the first (k_size=1) corresponds to the cubic root of the RVE size. The second (k_size=2) corresponds to the square root of the RVE 2D field of view when one dimension if fixed (case C). RVE size are stored in the results_correlation structure, so that RVE size can be correlated with other metrics.

```
%%  Function_subdomains_manage_results
```

The file Function_subdomains_manage_results.m (src\Microstructure_characterization) creates table to organize the many results from the RVA. RVE.results contains result for each subdomain, RVE.phase.statistics contains RVA statistics results, and RVE.sizeX contains RVE size. Lastly RVE.parameters contains all parameters of the current RVA case.

```
%%  TEXT DISPLAY AND SAVE RESULTS
```

This section is handled by the file Function_subdomains_display_and_save.m (src\Microstructure_characterization). Statistics (table RVE.phase.statistics) and RVE size (table RVE.sizeX) are printed in the command window. All fields of the structure RVE are then saved in an excel file, propertyname_RVE.xlsx. RVA results are saved in a different subsolfer for each RVA case (A-E).

```
%%  FIGURES
```

Once figure parameters are entered the file Function_create_figures_RVE.m (src\Microstructure_characterization) is called to print and save the RVE result figures. Only fields propertynameunit and Wholevolume_results must be changed for another method.

Once all RVE cases have been calculated, the structure RVE is stored in the result structure Results_cpsd.

- %% ENDING FUNCTION

Calculation time is plotted and saved in excel file using the function function_time_figure.m (src\Microstructure_characterization).

7. User-modification of the characterization module

Please follow instructions below to incorporate new algorithms or modifying existing ones.

a. Integrate a new algorithm in the toolbox

There are several advantages incorporating your own algorithm in the characterization module: statistical analysis (distribution function), graded analysis (parameter evolution along direction), voxel size dependence analysis and representative volume element analysis are already coded in the toolbox. Furthermore, it provides visualization and correlation tools. Therefore, you can focus on your core algorithm and let the toolbox doing all these generic tasks.

i. Prepare your algorithm

Incorporating an algorithm within any GUI required first to identify all the parameters. A GUI is essentially a user-friendly way to enter algorithm parameters. Once your algorithm input/output architecture is fixed, it is then worth it to integrate within a GUI. First, you will have to convert your code in a function. To be coherent with the other toolbox algorithms and keep short enough the code, please use the minimum number of parameters in your function. Also, please name this function using the naming convention: Function_propertytobecalculated_method_Algorithm.m, for instance:

```
[Particle_size] = Function_particle_size_CPSD_Algorithm(binary_phase);
```

Copy the new function Function_propertytobecalculated_method_Algorithm.m in the folder \Microstructure_analysis_toolbox\src\Microstructure_characterization\.

ii. Create function that will call your algorithm in the toolbox

You then need to create a new function, hierarchically above of the one you have created in the previous paragraph, that will call your algorithm and integrate it with all the side calculations, such as RVE. Please name it Function_propertytobecalculated_method.m, for instance:

```
[] = Function_particle_size_CPSD(Phase_microstructure, PROPERTY, OPTIONS, INFO)
```

The arguments are (almost) always 'Phase_microstructure', 'PROPERTY', 'OPTIONS', and 'INFO'. It is recommended to use existing functions as a template. If your new algorithm calculates a scalar (such as volume fraction), use Function_Volume_fractions.m as template. If the property is anisotropic, use Function_Tortuosity_factor_taufactor.m as template. If you calculate a pixel-wise information (such as particle size), use Function_particle_size_CPSD.m as a template. You should consider the function could be used as a standalone. That's the purpose of the very first section (%% DEFAULT VALUES), that allows user to call directly the function from the MATLAB command window, skipping the whole GUI. Open Function_Specificsurface_direct.m to see an example. The standalone syntax to call it is Function_Specificsurface_direct(array, voxelsize, corrective_factor), instead of Function_Specificsurface_direct(array, PROPERTY, OPTIONS, INFO) when called through the GUI. You will have to modify the following lines in to adapt it to your code:

```
if nargin == 3 % Minimum number of parameter + volume
   voxel_size = PROPERTY;
   corrective_factor_specificsurface = OPTIONS;
   clear PROPERTY OPTIONS
   […]
   % Parameter
   PROPERTY.specificsurfacearea_directmethod.correctivefactor =
corrective_factor_specificsurface;
else % Incorrect number of argument
   disp 'Error calling Function_Specificsurface_direct. Wrong number of argument.'
   help Function_Specificsurface_direct
end
```

Parameter specific to an algorithm are stored within the PROPERTY structure, with the naming convention: PROPERTY.propertyname_methodname.parametername. Then you will have to adapt the rest of the function to your code. If you have selected the good template, few changes will be required, mostly variable names. Copy the new function Function_propertytobecalculated_method.m in the folder \Microstructure_analysis_toolbox\src\Microstructure_characterization\. Pay special attention to the structure results_visualization and results_correlation as they are doing the link respectively with the visualization and the correlation module, see sections c and e. You will have also to modify the file function_summary_onevolume.m, see section d.

iii. Update the GUI

The last step consists in updating the GUI itself. Open microstructure_characterization_GUI.m. Copy-paste the following codes (choose the one that correspond to your template) and replace the text. First the PROPERTY default parameters:

```
% Default parameters for specific interface area (direct method)
PROPERTY.specificinterfacearea_directmethod.todo = true;
PROPERTY.specificinterfacearea_directmethod.voxel_size_dependence.todo = true;
PROPERTY.specificinterfacearea_directmethod.voxel_size_dependence.voxel = [2 3 4 5];

PROPERTY.specificinterfacearea_directmethod.number_RVE = 1;
PROPERTY.specificinterfacearea_directmethod.RVE(1).name = 'Independant subvolumes of same size +
keep initial aspect ratio (A)';
PROPERTY.specificinterfacearea_directmethod.RVE(1).savename = 'SubInitialAR';
PROPERTY.specificinterfacearea_directmethod.RVE(1).type = 'A';
PROPERTY.specificinterfacearea_directmethod.RVE(1).divisions = [2 3 4 5];
PROPERTY.specificinterfacearea_directmethod.RVE(1).subs2 = true;
```

```
PROPERTY.specificinterfacearea_directmethod.RVE(1).subs4 = true;
PROPERTY.specificinterfacearea_directmethod.RVE(1).Aspectratio = 'n/a';
PROPERTY.specificinterfacearea_directmethod.RVE(1).Constantdirection = 'n/a';
PROPERTY.specificinterfacearea_directmethod.RVE(1).Growthdirection = 'n/a';
PROPERTY.specificinterfacearea_directmethod.RVE(1).Growthperstep = 'n/a';
PROPERTY.specificinterfacearea_directmethod.RVE(1).firstuniquevolume_size='n/a';
PROPERTY.specificinterfacearea_directmethod.RVE(1).firstuniquevolume_unit='n/a';
PROPERTY.specificinterfacearea_directmethod.RVE(1).Growthrelativeto = 'n/a';
PROPERTY.specificinterfacearea_directmethod.RVE(1).threshold_std = default_threshold_std;
PROPERTY.specificinterfacearea_directmethod.RVE(1).threshold_numbersubvolumes =
defaut_threshold_numbersubvolumes;

default_correctivefactor_spidirect = 2/3;
PROPERTY.specificinterfacearea_directmethod.correctivefactor =
default_correctivefactor_spidirect;
```

Create tab, with name following the convention property name (method):

```
tab_specificsurfacearea = uitab('Parent', table_group_1,'BackgroundColor',background_tab,
'Title', 'Specific surface area (direct) options');
```

Select a color for the tab. Method that calculate the same property should use the same color.

```
tab_specificsurfacearea.ForegroundColor  =[0 0.3922 0];
```

Then copy-paste the whole section used to populate the template tab, for instance for the specific surface area calculated with the direct method, that all these lines:

```
%%
%% TAB SPECIFIC SURFACE AREA (DIRECT)
%%
[…]
%%
%% TAB the next section (method)
%%
```

You will have to rename all GUI and callback objects. The GUI object below the annotation lines control the parameter for this method, and then differ from method to method. You may have to erase or create GUI object to adjust them to your algorithm parameters. Your algorithm is now integrated within the toolbox, congratulations!

b. Modify an existing algorithm

If you want to modify an existing algorithm for whatever reasons (e.g., algorithm is too slow), you can implement your modifications quite easily. Method algorithms are named using the following convention: 'Function_propertytobecalculated_method_Algorithm.m'. For instance, the algorithm to calculate the particle size with a continuum particle size (c-PSD) method is 'Function_particle_size_CPSD_Algorithm.m'. You do not need to modify other files used for plotting, RVE analysis or other tasks. The basic input/output structure is:

```
[Particle_size] = Function_particle_size_CPSD_Algorithm(binary_phase)
```

With binary phase a 3D array, with voxels being equal to 1 for the investigated phase, and 0 for the complementary volume. You should use the minimum number of parameters to keep the algorithm file short enough to be readable, and to be coherent with the other toolbox's algorithms. For instance, in this case the voxel size is not even used (the algorithm is adimensional with a unit

voxel size). The script just above the algorithm Function_particle_size_CPSD.m scales afterwards the particle size:

```
Function_particle_size_CPSD(Phase_microstructure, PROPERTY, OPTIONS, INFO)
   […]
   [Particle_size] = Function_particle_size_CPSD_Algorithm(binary_phase); % Call algorithm
   […]
   Particle_size = Particle_size*voxel_size/1000; % nm->um
```

If your script uses the same parameters in the same manner of the legacy algorithm, no other modifications should be required. If you use less parameters, you must manually comment the lines that use such superfluous parameters. If you use more parameters, you can either hard coded them in the script, or you can control their value through the GUI. An example of GUI-controlled parameter can be found for the specific surface area calculation, for which a corrective factor is used. Open the file microstructure_characterization_GUI.m and search for the line

```
PROPERTY.specificsurfacearea_directmethod.correctivefactor = default_correctivefactor_spdirect;
```

Parameters are stored within the PROPERTY structure, following the syntax PROPERTY.properyname_methodname.parameter (note that parameters shared among various algorithms, such as the voxel size, are stored in the INFO structure). The parameter value is controlled through a edit box MATLAB GUI object:

```
Edit_spdirect_correctivefactor = uicontrol('Parent', tab_specificsurfacearea,'Style',
'edit','FontSize',font_size_small_GUI,'FontName',font_name_GUI,'String',default_correctivefactor_
spdirect,'BackgroundColor','w','HorizontalAlignment','center','Units','normalized','Position',
[pos_x_start 0.35-0.3*pos_delta_y 3.5*pos_delta_x 0.3*pos_delta_y],'Callback',
@edit_spdirect_correctivefactor_Callback);
   function edit_spdirect_correctivefactor_Callback(~,~)
      PROPERTY.specificsurfacearea_directmethod.correctivefactor =
str2double(Edit_spdirect_correctivefactor.String);
    end
```

You will have to use a similar code to add your new parameter.

c. Save a 3D array to be used in the visualization module

You can easily save new results to be later visualized through the 'Microstructure and results visualization' module. First, let us have a look on how existing array results are currently saved, for instance the particle size calculated with the spherical assumption method. To do so, open the file 'Function_particle_size_CPSD.m' and search for the line:

```
for current_phase=1:1:number_phase
   […]Particle_size is calculated for the current phase
   results_visualization(current_phase).Particle_diameter_CPSD = Particle_size;
end
```

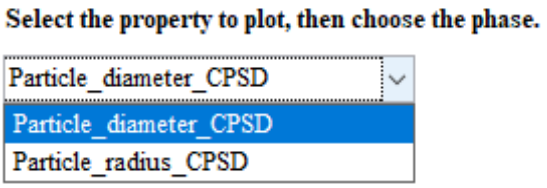
Here, 'Particle_size' is a 3D array with 'Particle_size(x,y,z)' the diameter of the phase numbered 'current_phase'. As a reminder, 'current_phase' is the MATLAB index number of the phase, starting from 1, and not the phase code itself. At the end of the script, the 'results_visualization' structure is saved as:

```
save([Current_folder 'Visualization_particlediameter_CPSD'],'results_visualization','-v7.3');
```

Let us say we want to visualize also the particle radius. To do this, the only modification you would have to perform is to add the line in the 'Function_particle_size_CPSD.m' file:

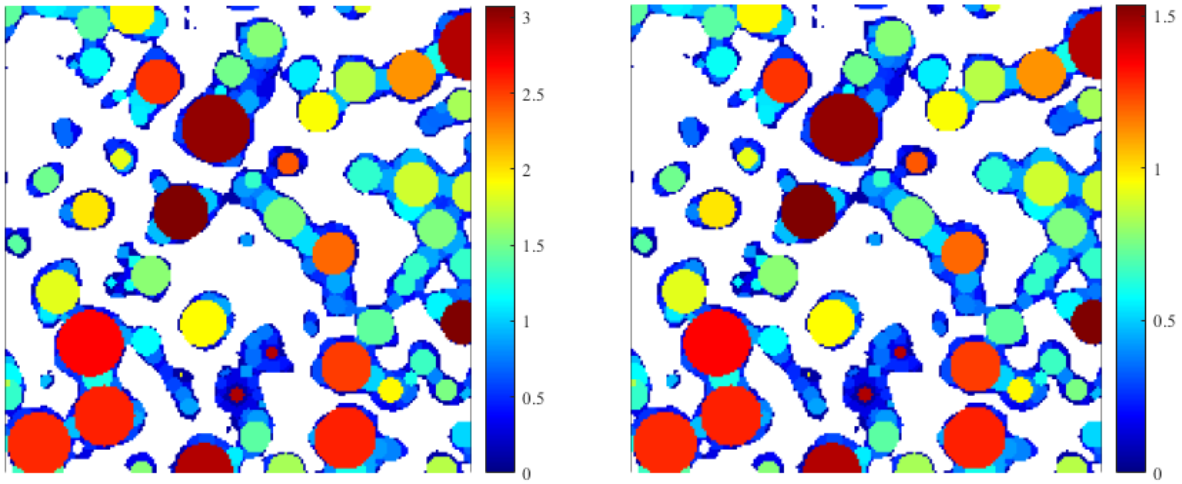
```
for current_phase=1:1:number_phase
   […]Particle_size is calculated for the current phase
   results_visualization(current_phase).Particle_diameter_CPSD = Particle_size;
   results_visualization(current_phase).Particle_radius_CPSD = Particle_size/2;
end
```

Note that you are free to name the new field (in this example, Particle_radius_CPSD) as you wish. Although, to keep syntax coherent, the recommended field name should be written as 'parametername_methodname'. Now, run the microstructure characterization toolbox, make sure that 'Calculate particle size with C-PSD method' is checked in the GUI for this example. Once finished, run the microstructure visualization toolbox, go to tab 'Volume' and click on 'Choice 2: import a volume result' and select the root folder of your calculation. The script will read all the fields of the structure results_visualization for you to choose. You should then have the following choice:



New visualization choice 'Particle_radius_CPSD' is now available in the popup menu.

Note that the new choice corresponds to the new structure field added to results_visualization. Select the phase, and then plot it.



Visualization with (left) 'Particle_diameter _CPSD' and (right) 'Particle_radius_CPSD' option. Note the difference in the colormap scale.

d. Save a new result in the summary table

We will keep with the radius example. Still in the file 'Function_particle_size_CPSD.m' add the line:

```
Table_cpsd =
table(INFO.phasename,PSD_results(:,1),PSD_results(:,2),PSD_results(:,3),PSD_results(:,4),PSD_resu
lts(:,5),...
    'VariableNames',{'Name' 'Min' 'Mean' 'Max' 'Std' 'Std_percents'});
Table_radius =
table(INFO.phasename,PSD_results(:,1)/2,PSD_results(:,2)/2,PSD_results(:,3)/2,PSD_results(:,4)/2,
PSD_results(:,5),...
    'VariableNames',{'Name' 'Min' 'Mean' 'Max' 'Std' 'Std_percents'});
```

Basically, you need to create a new table and write the minimum, mean, maximum, standard deviation and relative standard deviation of the new result. In this example, we can simply divide by two the diameter result (except for the relative standard deviation which is unchanged). Then save the new table in the result structure as:

```
Results_cpsd.Table_cpsd = Table_cpsd;
Results_cpsd.Table_radius = Table_radius;
```

At the end of the script, the result structure is saved in the summary folder:

```
save([Current_folder 'Results_particlediameter_CPSD'],'Results_cpsd')
```

Now open the file function_summary_onevolume.m and search for the string comparison that correspond to the result file name used above, and add the lines within:

```
if strcmp(current_result,'Results_particlediameter_CPSD.mat')
   […]
   line_table=line_table+1;
   Propertyname(line_table) = {'Equivalent radius'}; % Name of your choice
   Propertymethod(line_table) = {'C-PSD'}; % Name of the method
   T = datamat.Results_cpsd.Table_radius; % Table name
   for current_phase=1:1:number_phase
      array = T{current_phase,2:end};
      PropertyMin(line_table,current_phase) = {num2str(array(1),string_precision)}; % Min
      PropertyMean(line_table,current_phase) = {num2str(array(2),string_precision)}; % Mean
      PropertyMax(line_table,current_phase) = {num2str(array(3),string_precision)}; % Max
      PropertyStd(line_table,current_phase) = {num2str(array(4),string_precision)}; % Standard
deviation
      PropertyUnit(line_table,current_phase) = {'[um]'}; % Unit
      PropertyStdpercent(line_table,current_phase) = {num2str(array(5),string_precision)}; %
Standard deviation in percents of the mean
   end
```

Note that you must be coherent with the structure and table name: they must be the same used in 'Function_particle_size_CPSD.m' (here 'Results_cpsd.Table_radius'). Write the correct unit (here the radius is expressed in micrometers). Now, run the microstructure characterization toolbox, make sure that 'Calculate particle size with C-PSD method' is checked in the GUI for this example. Once finished, open the summary folder of the calculation root folder and open one of the image file. You should see:

Property	Method	Mean	Min	Max	Std	Unit	Std_percents
Equivalent diameter	C-PSD	1.546	0.1	3.562	0.9624	[um]	62.26
Particle level of details	C-PSD	0.9148	n/a	n/a	n/a	[]	n/a
Equivalent radius	C-PSD	0.773	0.05	1.781	0.4812	[um]	62.26

Result summary image has a new line 'Equivalent radius'.

Open the excel file 'Table_summary_results.xls'. The table has been updated too.

	A	B	C	D	E	F	G	H
1	**Property**	**Method**	**Mean**	**Min**	**Max**	**Std**	**Unit**	**Std_percents**
2	Equivalent diameter	C-PSD	1.546	0.1	3.562	0.9624	[um]	62.26
3	Particle level of details	C-PSD	0.9148	n/a	n/a	n/a	[]	n/a
4	Equivalent radius	C-PSD	0.773	0.05	1.781	0.4812	[um]	62.26
5								
6								

Table_summary_results excel sheet has a new line 'Equivalent radius'.

e. Save a result to be used in the correlation module

We will keep with the radius example. Still in the file 'Function_particle_size_CPSD.m' add the line:

```
for current_phase=1:1:number_phase
   […]PSD_results is calculated for the current phase
   results_correlation(current_phase).Particle_diameter_mean_CPSD = PSD_results(current_phase,2);
   results_correlation(current_phase).Particle_radius_mean_CPSD = PSD_results(current_phase,2)/2;
end
```

At the end of the script, the structure is saved in the correlation folder:

```
save([Current_folder 'Correlation_particlediameter_CPSD'],'results_correlation')
```

VII. MICROSTRUCTURE AND RESULTS VISUALIZATION

This module enables the user to visualize 3D data, customize visualization, and save it both in static images and in videos, in a user-friendly way. It is particularly valuable for presenting/sharing microstructure data with people not familiar with tif file and/or visualization software such as ImageJ: instead of sharing a tif file, you can make a video of the microstructure data and incorporate it in a PowerPoint you share. There are currently three different visualization types:

- Visualize one or several 3D tif file (cf. §VII-1).
- Visualize the relevance of a segmentation comparing grey-level and segmented volumes side by side (cf. §VII-2).
- Visualize pixel-wise results (such as particle size) obtained from the microstructure characterization module (cf. §VII-3).

1. Tiff data (grey-level and segmented volumes)

Procedure is illustrated step-by-step in the following figures. The input file format is a 3D (i.e., stack) tif file.

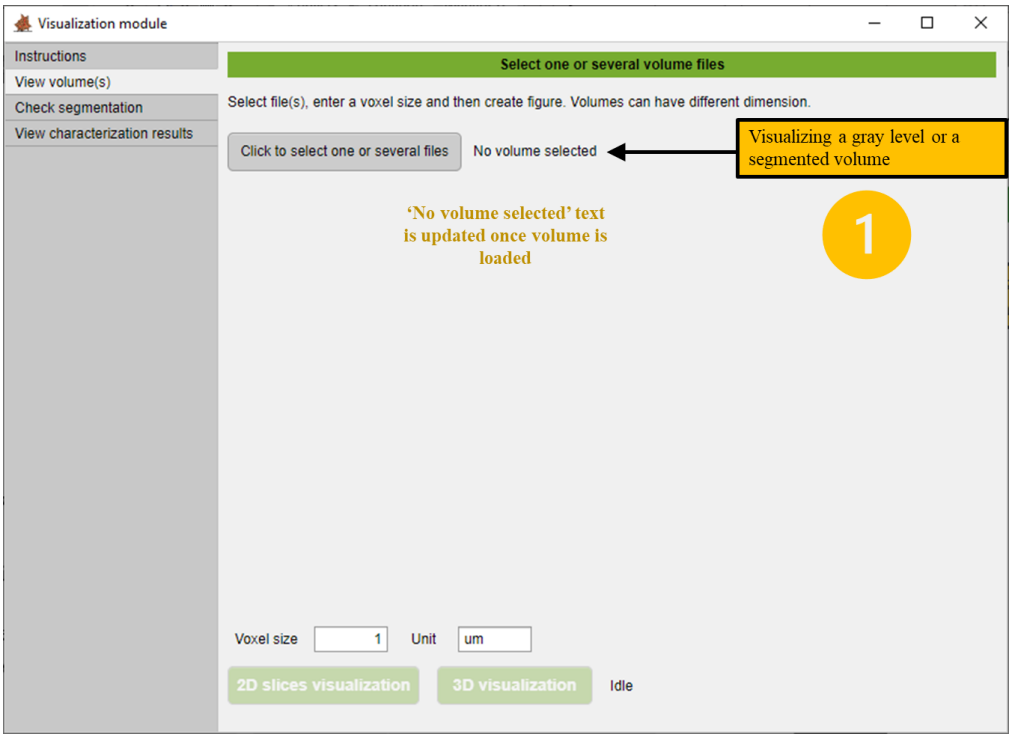


Figure VII-1a. Select volume(s) to visualize. You can select one or several files. To stop selecting new files and start the import process, click on the cancel button. Depending on you have selected one or several files, the GUI is updated accordingly.

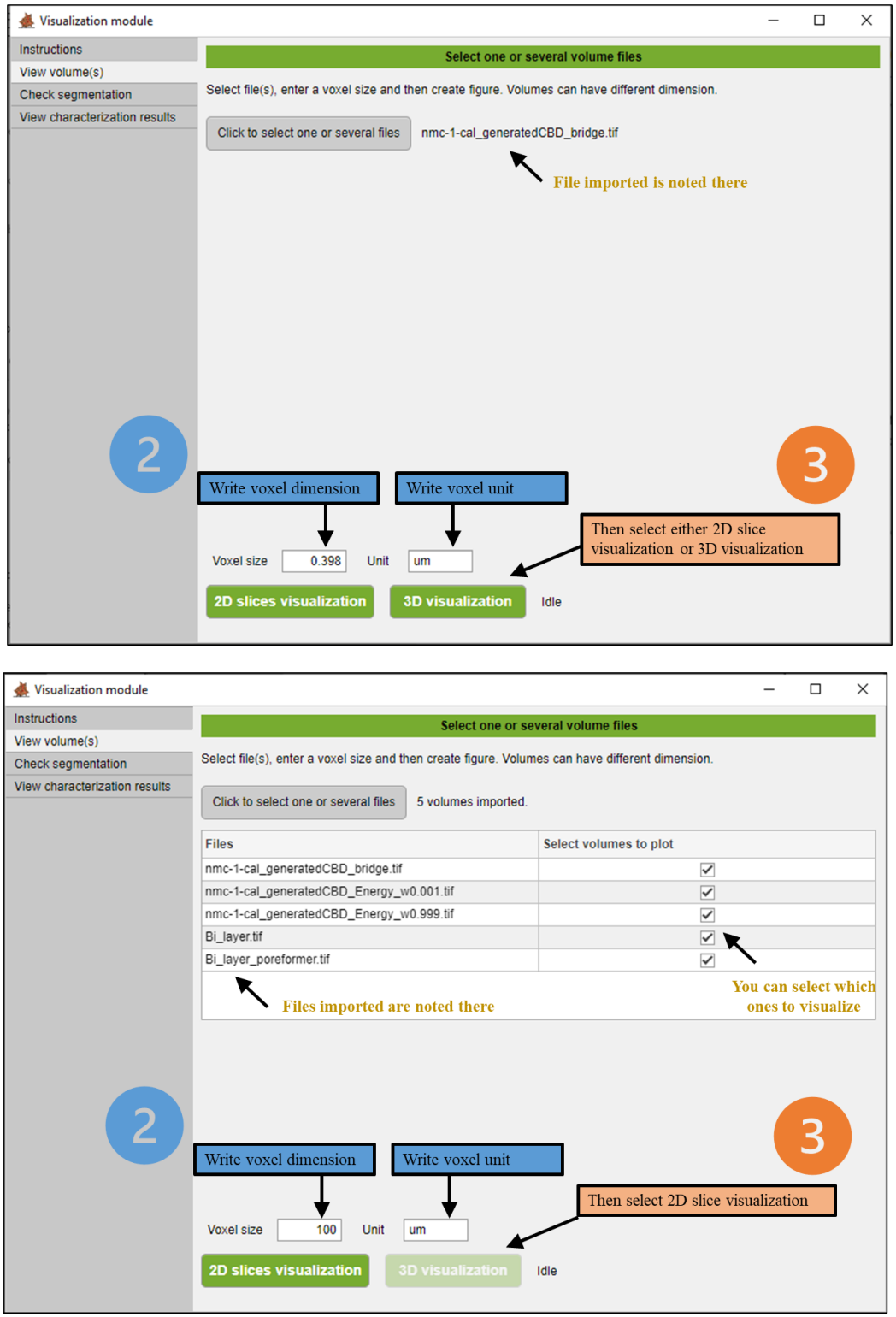
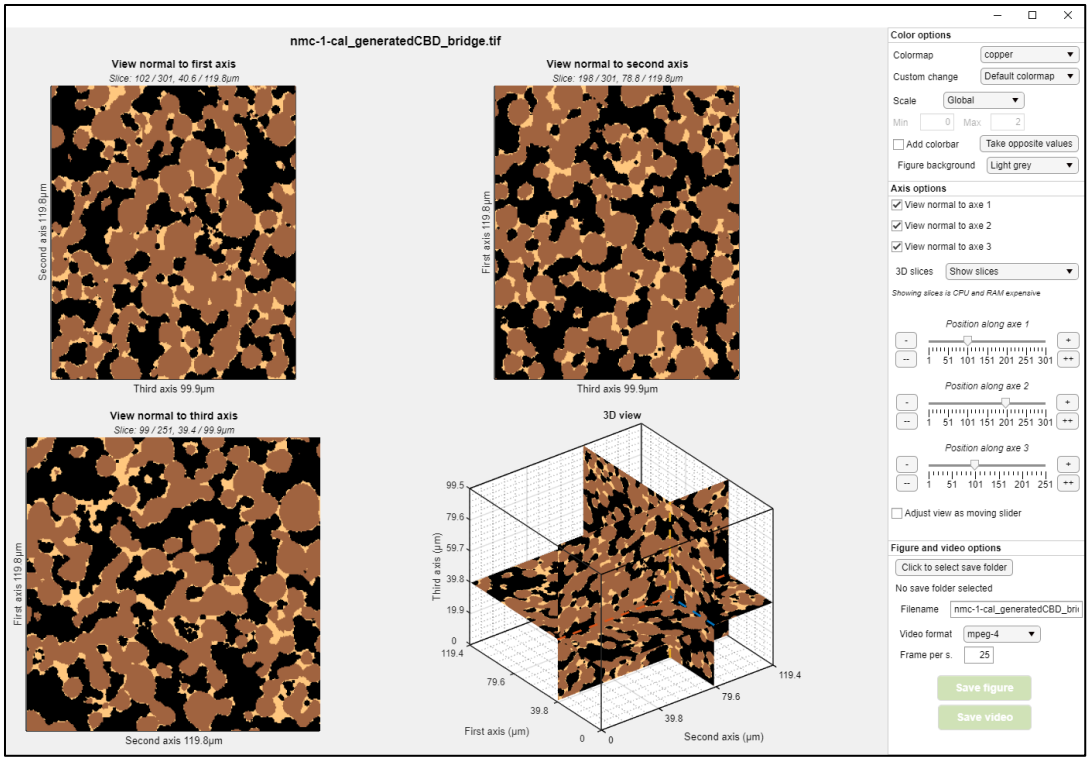


Figure VII-1b. (top) One volume selected and (bottom) several volumes selected. Step 2: select voxel size and click to create figure. If you have selected several volumes (volumes can have different dimensions), you can also choose which one(s) to plot in the dedicated table.



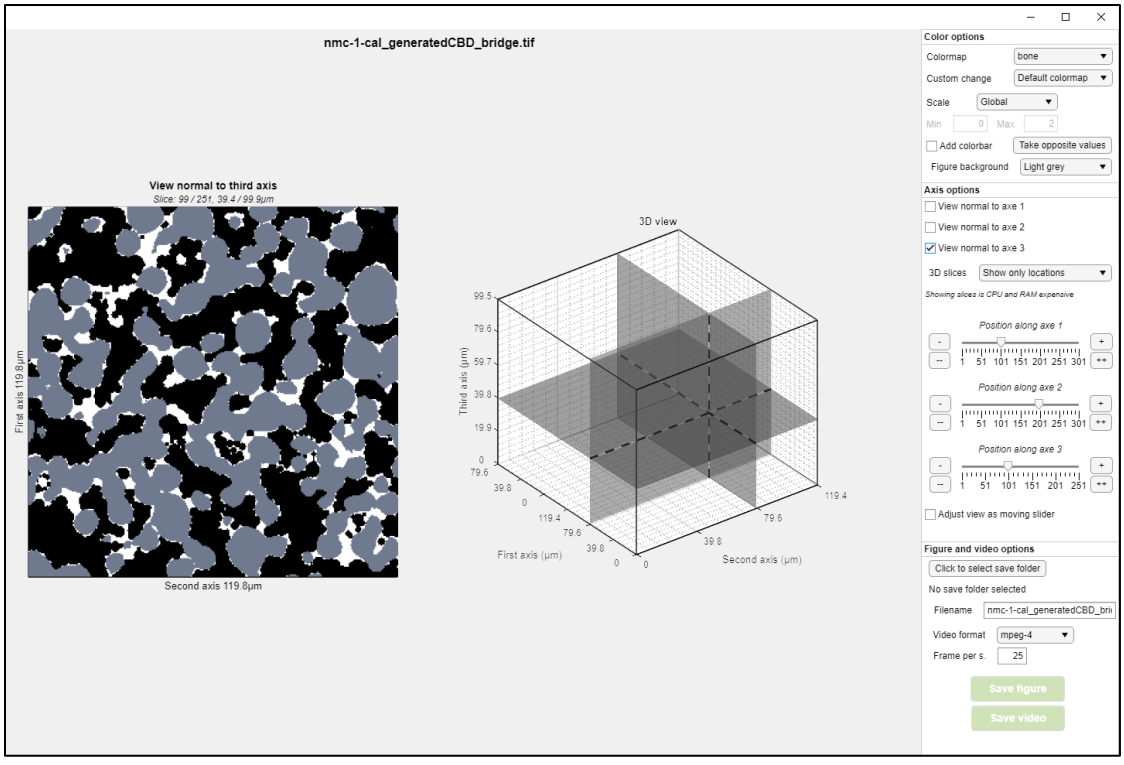


Figure VII-1c. GUI when only one volume is selected (2D slices visualization). You can adjust color map, changes axes, and save figure or video using the right panel. Note that the right

panel is not visible when saving a figure or a video to produce better looking representations to share.

Colormap options are further detailed in §VII-3 but are quite self-explanatory. "Save video" will create an animation with each frame incrementing the slice position, looping on the axis displayed. If you have selected several volumes, then the function *Microstructure_comparison_visualization_interface.m* is called instead (that you can run directly from the command window for ease, see §X for an example) and proposed a much-simplified GUI:

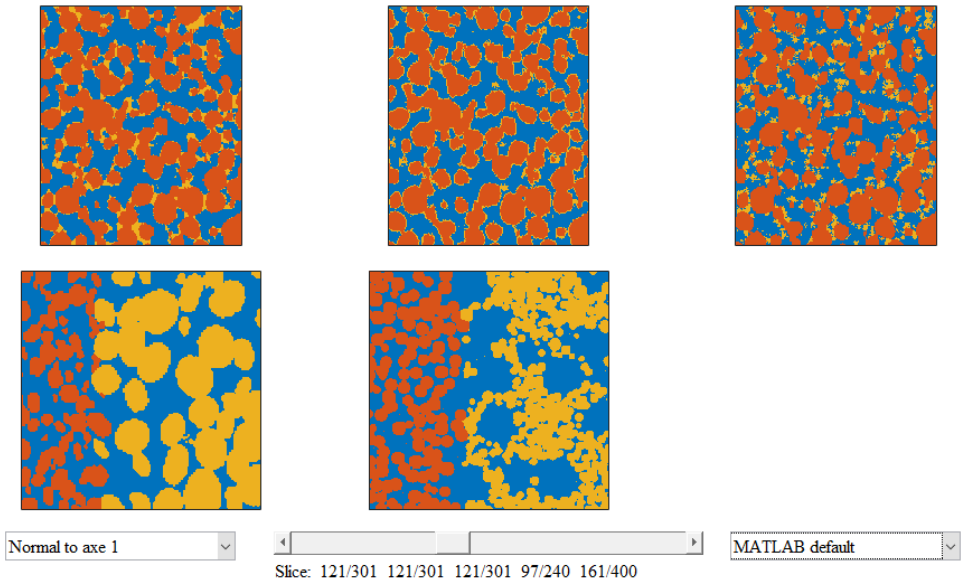


Figure VII-1d. Simplified GUI when several volumes are selected (2D slices visualization). Note that volumes have different dimensions. The slider control both volumes and represent the normalized position along the selected axe. All volumes share the same colormap.

3D visualization, available only if you have selected one volume, will simply call the function *volshow* without much more refinement. For a better 3D rendering (e.g., adding lighting), you can customize the code directly:

```
Fig_3D=figure;
col_ = bone;
volshow(app.uniquevolume,'Parent',Fig_3D,'BackgroundColor','w','Colormap',col_,'Renderer','Volume
Rendering');
```

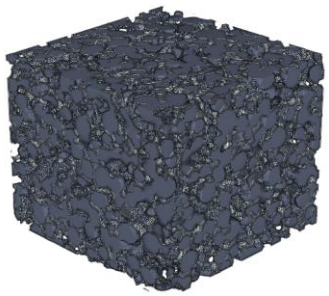


Figure VII-1e. 3D visualization.

2. Comparing grey-level and segmented volumes

Evaluating segmentation relevance often requires a visual inspection. The next tab of this module allows you to compare side by side a grey-level image and its segmented version. Both tif files must share the same dimension. In addition, an overlay or a checkerboard (cf. Fig. VII-2b) representation are available to help checking if the segmentation is a good match for the volume. The user can choose the slice orientation and customize the phase colors as well. Picture and video can be saved, with the GUI automatically removed for a better-looking figure.

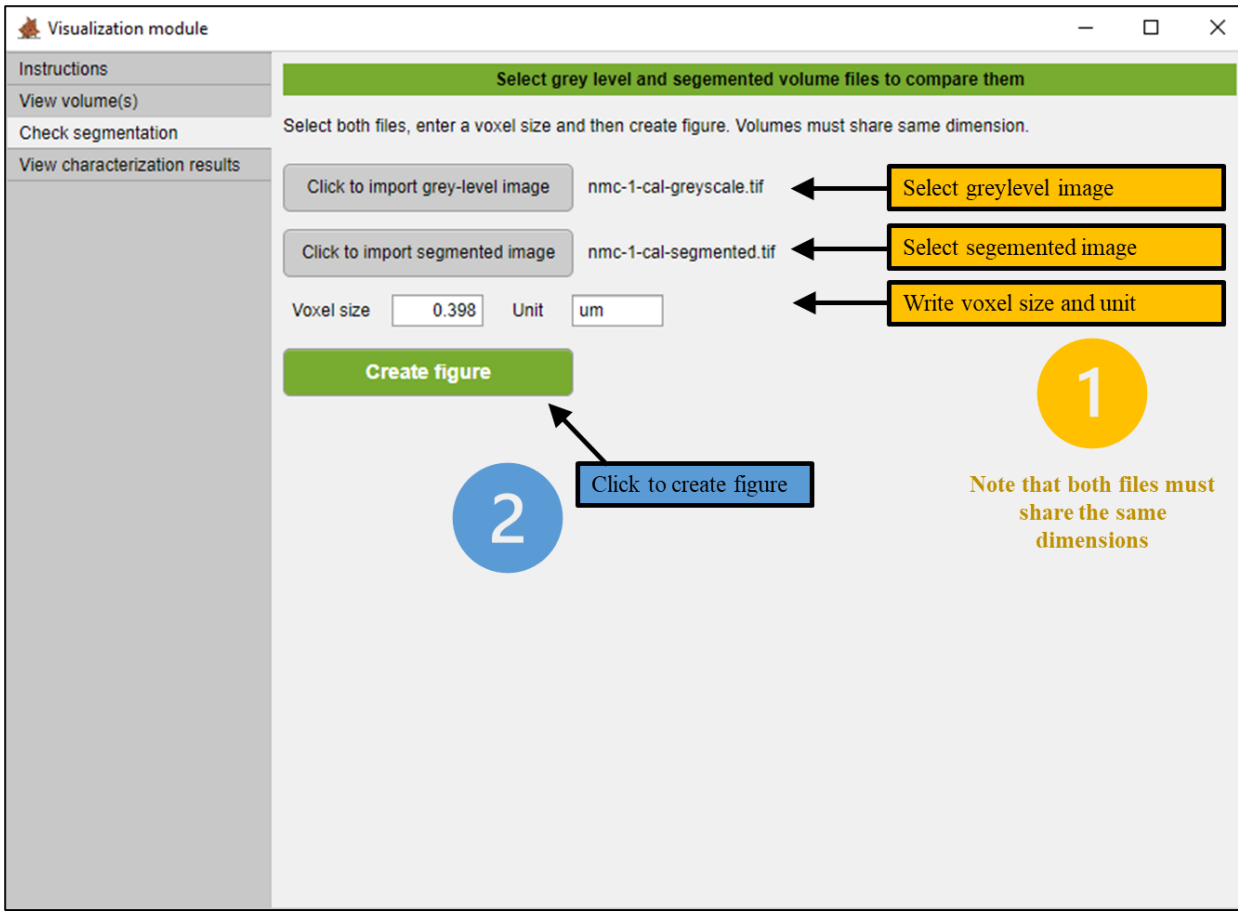


Figure VII-2a. Select grey-level and segmented images.

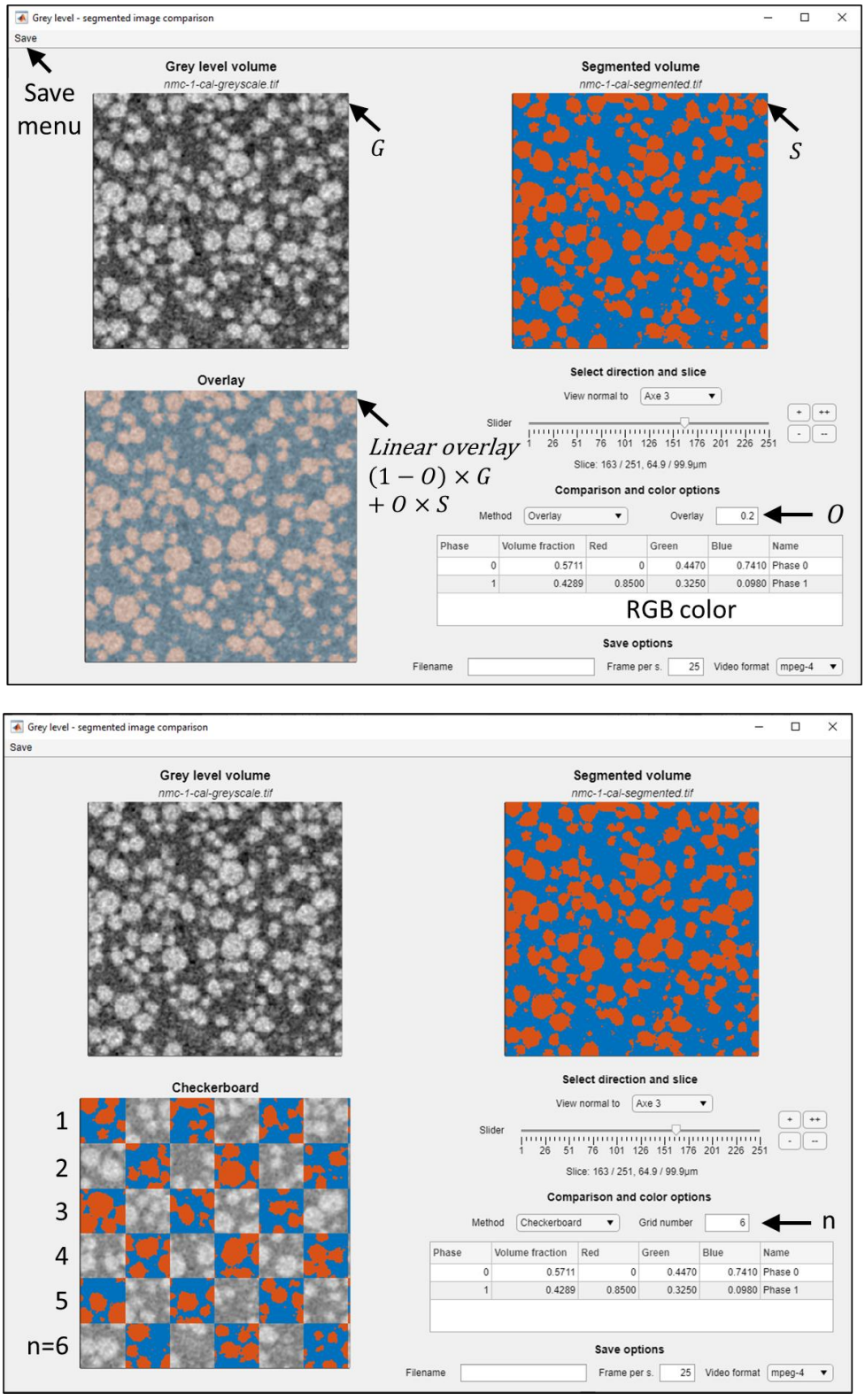


Figure VII-2b. Grey-level and segmented slices side by slide, and (top) linear combination of the two images (bottom) checkerboard representation.

3. Microstructure characterization result

Voxel-wise data is information defined, as indicated by its name, locally for each voxel. For example, particle size, particle label, and cluster connectivity are voxel-wise. Not every microstructure property is voxel-wise: properties defined over the whole volume, such as volume fraction or tortuosity factor, are not.

Unlike the two previous sections, you need first to calculate microstructure properties on a volume with the microstructure characterization module before visualization. Select the save folder you have used previously with the microstructure characterization module. The visualization module will find automatically all results stored specifically for visualization. For more details on how to set what properties can be visualized, see §VI-7b and search for the MATLAB structure results_visualization. Then select the property you want to plot and for which phase, the voxel size and its unit, and also the unit of the property you just have selected and plot it.

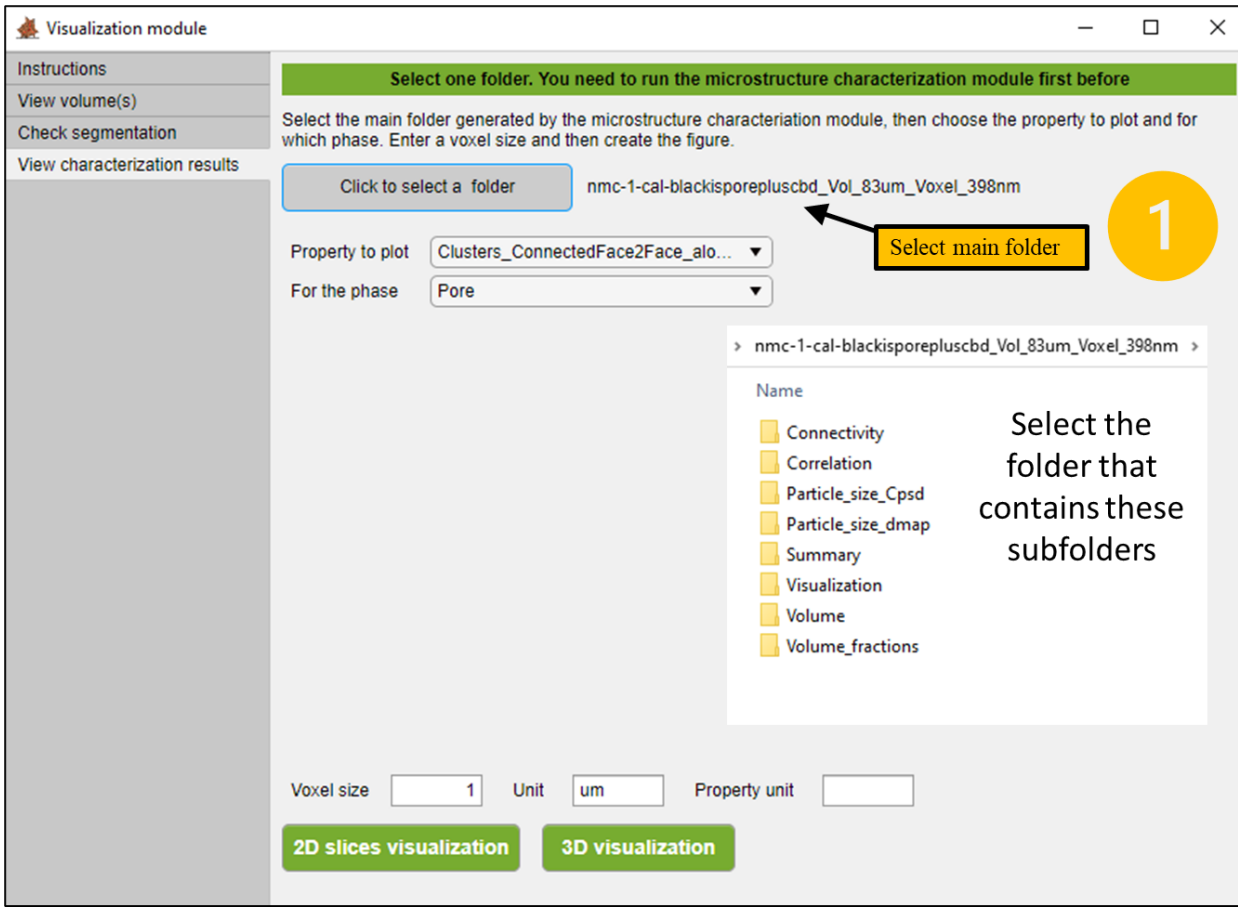


Figure VII-3a. Select folder used to save result in the microstructure characterization module.

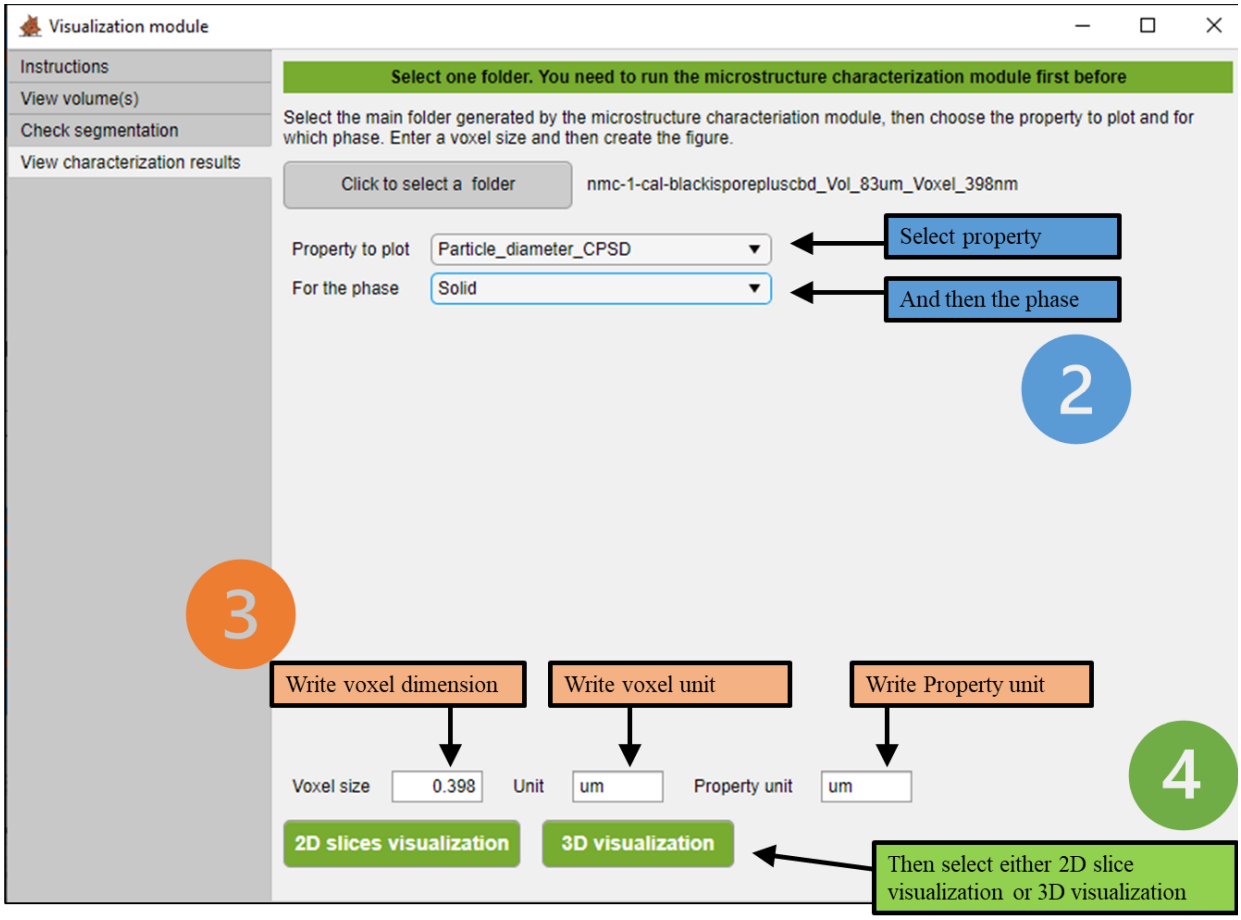


Figure VII-3b. Select what property to visualize and for what phase, then enter voxel size and property unit.

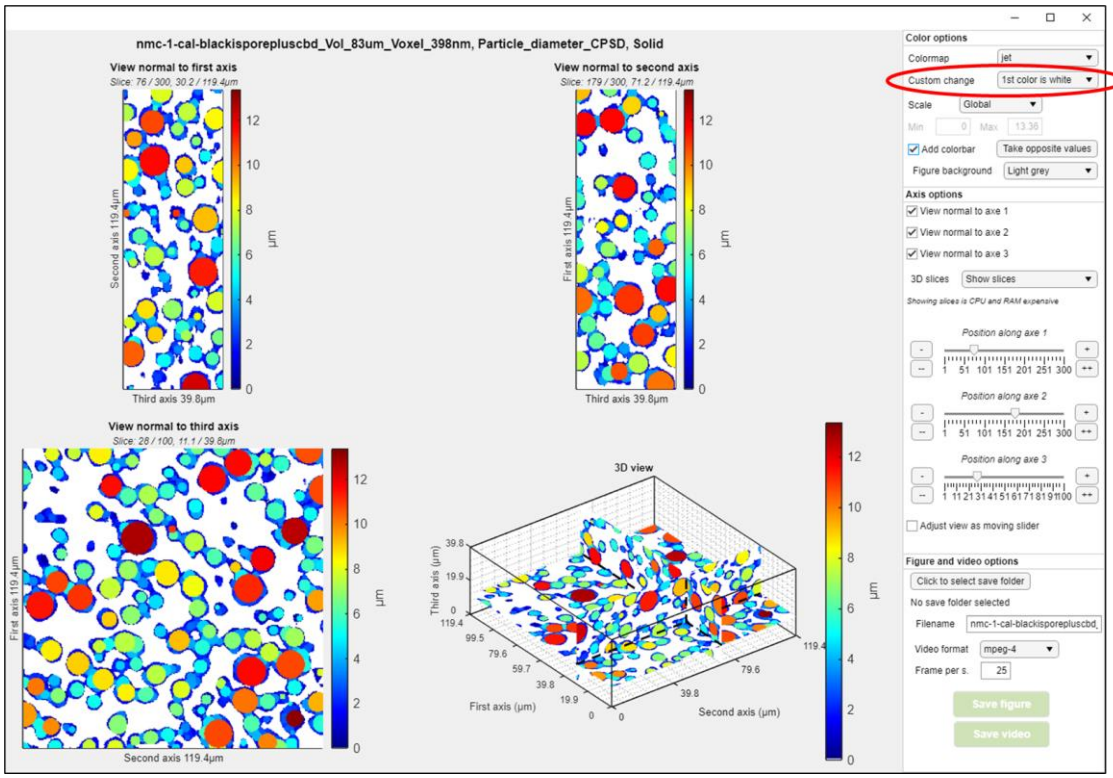
Note that the property name corresponds to the structure field used in the microstructure characterization module, e.g.:

```
results_visualization(current_phase).Particle_diameter_CPSD = Particle_size;
```

The phase names correspond to the ones you have set in the microstructure characterization module GUI (see figure VI-11).

The right panel allows you to customize the figure. For instance, you can force the 1st color of the selected color map to be white (or black) to distinguish the phase selected from the background (cf. Fig. VII-3c). You can also choose to use global, local or custom bounds for the color bar. Similarly as §VII-1 you can save the visualization in a figure or in a video.

The 3D visualization call the function *volshow* without much more refinement (it's up to the user to modify it to its own needs).



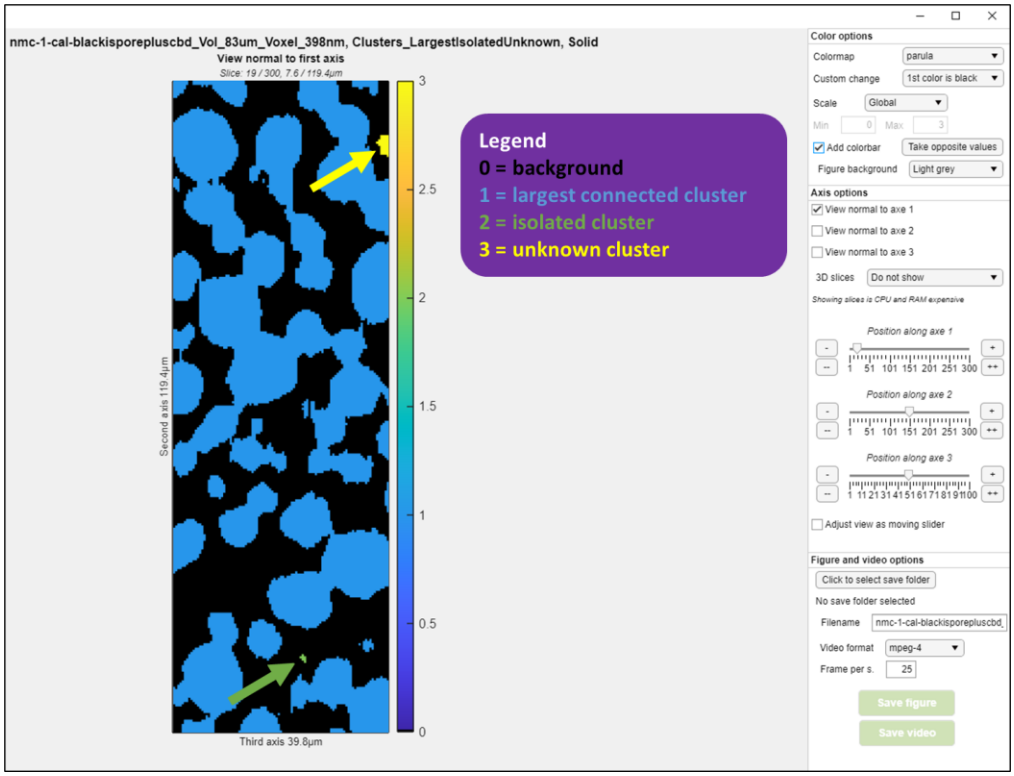


Figure VII-3c. Example of visualization results: (top) particle size, (bottom) cluster connectivity. Figure title is volume name, property name, phase name.

VIII. PROPERTIES CORRELATION

This module enables the user to import various microstructure property values determined on multiple microstructures in order to plot them as a function of each other (i.e., a correlation matrix). It is particularly valuable for summarizing data obtained on various microstructures and to establish correlation between dependent microstructure properties. Note that to generate such property results, the microstructure characterization module must be used first. At least two microstructures must have been characterized before importing their data in the correlation module. Results that can be imported are saved in the subfolder 'correlation' of each result folder generated with the microstructure characterization module (cf. §VI-4d). The user can add new property results to be correlated as indicated in §VI-7e.

Procedure is illustrated step-by-step in the following figures.

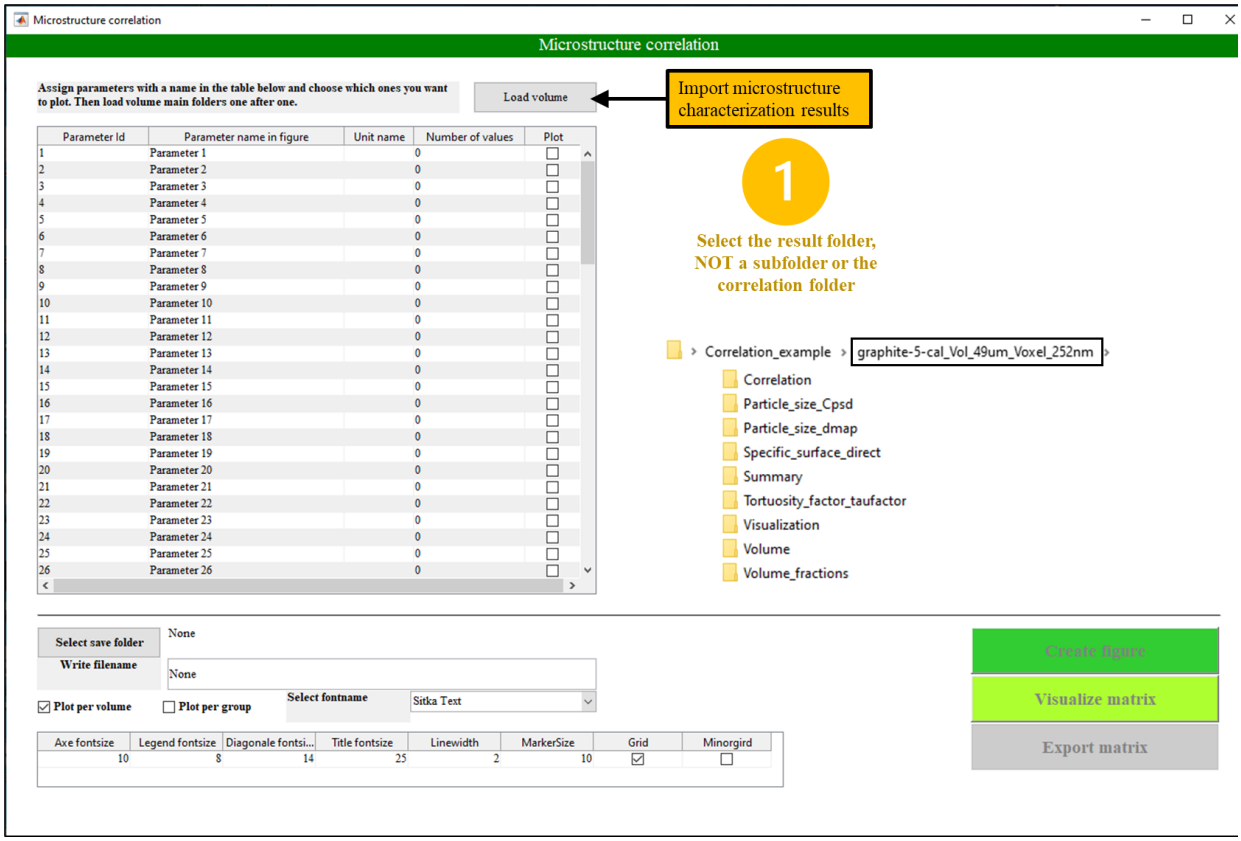


Figure VIII-1. Step 1: select the result folder from a previous microstructure characterization

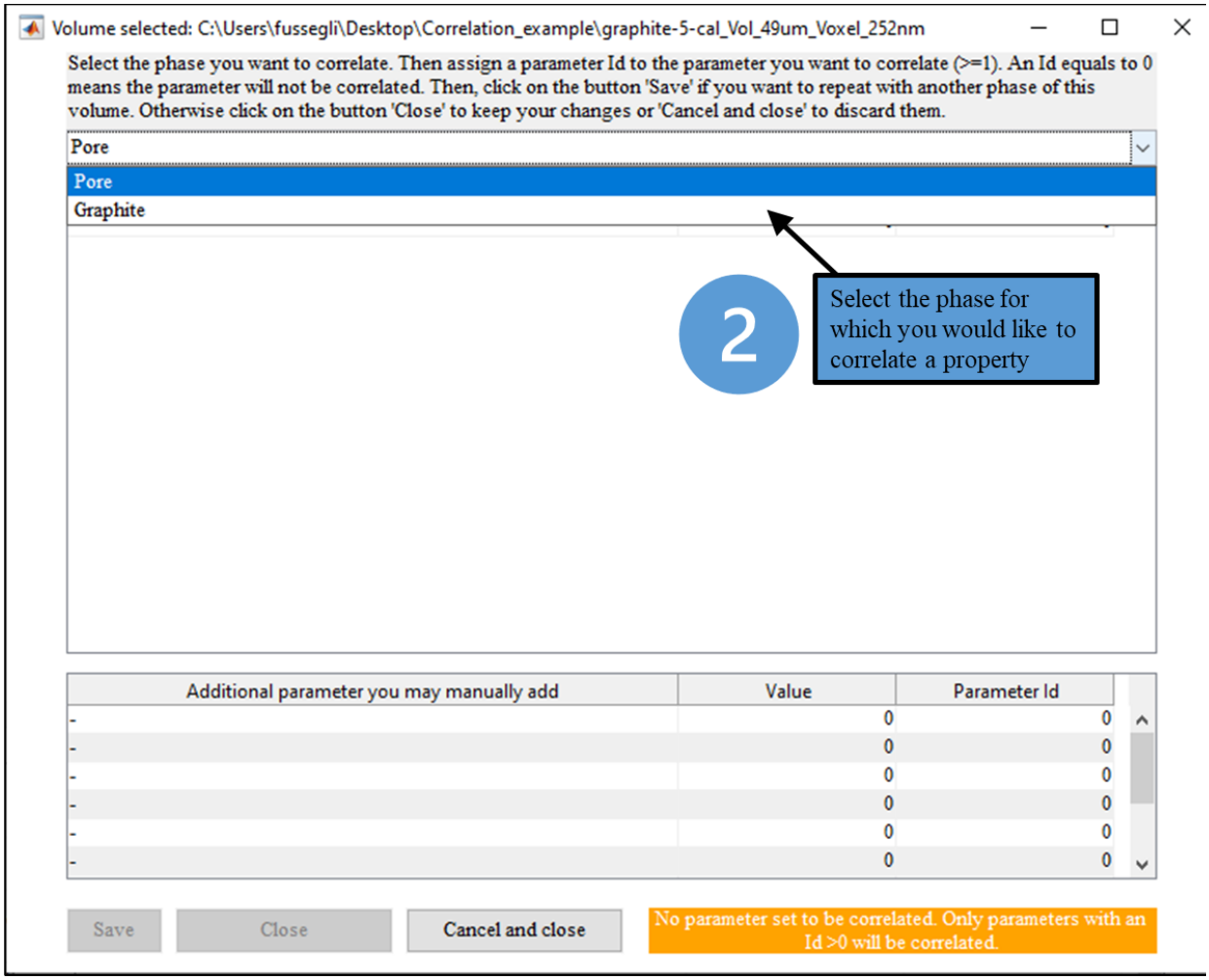


Figure VIII-2. Step 2: select the phase for which you want to import a property result you want to correlate. The phase name (here, 'Pore' and 'Graphite' are those entered in the import tab of the microstructure characterization module).

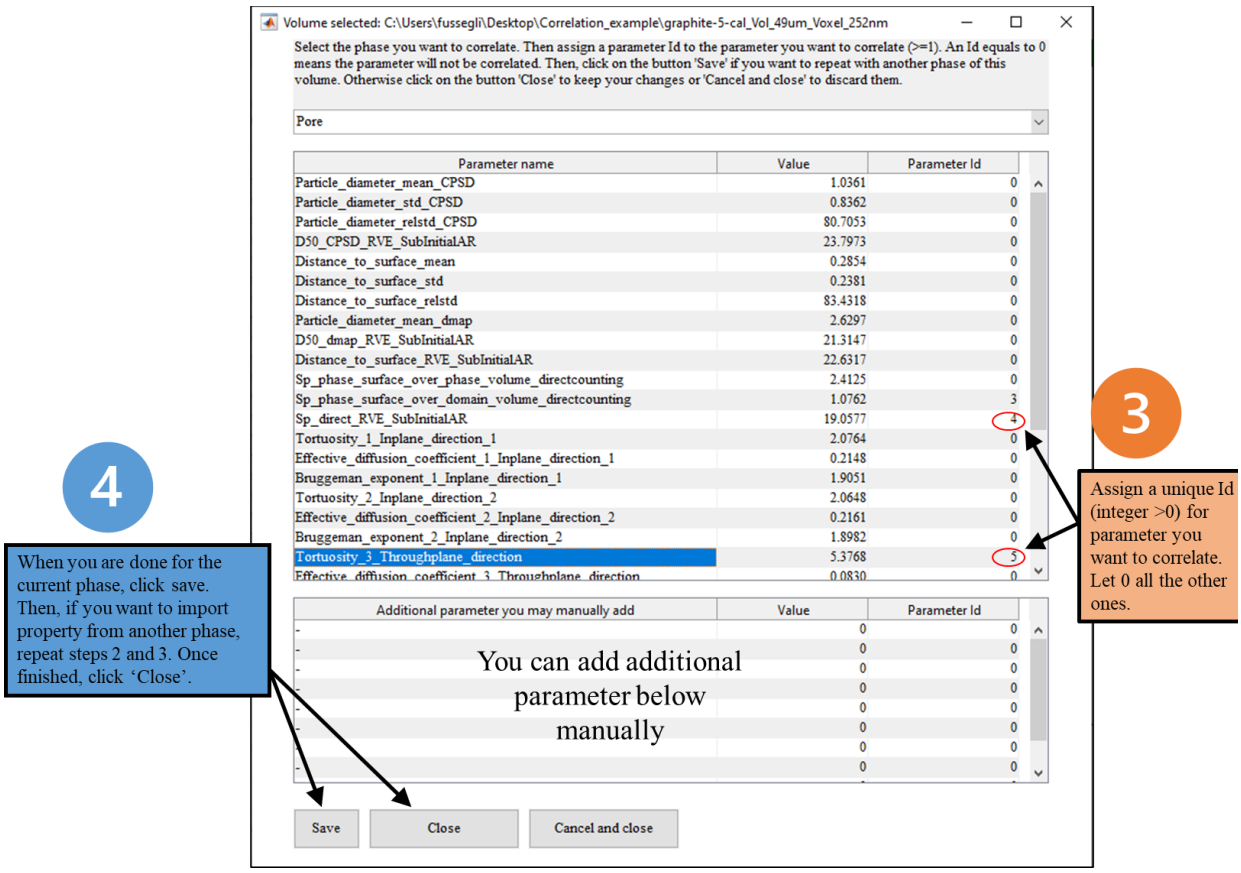


Figure VIII-3. Step 3 and 4: Select properties you want to correlate.

The names reported in the Parameter name are those used in the 'Result_correlation' MATBAL structure, e.g., in Function_Tortuosity_factor_taufactor.m (src\Microstructure_characterization):

```
results_correlation(current_phase).(['Tortuosity_' num2str(current_direction) '_' str_direction])
= Tortuosity_factor(current_phase,current_direction);
% Str_direction is the direction name entered in the microstructure characterization import tab.
```

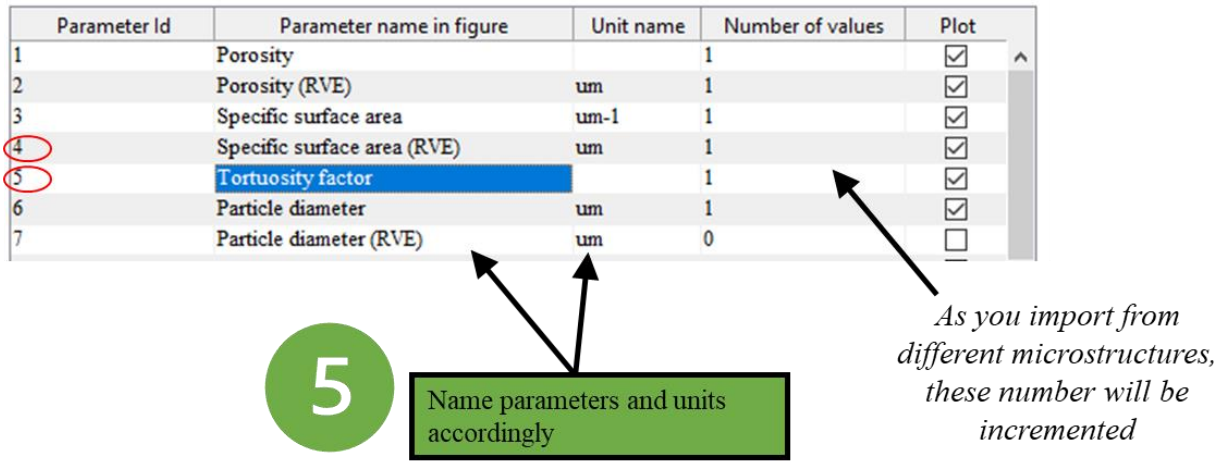
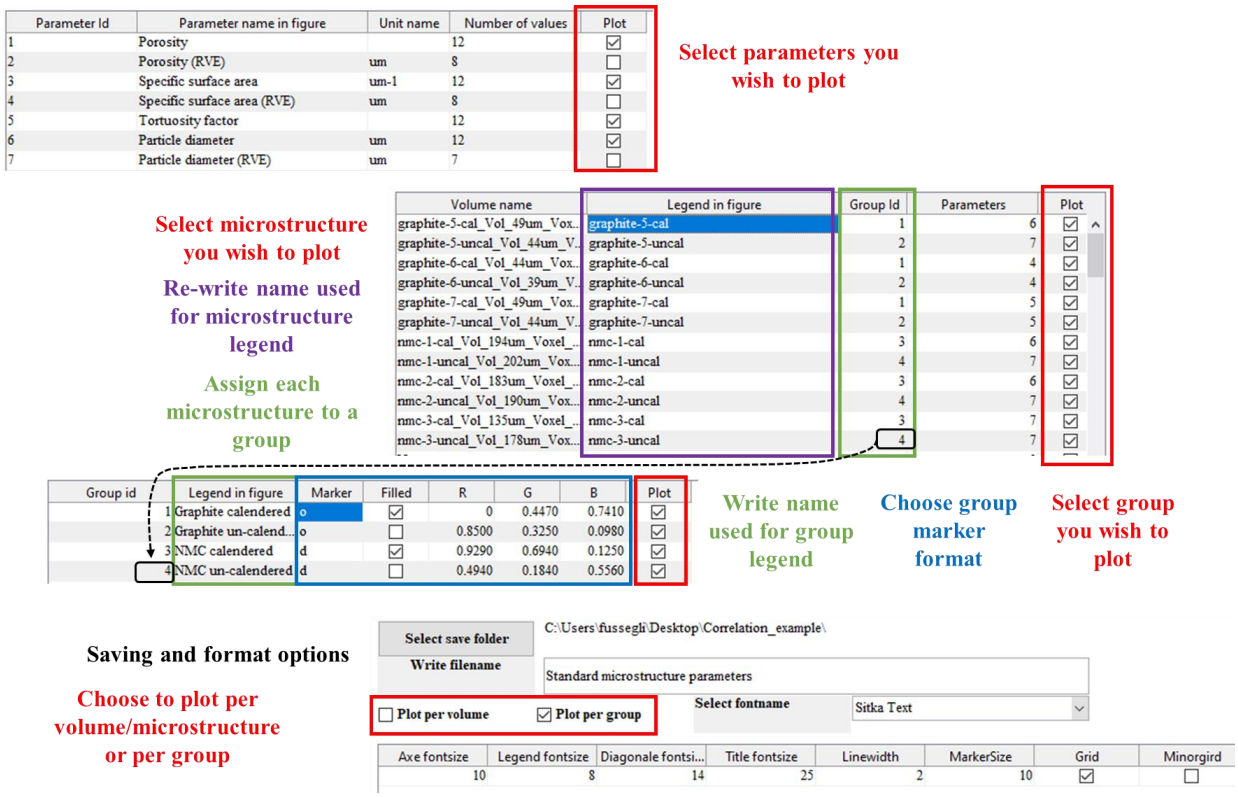


Figure VIII-4. Step 5: Name accordingly parameters (names will be used for figure). You have to do it only one time.

Then repeat steps 1-4 for each microstructure you would like to import. It is not required that each microstructure has all the microstructure properties. In the figure above, for instance, one RVE value was not available for the first volume. Once done, you can choose your plotting options and create the figure with the corresponding button.



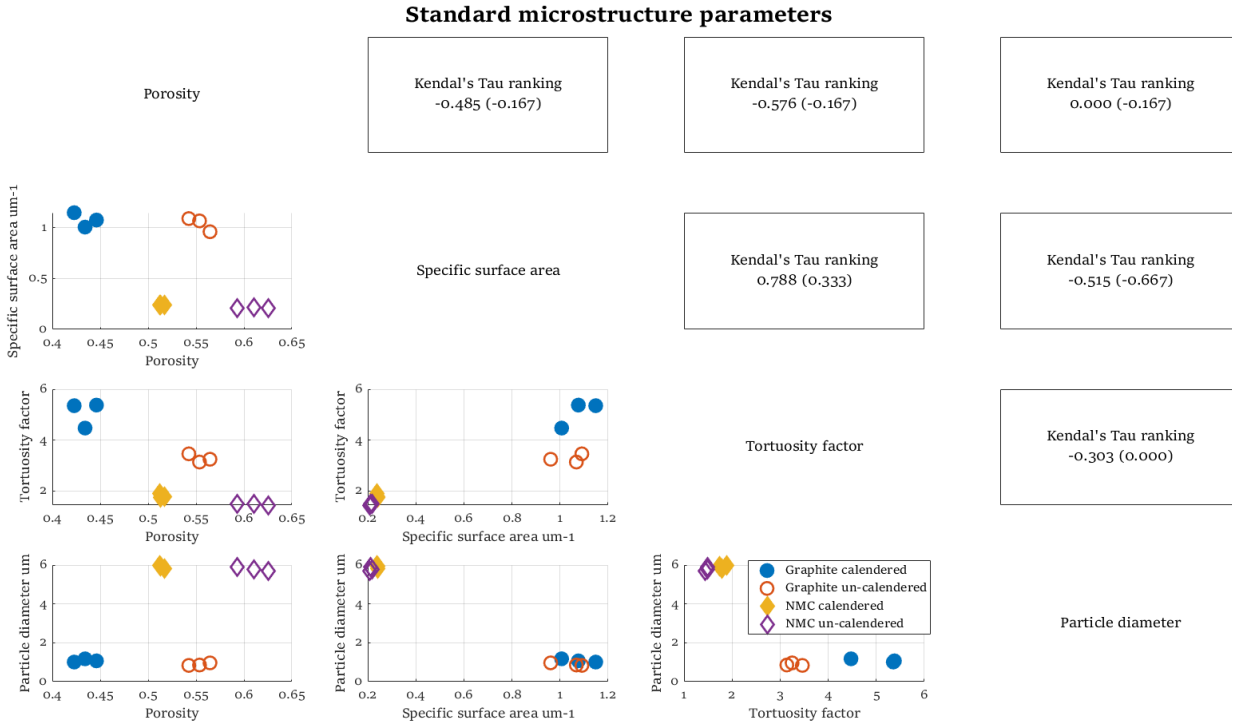


Figure VIII-5. (Top) plotting options, and (bottom) figure created. On the top right the Kendal's tau ranking value is indicated (first value is obtained considering all data points, while value in parenthesis is obtained by first calculating the coefficient within each group and then takes its average).

The Kendal rank correlation coefficient[59] (also called Kendall's tau coefficient) is also calculated for each property doublet. It is a non-parametric measure of the ordinal association, or rank correlation, between two variables. Any pair (x_i, y_i) and (x_j, y_j) are said to be concordant if they share the same rank (i.e., if $x_i > x_j$ and $y_i > y_j$ or if $x_i < x_j$ and $y_i < y_j$). Otherwise, they are said to be discordant. If $x_i = x_j$ or $y_i = y_j$ they are neither concordant nor discordant. The Kendall's tau coefficient τ is then defined as:

$$\tau = \frac{c - d}{\frac{n \times (n-1)}{2}} \quad -1 < \tau < 1$$

$$with \begin{cases} c \text{ the number of concordant pairs} \\ d \text{ the number of discordant pairs} \\ n \text{ the number of pairs} \end{cases} \qquad \text{[VIII-1]}$$

If the two properties share the same ranking (i.e., when *x* is increasing, so does *y*), then $\tau = 1$. Inversely, if their ranking is the reverse of the other (i.e., when x is increasing, y is decreasing), then $\tau = -1$. Both cases indicate a strong correlation between the two properties. If $\tau = 0$, then it means the properties are independent: no relationship is expected to exist between them. If

volumes are sorted per group, the ranking coefficient is also calculated within each group, and then its average value among all groups is calculated. This is relevant for establishing two parameters are correlated when such correlation is microstructure specific, for instance for tortuosity factor and porosity. Kendall's tau coefficient is calculated with Function_Kendall_tau_ranking.m (src/Properties_correlation).

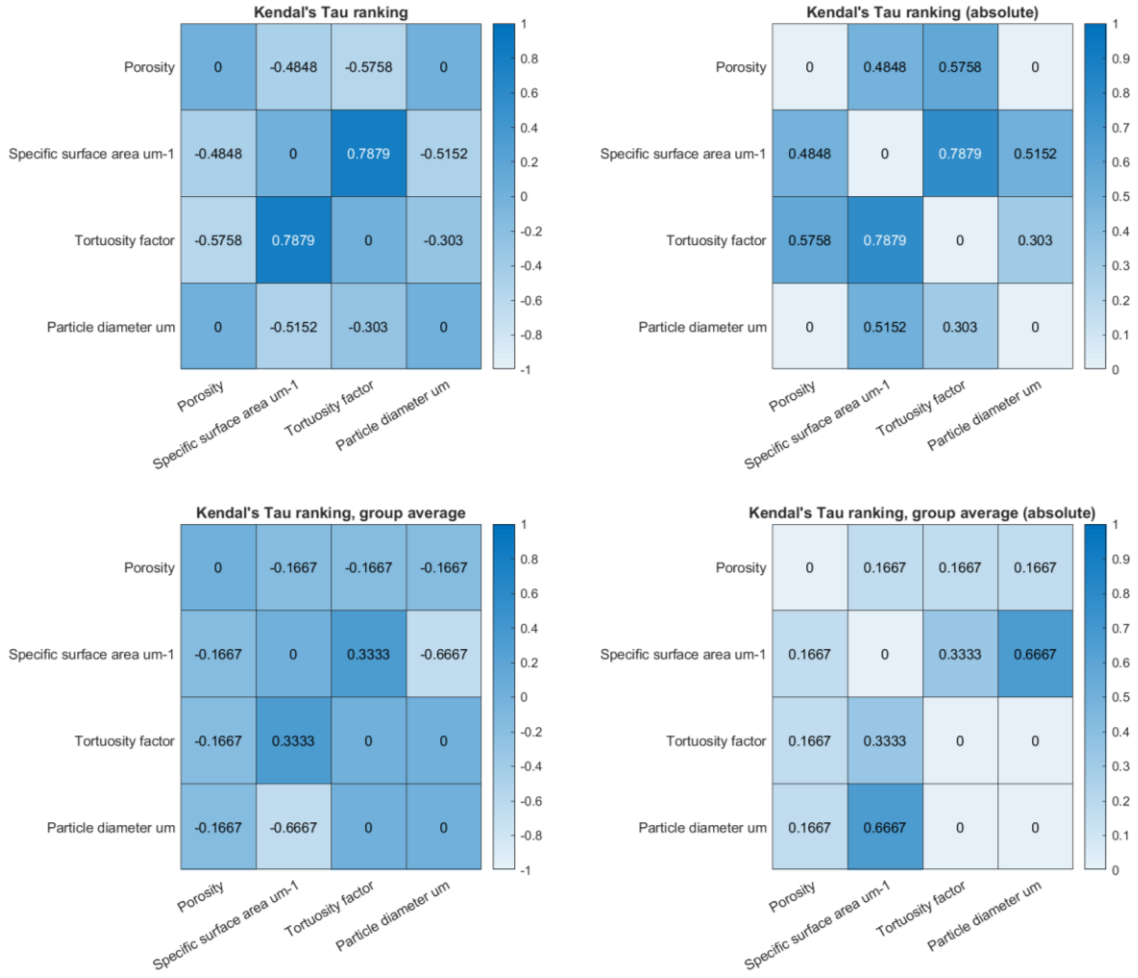


Figure VIII-6. Correlation map using Kendal's tau ranking (more data points are required to provide a relevant correlation analysis).

You can tailor your plotting changing the GUI options, for instance:

Parameter Id	Parameter name in figure	Unit name	Number of values	Plot
1	Porosity		12	☑
2	Porosity (RVE)	um	8	☐
3	Specific surface area	um-1	12	☐
4	Specific surface area (RVE)	um	8	☐
5	Tortuosity factor		12	☑
6	Particle diameter	um	12	☐
7	Particle diameter (RVE)	um	7	☐

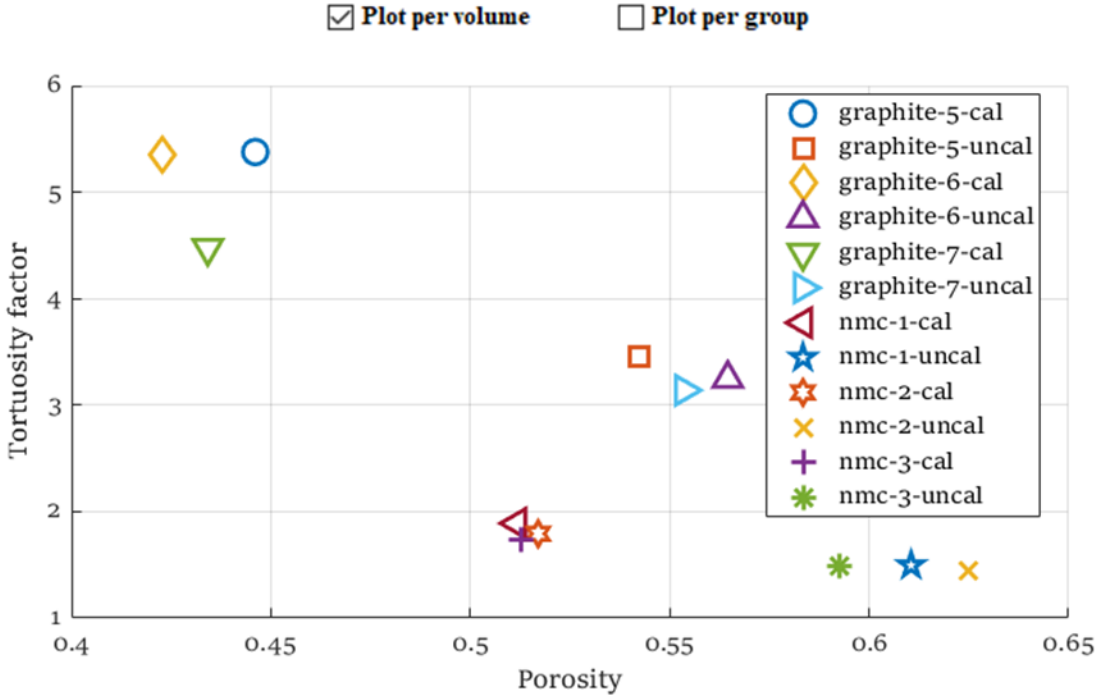


Figure VIII-7. Plotting per volume instead of per group to identify who's who.

You can also visualize in the MATLAB command window or export in an excel file the correlation matrix with the corresponding buttons, valuable to build a database for instance.

	A	B	C	D	E	F	G	H
1	Volume	Porosity	Porosity_RVE	Specific_surface_area	Specific_surface_area_RVE	Tortuosity_factor	Particle_diameter	Particle_diameter_RVE
2	graphite-5-cal	0.446099503	21.56040577	1.076195201	19.05771619	5.376845399	1.069881493	
3	graphite-5-uncal	0.542442657	19.11721511	1.090703782	16.95753699	3.461939333	0.835761682	22.09227643
4	graphite-6-cal	0.422851695		1.148414145		5.353910394	1.008500988	
5	graphite-6-uncal	0.564568331		0.96101055		3.248735576	0.96614524	
6	graphite-7-cal	0.434182556		1.00653066	20.78478866	4.473935702	1.175635718	
7	graphite-7-uncal	0.553636258		1.067807124	21.62872156	3.139503047	0.856610975	
8	nmc-1-cal	0.511727304	45.00266561	0.238204971		1.889649182	5.95969952	44.60387945
9	nmc-1-uncal	0.61054028	49.55930566	0.219394306	100.9922931	1.495714775	5.769604257	43.86354278
10	nmc-2-cal	0.517022313	109.0172238	0.242769811		1.791468629	5.823633193	44.89577271
11	nmc-2-uncal	0.625067781	42.5535276	0.209549987	48.94954797	1.447035839	5.683118822	47.16511795
12	nmc-3-cal	0.512883733	40.03395205	0.241691397	28.16334031	1.737745771	5.933161289	51.37086654
13	nmc-3-uncal	0.592541149	84.39252973	0.21230343	50.93184252	1.495581124	5.884003783	50.96405407
14	Unit		um	um-1	um		um	um

Parameters | Kendall_correlation | Kendall_correlation_group_avg

	A	B	C	D	E	F	G	H
1	Parameters	Porosity	Porosity_RVE	Specific_surface_area	Specific_surface_area_RVE	Tortuosity_factor	Particle_diameter	Particle_diameter_RVE
2	Porosity	0	0.142857143	-0.484848485	0.5	-0.575757576	0	-0.047619048
3	Porosity_RVE	0.142857143	0	-0.428571429	0.866666667	-0.357142857	0.428571429	0.047619048
4	Specific_surface_area	-0.484848485	-0.428571429	0	-0.714285714	0.787878788	-0.515151515	-0.238095238
5	Specific_surface_area_RVE	0.5	0.866666667	-0.714285714	0	-0.642857143	0.571428571	0
6	Tortuosity_factor	-0.575757576	-0.357142857	0.787878788	-0.642857143	0	-0.303030303	-0.428571429
7	Particle_diameter	0	0.428571429	-0.515151515	0.571428571	-0.303030303	0	0.428571429
8	Particle_diameter_RVE	-0.047619048	0.047619048	-0.238095238	0	-0.428571429	0.428571429	0

Parameters | Kendall_correlation | Kendall_correlation_group_avg

	A	B	C	D	E	F	G	H
1	Parameters	Porosity	Porosity_RVE	Specific_surface_area	Specific_surface_area_RVE	Tortuosity_factor	Particle_diameter	Particle_diameter_RVE
2	Porosity	0	-0.533333333	-0.466666667	0.166666667	-0.666666667	-0.6	-0.066666667
3	Porosity_RVE	-0.533333333	0	-0.466666667	0.833333333	0.533333333	0.466666667	-0.333333333
4	Specific_surface_area	-0.466666667	-0.466666667	0	-0.333333333	0.533333333	0.066666667	0.066666667
5	Specific_surface_area_RVE	0.166666667	0.833333333	-0.333333333	0	-0.166666667	0	-0.666666667
6	Tortuosity_factor	-0.666666667	0.533333333	0.533333333	-0.166666667	0	0.533333333	-0.2
7	Particle_diameter	-0.6	0.466666667	0.066666667	0	0.533333333	0	0.2
8	Particle_diameter_RVE	-0.066666667	-0.333333333	0.066666667	-0.666666667	-0.2	0.2	0

Parameters | Kendall_correlation | Kendall_correlation_group_avg

Figure VIII-8. Correlation matrix exported in excel.

IX. CREATE MESH FOR FEM

Disclaimer: the unstructured mesh generation provided by the third-party code package Iso2mesh[3] may fail when generating large meshes and/or volumes with lot of phases. Workaround are discussed in the text. Nevertheless, it is possible to mesh full-cell geometry with dozens of millions of cells, enough for RVE scale electrochemical simulation as shown at the end of this section.

1. Meshing capabilities

The module enables you to create volumetric tetrahedron-based meshes from a single n-phases volume (e.g., meshing an electrode to perform homogenization calculations) or from several n-phase volumes combined together (e.g., meshing a full cell from anode and cathode volumes). Module input are segmented stack tif files. Meshes can be generated for the whole volume (i.e., a unique mesh), for groups of phases (e.g., a first mesh for the union of positive current collector and cathode solid material, a second mesh for the union of separator and electrolyte of both electrodes, and a third mesh for the union of negative current collector and anode solid material), and for each phase, with meshes having conforming interfaces. The above choice depends on your modeling: a monolithic model would require a single mesh for the whole volume, while a segregated model[60] – that solves domain sequentially – would need meshes per phase or per group of phases. Furthermore, two meshing generation approaches are available:

- structured mesh with cuboid representation: (+) simple, fast and robust generation, high mesh quality cells, (-) vertices expensive, no surface smoothing, no mesh density control
- unstructured mesh: (+) mesh density control, smooth surface, (-) time and RAM expensive, variable mesh quality cell, may fail for large volume and/or volume with large number of phases.

The structure mesh is using a simple in-house algorithm while the unstructured mesh requires Iso2mesh[3] to be installed first (cf. §II for instructions). Iso2mesh is using constrained Delaunay tetrahedralization (CGAL) for surface mesh extraction, Laplacian, Laplacian-HC, and Low-pass filters for surface mesh smoothing, and Tetgen for volumetric mesh generation and adaptative mesh resolution.

All meshes illustrated in this section have been obtained with a Windows computer with 10 cores and 32 GB RAM. In case of MATLAB memory error, it is recommended to switch to a Unix-based system OS (note that Iso2mesh has libraries specific for Windows and Unix OS, please choose the ones relevant for your OS). Meshing time for the larger meshes presented here is below half an hour.

2. Utilization

The module is organized in three main tasks: import and pre-processing (all blue tabs), meshing options (orange tabs), and meshing (red tab). The first step consists in selecting your save folder in the "Folder and save options". You can also choose to save parameters and tif files in addition to the meshes to-be-generated for your record. Module utilization is detailed for two cases: a full cell geometry (cf. §IX-2a) and a particle-scale mesh (cf. §IX-2b).

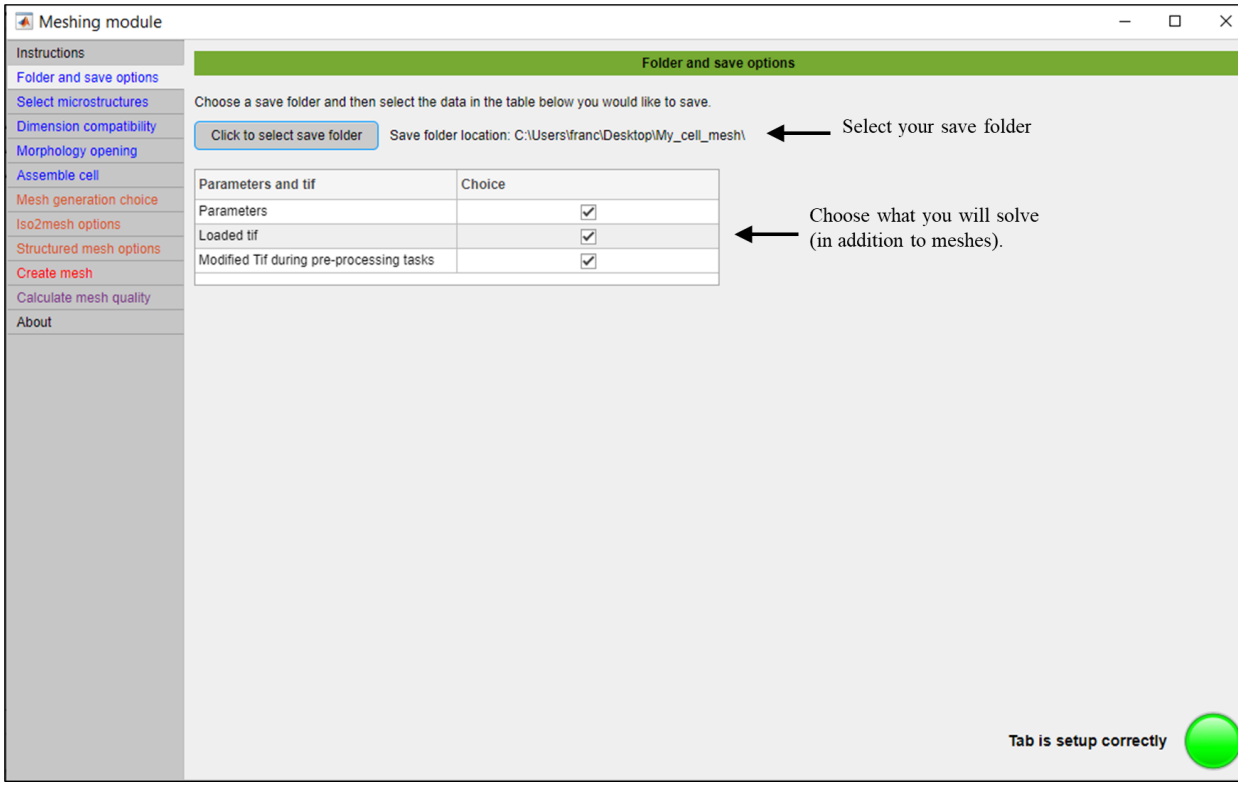


Figure IX-1. Saving options.

a. Full cell mesh generation

In this example, a full cell, 5-volumes (positive current collector, NMC cathode, separator, graphite anode, negative current collector) and 7-phase mesh (positive current collector solid, cathode NMC, cathode electrolyte, separator electrolyte, anode graphite, anode electrolyte, and negative current collector solid), is generated. Electrode volumes have been obtained from X-ray computed tomography, therefore with a complex geometry difficult to mesh, while current collectors and electrolyte are homogenous materials.

i. Import, ROI, and scaling

Three choices are available in the "Select Microstructures" tab: "One unique microstructure", "Half-cell or full-cell" (this example) and "Polycrystalline architecture". The first case only differs from the second by importing a single tif file and skipping the "dimension

compatibility" tab, and is therefore not detailed in the documentation. For each volume you can either import a segmented .tif file (in this example, the anode and graphite heterogenous domains), or generate an homogenous, i.e., one-phase, volume (in this example the current collectors and separator).

The first option, "import .tif" will provide you with Region Of Interest (ROI) and scaling options once file import is successful (cf. Fig. IX-2a). The top section indicates the location of the domains along the first axis (from left to right). Thickness is set, by convention in this module, along the first axis and you can swap between axis if imported volumes have a different orientation. You can also reverse (flip) axis, so that the first slice become the last slice and vice-versa, to make sure left side of the left electrode is in contact with the current collector and not with the separator for instance. More complex ROI selection (such as rotation) can be performed in the filtering/segmentation module if needed. Lastly you can either upscale or downscale the volume (the algorithm used is the same used in §V-5). Upscaling/downscaling is required when imported volumes have not been imaged or numerically generated with the same voxel size. To be coherent, upscaling and/or downscaling must be applied so that each volume share the same voxel size. Downscaling is also interesting to ease the meshing process and reduce CPU and RAM usage. Left table shows volume fractions and is updated if you subsequently modify the Region of Interest (ROI) or perform upscaling or downscaling. At any moment you can click on visualize microstructure to see the current state of the volume.

The second option, "homogenous medium", will only ask the number of voxel of the volume along the cell thickness. Homogenous representation is relevant for modeling current collectors or a microstructure that exhibits features/particle size much smaller compared with the other microstructures considered for the cell. This is typically the case for the separator, for which pores are order of magnitude smaller compared with NMC/graphite electrodes. Meshing materials with order of magnitude difference feature size would add a too high constraint on the mesh.

Once satisfied, click on the 'save electrode' button and repeat for the other materials.

ii. Dimension compatibility

This step is skipped if you have chosen "One unique microstructure" or "Polycrystalline architecture" in the previous tab. Imported volumes may not share the same in-plane dimensions. If not, you can manually crop them or use an auto-crop feature function that will keep the centered part of the volumes. Once dimensions are compatible you can visualize the full cell and save it to move on to the next step. Voxel size is optional, and is only used to provide a physical length.

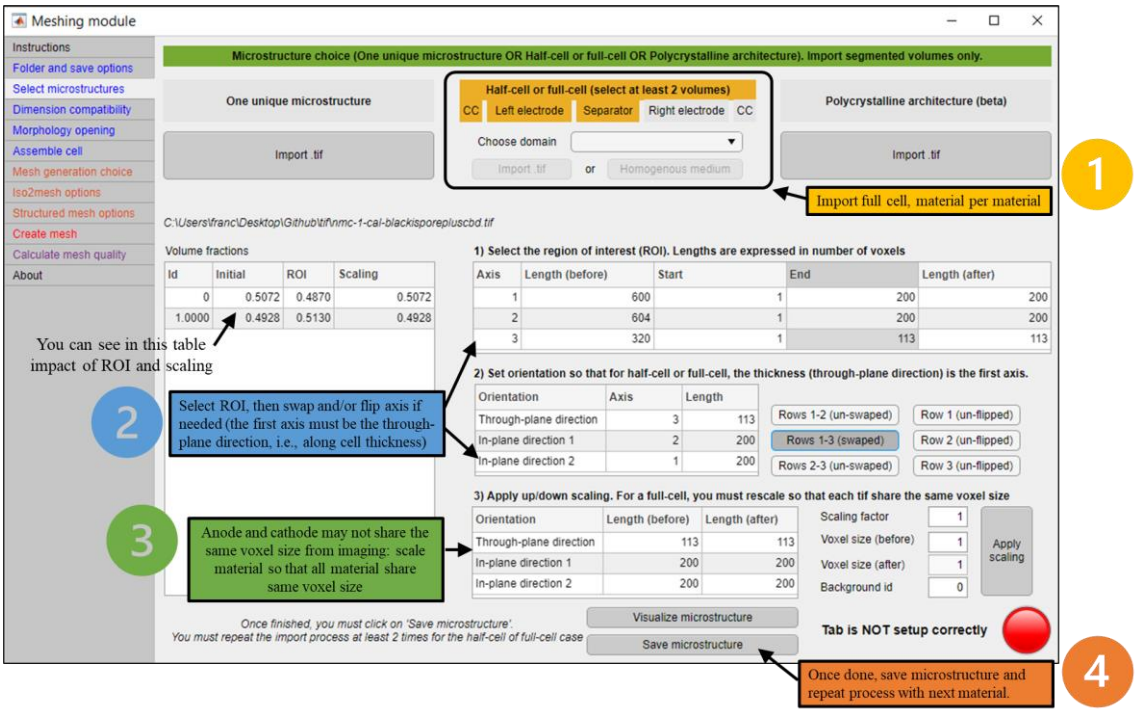


Figure IX-2a. Region of interest and scaling options. Left current collector, left electrode, and separator have been already imported and saved while right electrode is being processed.

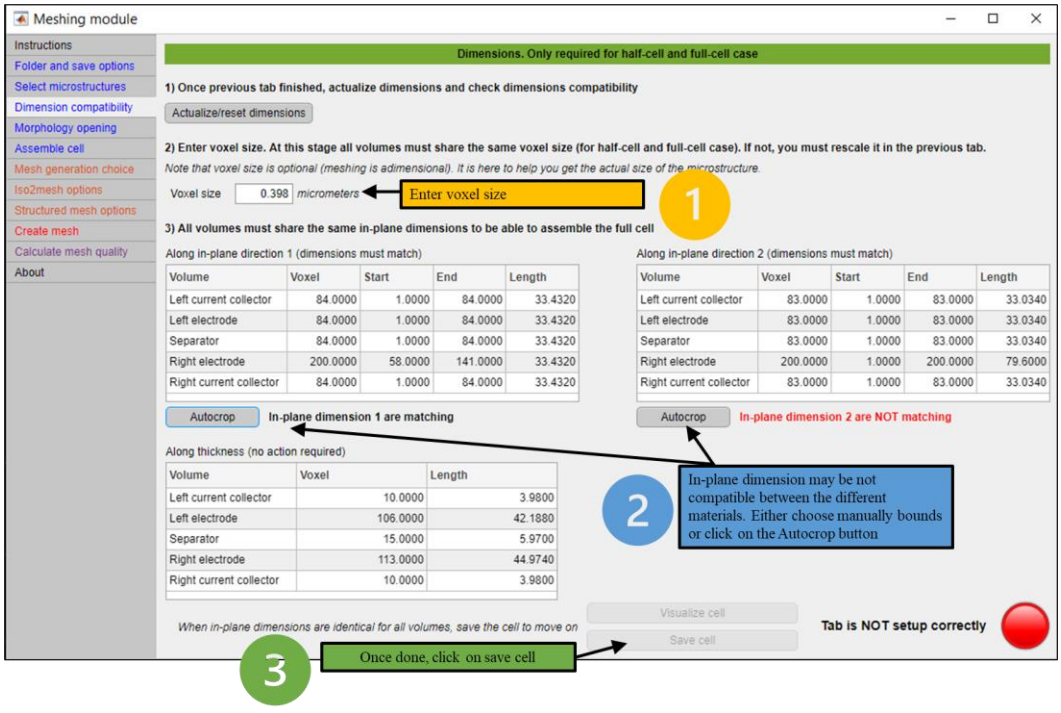


Figure IX-2b. In-plane dimension must be equal to move on. In this example, the second in-plane dimension still need to be corrected.

iii. Morphology opening

The 3D array can be modified to ease the meshing generation. You can choose to apply modifications independently for each volume (for instance, there is no need to apply any morphology openings to homogenous volumes).

- The first step consists in a standard morphology opening: **an erosion step followed by a dilatation step** as illustrated in Fig. IX-2c and mathematically defined in equation IX-1. By default, only one iteration is performed with a unit voxel length for both the erosion and the dilatation distance, but you can modify them at your convenience (cf. Fig. IX-2e). For a volume with a high enough image resolution this step has an unsignificant impact on the volume fraction and the global shape of the particles. Indeed, modifying the one-voxel thick surface layer of a phase is a non-issue as the combined error of experimental imaging and segmentation gives anyway a low confidence on its actual state.

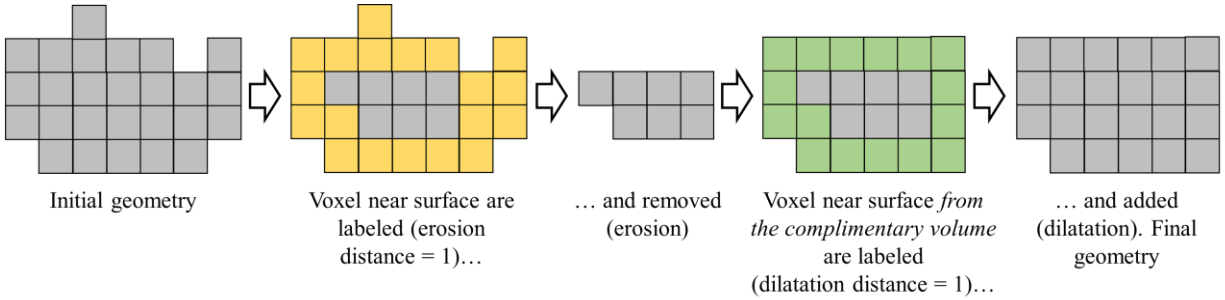


Figure IX-2c. Erosion-dilatation morphology opening illustrated in 2D, using a Chessboard distance to label voxel near surface. Note that the surface roughness/complexity has decreased from the initial geometry (left) to the final geometry (right)

$$\begin{aligned}
&\text{Initialization} && \begin{cases} BW(x) = \begin{cases} 1 \text{ if } x \in \text{solid phase} \\ 0 \text{ otherwise} \end{cases} \end{cases} \\
&\text{Erostion step} && \begin{cases} dmap(x) = min(\|x-y\|)\ \forall x \in \text{solid phase and } \forall y \notin \text{solid phase} \\ BW(dmap \leq erosion\ distance) = 0 \end{cases} \\
&\text{Dilatation step} && \begin{cases} dmap'(x) = min(\|x-y\|)\ \forall x \notin \text{solid phase and } \forall y \in \text{solid phase} \\ BW(dmap' \leq dilatation\ distance) = 1 \end{cases} \\
& && \|x-y\| = \max\left(|x_1-y_1|, |x_2-y_2|, |x_3-y_3|\right)
\end{aligned} \qquad \text{[IX-1]}$$

The algorithm is available in Function_morphologyopening_erosiondilatation_algorithm.m and is pretty simple (the distance method is set by default to "Chessboard"). The algorithm can be applied iteratively, and either on the solid phase union or phase per phase (see Function_morphologyopening_erosiondilatation.m for implementation).

```
function [BW] =
Function_morphologyopening_erosiondilatation_algorithm(BW,erosion_distance,tolerance,dilatation_d
istance,distance_method)
% Morphology opening (erosion and dilation) to simplify binary image surface
% BW = Dilation( Erosion (BW) )
```

```
distance_map = bwdist(~BW,distance_method); % Distance map
BW(distance_map<=erosion_distance+tolerance)=0; % Erosion
distance_map = bwdist(BW,distance_method); % Distance map
BW(distance_map<=dilatation_distance+tolerance)=1; % Dilatation
end
```

- The second step consists in "cleaning" voxel connections and/or converting phases in unique clusters. Voxels that belong to the same phase may have **incorrect voxel-voxel connection** as illustrated in figure IX-2d. Adjacent voxels connected through only a line or a point can lead to a mesh with adjacent cells connected through only an edge or a vertex for which flux density are ill-defined (singular) which can result in numerical instability, in addition to be more challenging (and for large and complex geometries, impossible) to mesh. Therefore, this morphology opening step consists in identifying such line-to-line and point-to-point connections and assign the problematic solid phase voxel to the pore domain, following the set of operations defined in equations IX-2 to 4. The element-wise definition enables parallel implementation of the voxel connectivity analysis. The algorithm is detailed in Function_clean_voxelconnection.m and called by Function_clean_voxelconnection_multiphase.m and applied for the union of all solid phase, or solid phase per solid phase (cf. Fig. IX-2e).

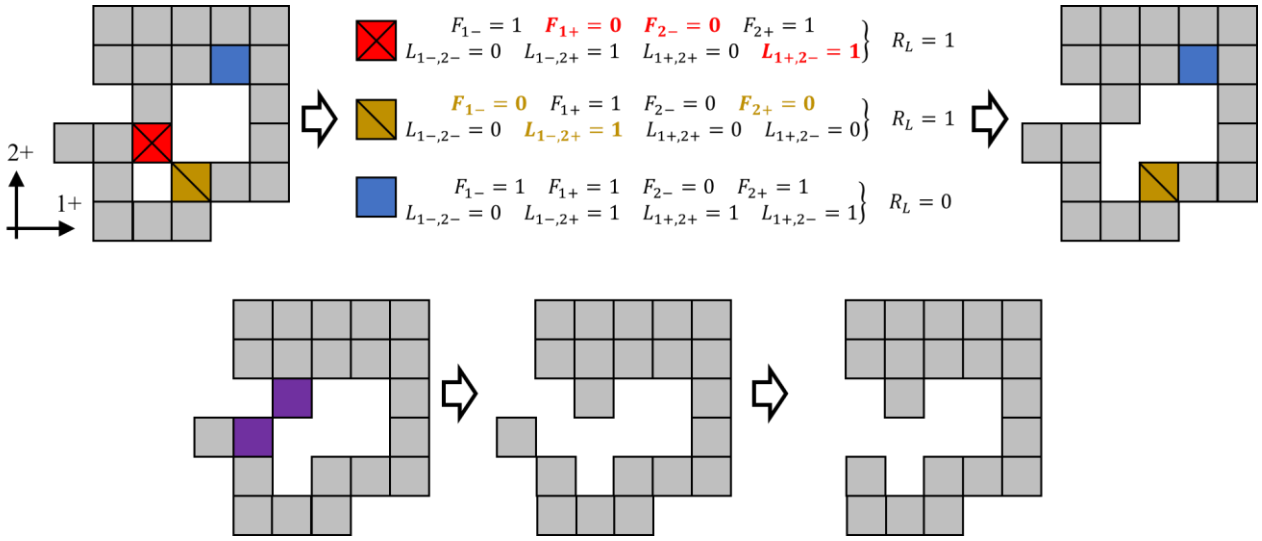


Figure IX-2d: Voxel connectivity correction step illustrated in 2D. (top left) Initial geometry, with three pixels colored for which method calculation is detailed. (top center) For each voxel, the 4 (6 in 3D) face-to-face connections F and 4 (12 in 3D) line-to-line connections L are calculated, respectively with equations IX-3a and IX-3b. In 3D, the point-to-point connections P would also be calculated with equation IX-3c. Both red and brown voxels have line-to-line connections not reinforced by face-to-face connections (bold terms), while all the blue voxel connections are correct. Red and brown voxels and then labelled to be removed ($R_L = 1$), although to fix the voxel connectivity issue only one need to be removed. This is considered in equations IX-2b (and IX-2c in 3D) as only half of the connections are checked due to symmetry; therefore, only red pixel is removed using equation IX-2a. (top right) Geometry after one iteration. (bottom row) The while loop is continued until convergence. (bottom right) Final geometry is stable.

Remove voxel based on connections $BW(x) = BW(x) \circ (\sim R(x))$ $R = R_L + R_P$	[IX-2a]
$R_L = L_{1+,2+} \circ \sim(F_{1+} + F_{2+}) + L_{1+,2-} \circ \sim(F_{1+} + F_{2-}) + L_{1+,3+} \circ \sim(F_{1+} + F_{3+}) + L_{1+,3-} \circ \sim(F_{1+} + F_{3-}) + L_{2+,3+} \circ \sim(F_{2+} + F_{3+}) + L_{2+,3-} \circ \sim(F_{2+} + F_{3-})$	[IX-2b]
$R_P = \begin{matrix} P_{1_+,2_+,3_+} \circ \sim\left(F_{1+} \circ L_{1+,2+} + F_{2+} \circ L_{1+,2+} + F_{1+} \circ L_{1+,3+} + F_{3+} \circ L_{1+,3+} + F_{2+} \circ L_{2+,3+} + F_{3+} \circ L_{2+,3+}\right) \\ + P_{1_+,2_+,3_-} \circ \sim\left(F_{1+} \circ L_{1+,2+} + F_{2+} \circ L_{1+,2+} + F_{1+} \circ L_{1+,3-} + F_{3-} \circ L_{1+,3-} + F_{2+} \circ L_{2+,3-} + F_{3-} \circ L_{2+,3-}\right) \\ + P_{1_+,2_-,3_+} \circ \sim\left(F_{1+} \circ L_{1+,2-} + F_{2-} \circ L_{1+,2-} + F_{1+} \circ L_{1+,3+} + F_{3+} \circ L_{1+,3+} + F_{2-} \circ L_{2-,3+} + F_{3+} \circ L_{2-,3+}\right) \\ + P_{1_-,2_+,3_+} \circ \sim\left(F_{1-} \circ L_{1-,2+} + F_{2+} \circ L_{1-,2+} + F_{1-} \circ L_{1-,3+} + F_{3+} \circ L_{1-,3+} + F_{2+} \circ L_{2+,3+} + F_{3+} \circ L_{2+,3+}\right) \end{matrix}$	[IX-2c]
Face-to-face neighbors' detection (4 in 2D, 6 in 3D) $F_{i_e}(1 + \delta_{1i}\delta_{-e}: n_1 - \delta_{1i}\delta_{+e}, 1 + \delta_{2i}\delta_{-e}: n_2 - \delta_{2i}\delta_{+e}, 1 + \delta_{3i}\delta_{-e}: n_3 - \delta_{3i}\delta_{+e}) = BW(1 + \delta_{1i}\delta_{-e}: n_1 - \delta_{1i}\delta_{+e}, 1 + \delta_{2i}\delta_{-e}: n_2 - \delta_{2i}\delta_{+e}, 1 + \delta_{3i}\delta_{-e}: n_3 - \delta_{3i}\delta_{+e}) \circ BW(1 + \delta_{1i}\delta_{+e}: n_1 - \delta_{1i}\delta_{-e}, 1 + \delta_{2i}\delta_{+e}: n_2 - \delta_{2i}\delta_{-e}, 1 + \delta_{3i}\delta_{+e}: n_3 - \delta_{3i}\delta_{-e})$ e.g.: $F_{2_+}(1{:}n_1, 1{:}n_2 - 1, 1{:}n_3) = BW(1{:}n_1, 1{:}n_2 - 1, 1{:}n_3) \circ BW(1{:}n_1, 2{:}n_2, 1{:}n_3)$	[IX-3a]
Line-to-line neighbors' detection (4 in 2D, 12 in 3D) $L_{i_e,j_f}\begin{pmatrix} 1 + \delta_{1i}\delta_{-e} + \delta_{1j}\delta_{-f}: n_1 - \delta_{1i}\delta_{+e} - \delta_{1j}\delta_{+f}, \\ 1 + \delta_{2i}\delta_{-e} + \delta_{2j}\delta_{-f}: n_2 - \delta_{2i}\delta_{+e} - \delta_{2j}\delta_{+f}, \\ 1 + \delta_{3i}\delta_{-e} + \delta_{3j}\delta_{-f}: n_3 - \delta_{3i}\delta_{+e} - \delta_{3j}\delta_{+f} \end{pmatrix} = BW\begin{pmatrix} 1 + \delta_{1i}\delta_{-e} + \delta_{1j}\delta_{-f}: n_1 - \delta_{1i}\delta_{+e} - \delta_{1j}\delta_{+f}, \\ 1 + \delta_{2i}\delta_{-e} + \delta_{2j}\delta_{-f}: n_2 - \delta_{2i}\delta_{+e} - \delta_{2j}\delta_{+f}, \\ 1 + \delta_{3i}\delta_{-e} + \delta_{3j}\delta_{-f}: n_3 - \delta_{3i}\delta_{+e} - \delta_{3j}\delta_{+f} \end{pmatrix} \circ BW\begin{pmatrix} 1 + \delta_{1i}\delta_{+e} + \delta_{1j}\delta_{+f}: n_1 - \delta_{1i}\delta_{-e} - \delta_{1j}\delta_{-f}, \\ 1 + \delta_{2i}\delta_{+e} + \delta_{2j}\delta_{+f}: n_2 - \delta_{2i}\delta_{-e} - \delta_{2j}\delta_{-f}, \\ 1 + \delta_{3i}\delta_{+e} + \delta_{3j}\delta_{+f}: n_3 - \delta_{3i}\delta_{-e} - \delta_{3j}\delta_{-f} \end{pmatrix}$ e.g.: $L_{1_-,2_+}(2{:}n_1, 1{:}n_2 - 1, 1{:}n_3) = BW(2{:}n_1, 1{:}n_2 - 1, 1{:}n_3) \circ BW(1{:}n_1 - 1, 2{:}n_2, 1{:}n_3)$	[IX-3b]
Point-to-point neighbors' detection (8 in 3D) $P_{i_e,j_f,k_g}\begin{pmatrix} 1 + \delta_{1i}\delta_{-e} + \delta_{1j}\delta_{-f} + \delta_{1k}\delta_{-g}: n_1 - \delta_{1i}\delta_{+e} - \delta_{1j}\delta_{+f} - \delta_{1k}\delta_{+g}, \\ 1 + \delta_{2i}\delta_{-e} + \delta_{2j}\delta_{-f} + \delta_{2k}\delta_{-g}: n_2 - \delta_{2i}\delta_{+e} - \delta_{2j}\delta_{+f} - \delta_{2k}\delta_{+g}, \\ 1 + \delta_{3i}\delta_{-e} + \delta_{3j}\delta_{-f} + \delta_{3k}\delta_{-g}: n_3 - \delta_{3i}\delta_{+e} - \delta_{3j}\delta_{+f} - \delta_{3k}\delta_{+g} \end{pmatrix} = BW\begin{pmatrix} 1 + \delta_{1i}\delta_{-e} + \delta_{1j}\delta_{-f} + \delta_{1k}\delta_{-g}: n_1 - \delta_{1i}\delta_{+e} - \delta_{1j}\delta_{+f} - \delta_{1k}\delta_{+g}, \\ 1 + \delta_{2i}\delta_{-e} + \delta_{2j}\delta_{-f} + \delta_{2k}\delta_{-g}: n_2 - \delta_{2i}\delta_{+e} - \delta_{2j}\delta_{+f} - \delta_{2k}\delta_{+g}, \\ 1 + \delta_{3i}\delta_{-e} + \delta_{3j}\delta_{-f} + \delta_{3k}\delta_{-g}: n_3 - \delta_{3i}\delta_{+e} - \delta_{3j}\delta_{+f} - \delta_{3k}\delta_{+g} \end{pmatrix} \circ BW\begin{pmatrix} 1 + \delta_{1i}\delta_{+e} + \delta_{1j}\delta_{+f} + \delta_{1k}\delta_{+g}: n_1 - \delta_{1i}\delta_{-e} - \delta_{1j}\delta_{-f} - \delta_{1k}\delta_{-g}, \\ 1 + \delta_{2i}\delta_{+e} + \delta_{2j}\delta_{+f} + \delta_{2k}\delta_{+g}: n_2 - \delta_{2i}\delta_{-e} - \delta_{2j}\delta_{-f} - \delta_{2k}\delta_{-g}, \\ 1 + \delta_{3i}\delta_{+e} + \delta_{3j}\delta_{+f} + \delta_{3k}\delta_{+g}: n_3 - \delta_{3i}\delta_{-e} - \delta_{3j}\delta_{-f} - \delta_{3k}\delta_{-g} \end{pmatrix}$ e.g.: $P_{1_-,2_+,3_+}(2{:}n_1, 1{:}n_2 - 1, 1{:}n_3 - 1) = BW(2{:}n_1, 1{:}n_2 - 1, 1{:}n_3 - 1) \circ BW(1{:}n_1 - 1, 2{:}n_2, 2{:}n_3)$	[IX-3c]
With $\sim A(x) = \begin{cases} 1 \text{ if } A(x) = 0 \\ 0 \text{ otherwise} \end{cases}$, $\delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}$ $i,j \in [1,2,3]$, $\delta_{ef} = \begin{cases} 1 \text{ if } e = f \\ 0 \text{ otherwise} \end{cases}$ $e,f \in [-,+]$, and $A \circ B$ element-wise tensor multiplication. All tensors share same dimensions $[n_1, n_3, n_3]$. Indices 1,2,3 refers to axis orthogonal directions, indices +,- refer to axis orientation, x refers to the voxel location. $A(a{:}b, c{:}d, e{:}f)$ refers to the subvolume cropped from indices a to b for axis 1, c to d for axis 2, and e to f for axis 3.	[IX-4]

- Some models may require phases to be connected, thus the third steps consist in **converting phase in unique cluster**. The user can choose to apply this step on either or both the union of all solid phase and the pore domain (by default the label 0), cf. Fig. IX-2e. Isolated clusters (here defined as the clusters that do not touch the largest cluster) are assigned to the surrounding phase. For instance, applied to the pore, a closed porosity cluster will be assigned to the solid phase that has the most contact with it. As a result, as-modified domains (pore or union of solids) are a 100% connected cluster. This geometry simplification impacts very few Lithium-ion battery electrodes computed tomography images, as phase connection is typically >99% while it helps simplifying subsequent electrochemical modeling. The algorithm is detailed in Function_convert_to_unique_cluster.m and the assignment of isolated clusters to surrounding phase is detailed in Function_assign_voxels_based_on_contact.m

Since modifying voxel connectivity can degrade the phase connectivity, these two last steps (voxel connectivity is performed first then phase connectivity) are realized in a while loop, until convergence if both are selected by the user (see Function_correct_microstructure.m for details). For LIB electrode microstructure volumes with high-enough image resolution, the volume fractions are modified by less than 1%. However, if the volume is too coarsely represented, these steps will significantly degrade the microstructure topology, indicating the microstructure should be upscaled (or re-imaged) before meshing.

The second step (voxel-voxel connectivity) is highly recommended to provide high quality mesh. Unstructured mesh generation may fail without this step.

If the image resolution is too coarse, the volume fraction listed in the right table (cf. Fig. IX-2e) may change significantly, indicating the microstructure is degraded. For a fine enough image resolution, the change should be insignificant.

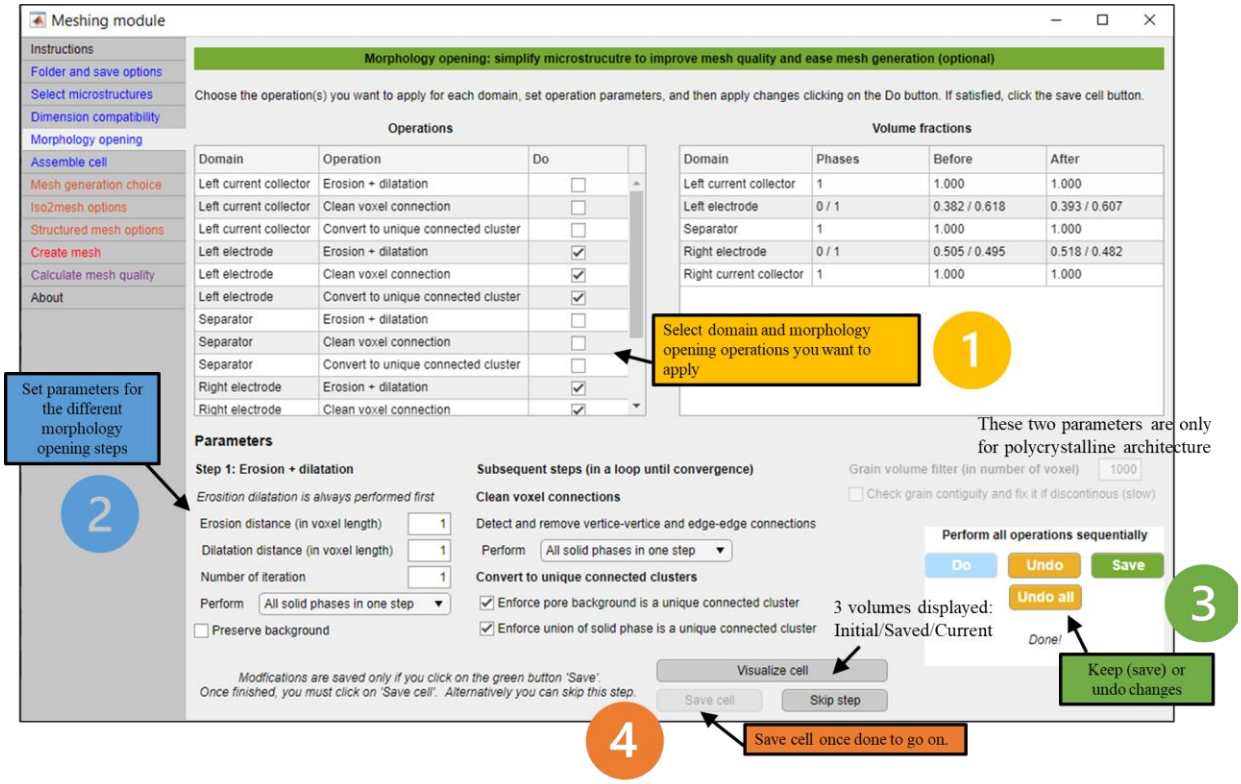


Figure IX-2e. Morphology opening options. Note that you can skip this step if do not wish to apply any modifications.

iv. Assemble cell

Up to this stage, the imported volumes were using their initial label (i.e., label 1 could both identify the anode solid and the cathode solid). The next step consists in labelling each phase at the cell level. In addition, phase can be grouped (for instance, a group that it the union of all electrolyte domains) which is relevant if meshes are planned to be used for a segregated model[60]. You can visualize your phase and group assignment by clicking on the "visualize cell" button (cf. Fig. IX-2e). If you have checked "modified tif during pre-processing tasks" in the save options tab, the group and phase assignment are saved.

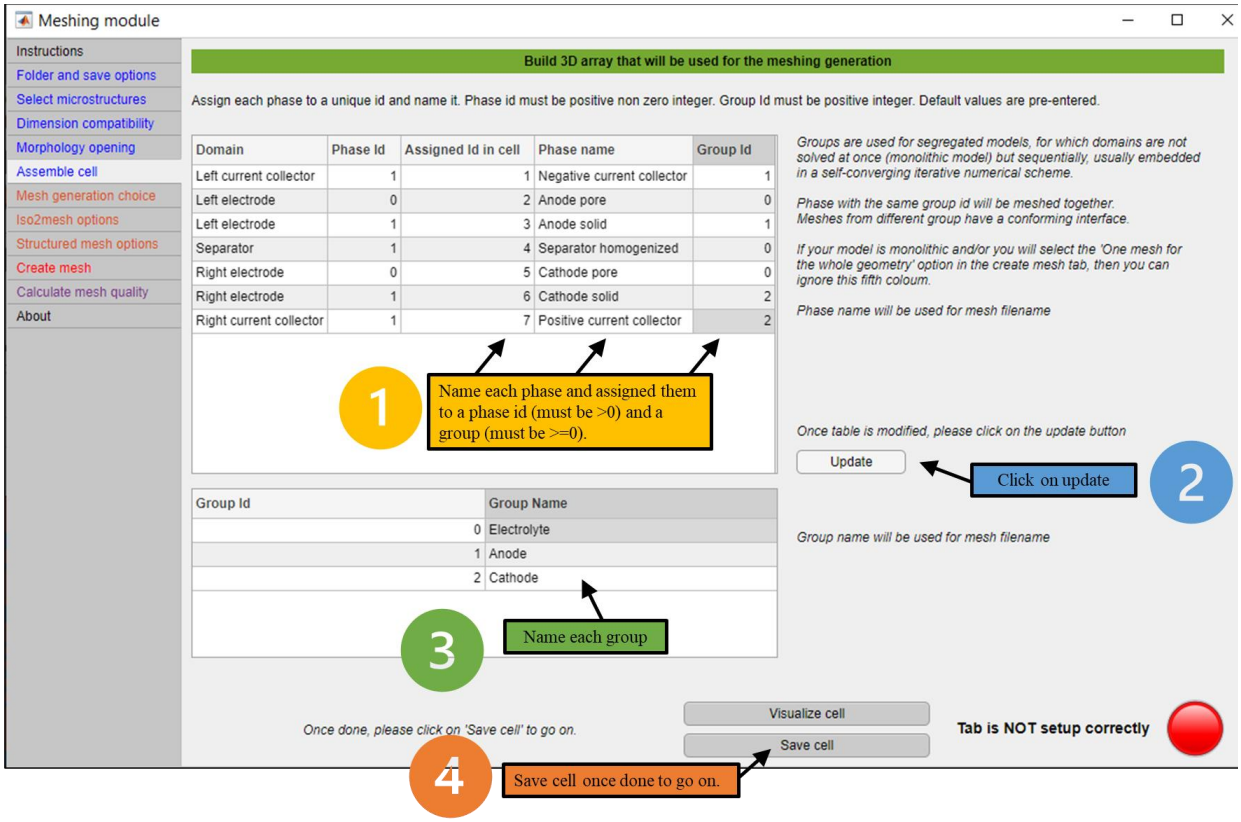


Figure IX-2f. Assemble options. Note that the phase id must be integer >0. Names will be used for mesh filenames.

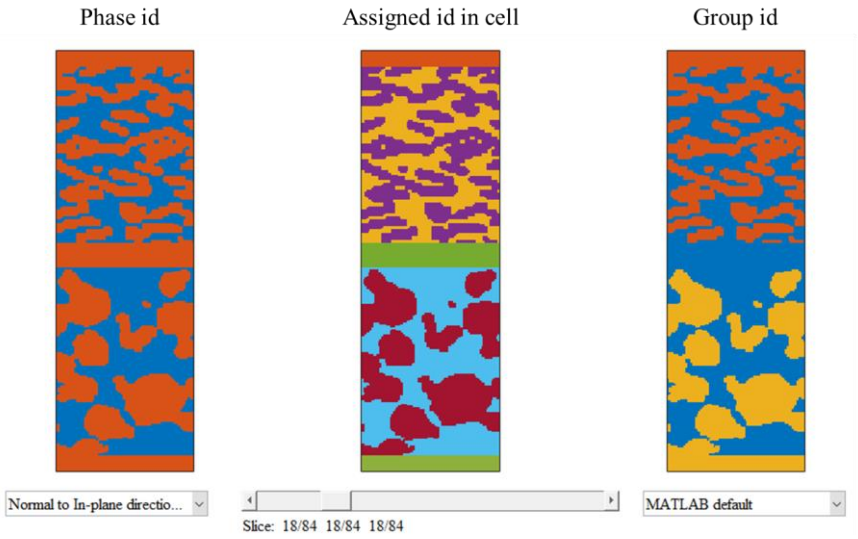


Figure IX-2g. (Left) cell with imported labels, (center) cell with phase id after assemble, and (right) cell with group id after assemble, with 3 groups: {anode current collector, anode solid}, {anode electrolyte, separator, cathode electrolyte}, and {cathode solid, cathode current collector}.

v. Select mesh type and mesh-related options

The next tab will let you select between an unstructured mesh and a structured mesh (as explained in §IX-1).

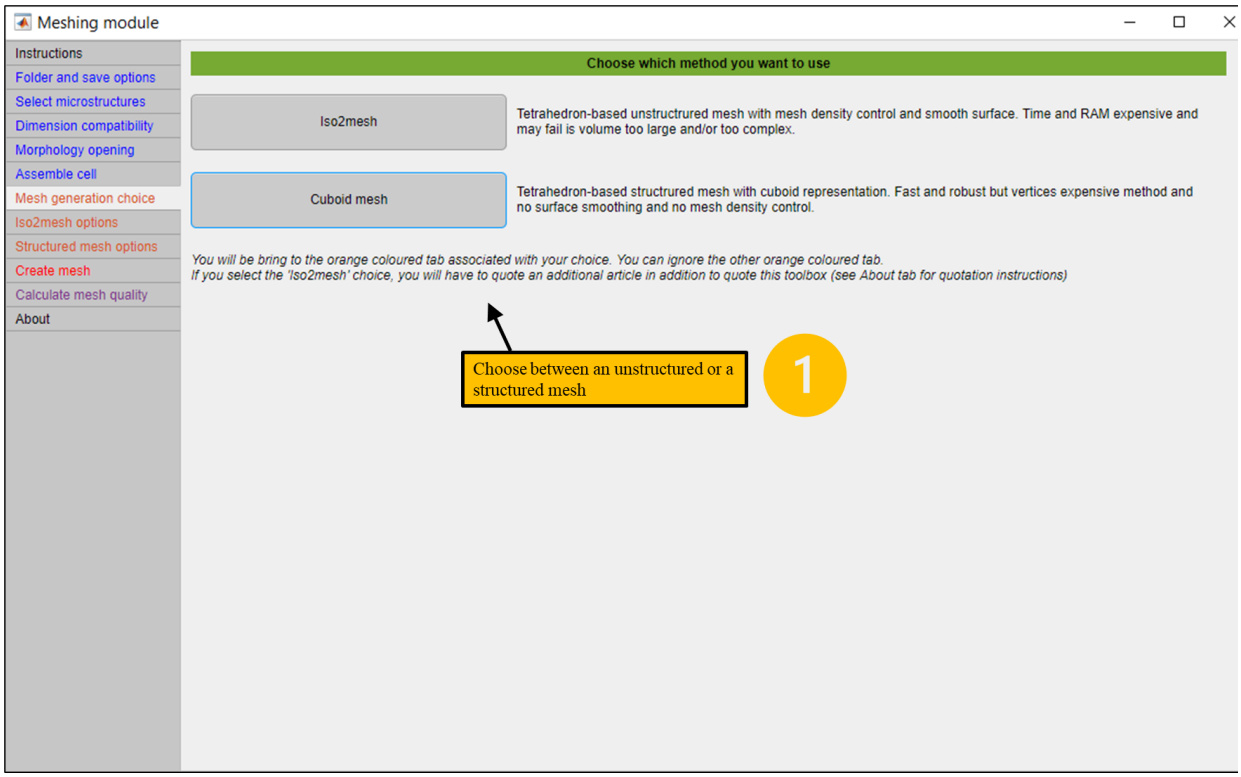


Figure IX-2h. Choose between structured and unstructured mesh generation.

"Iso2mesh" choice will bring you to the tab dedicated to Iso2mesh parameters (cf. Fig. IX-2i), while "Cuboid mesh" will bring you the other tab to simply choose how to discretize a cube with tetrahedrons (cf. Fig. IX-2k).

Iso2mesh options are detailed in the Iso2mesh documentation (see http://iso2mesh.sourceforge.net/cgi-bin/index.cgi?Doc/FunctionList).

- Radbound can be lowered to refine the surface geometry, useful when the microstructure surface is very complex and mesh generation fails. Be careful as reducing it will strongly increase the number of vertices. It roughly indicates the tetrahedron edge length at the phase surface, with unit being the initial voxel length.
- Smoothing can be performed with three different algorithms. Generally, "lowpass" provides the mesh with the higher mesh quality and better volume conservation, while "Laplacian" provides a nice-looking mesh (very smooth) but with smaller mesh quality and an overall slight shrinking of the phase volume. If mesh generation fails, you can reduce the number of iteration as well as the smoothing parameter "useralpha".

- Mesh density is controlled with the "maxvol" parameter. Maxvol sets the maximum tetrahedron volume, normalized with the smallest tetrahedron volume. A unit value indicates all tetrahedron will share roughly the same size (i.e., no mesh density control), while a high value will allow cells far the surface, in the bulk, to be significantly larger than the fine cells located at the phase surface. Note that Iso2mesh will try to reach this volume ratio, but there is no guarantee it will reach it. If mesh generation fails, you can reduce the this parameter.

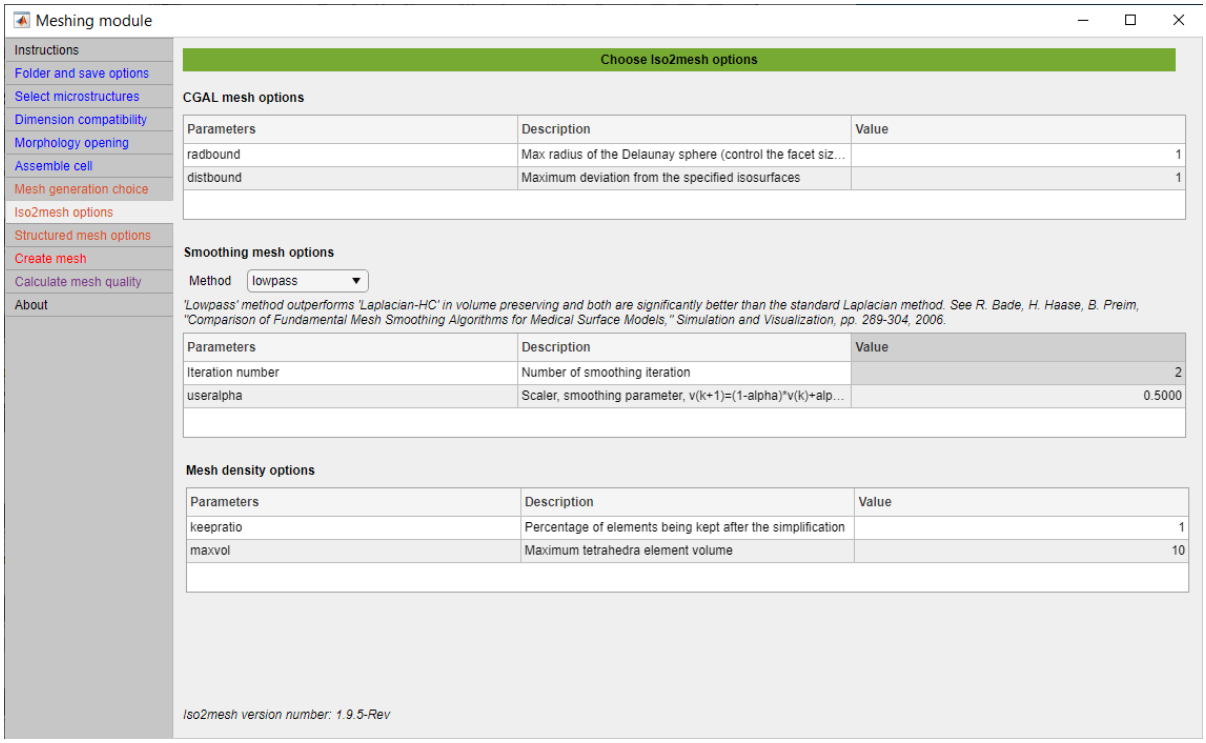


Figure IX-2i. Iso2mesh default options.

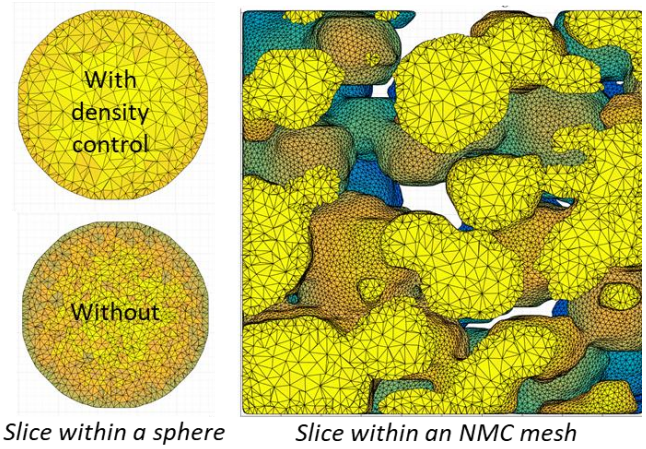


Figure IX-2j. Example of (left) mesh density control and (right) smooth interface obtained with Iso2mesh.

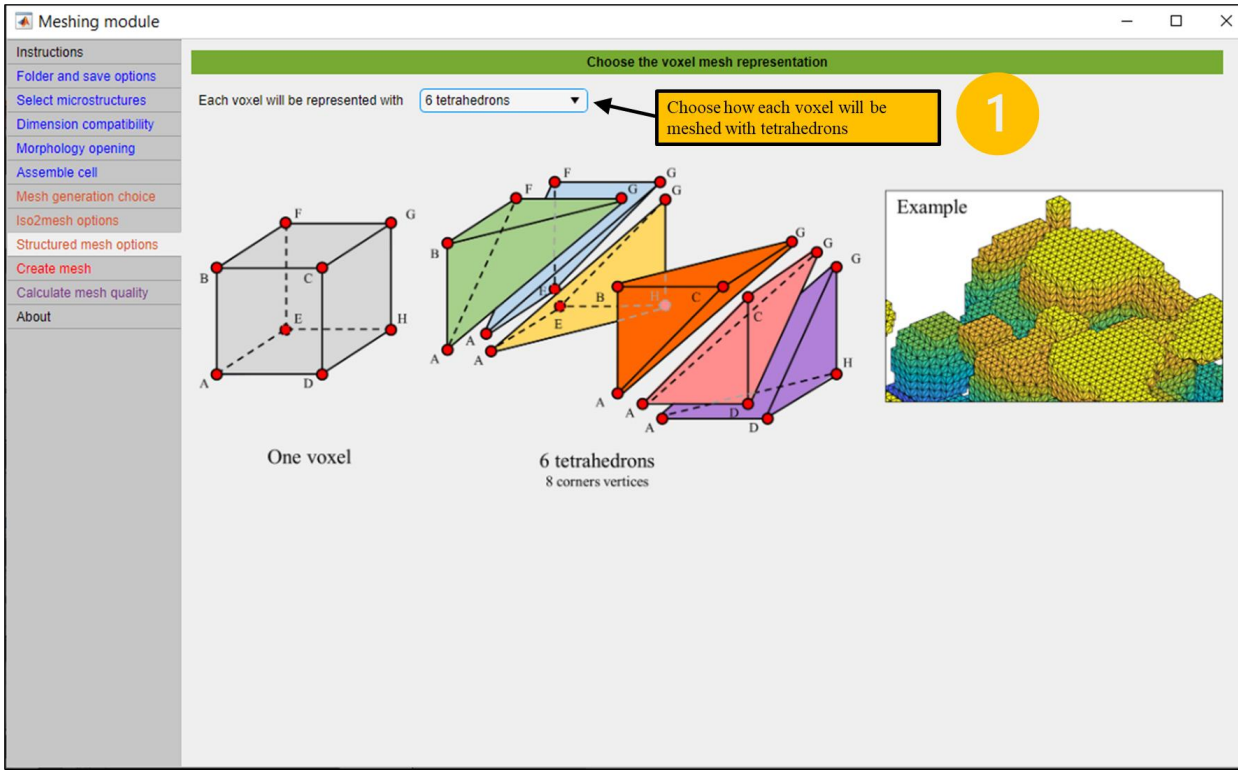


Figure IX-2k. Cuboid mesh option.

vi. Create, visualize and save mesh

You are now ready to generate the mesh. First select if you want to create a mesh per phase, and/or a mesh per group, and/or a mesh for the whole geometry. The two first choices are relevant for segregated models, or homogenization calculations, while the last one is relevant for monolithic model. If you have selected a structure mesh, you can choose to create or not the facet array – which is only require for the MATLAB visualization. Then click on the "create mesh" button. The mesh generation task is time, CPU, and RAM expensive, so you may want to test first the module and its options on a relatively small domain first. The function call to create the meshes are:

- function_iso2mesh_from_array.m for the structure mesh. You can open/edit the file to understand how the parameters (cf. Fig. *IX-2i)* are used. The meshing procedure is simply: surface mesh (*v2s*), smoothing (*smoothsurf*), and volumetric mesh (*surf2mesh*). Iso2mesh has a tendency to ill-assigned tetrahedron cells to a label 0 for large volume. In Iso2mesh terminology, an additional region labeled 0 is, *sometimes*, created in addition to the other relevant regions that correspond each to a phase. You will find a section "MESH CORRECTION" at the end of the file that fix this issue. The "renumerotation" flag is only used for volumes with more than 255 phases (i.e., for polycrystalline architecture). **If you want to add custom modifications/corrections to the mesh or change the Iso2mesh commands, you only have to edit function_iso2mesh_from_array.m.**

- function_regularmesh_from_array.m and function_create_vertices_cell_from_array.m are used to generate cuboid meshes. The approach simply consists in discretizing each cuboid voxel in tetrahedrons.

You can visualize mesh and save it once meshing generation is over. Some examples of visualization are provided in §IX-4. You can save mesh in a variety of functions. GMSH, ABAQUS, STL, BINARY STL files are created using Iso2mesh functions, while .MAT and .CSV files are saving vertices coordinates, cell connectivity, and phase label in different files that can be imported to re-create the mesh as explained in the next section §IX-3.

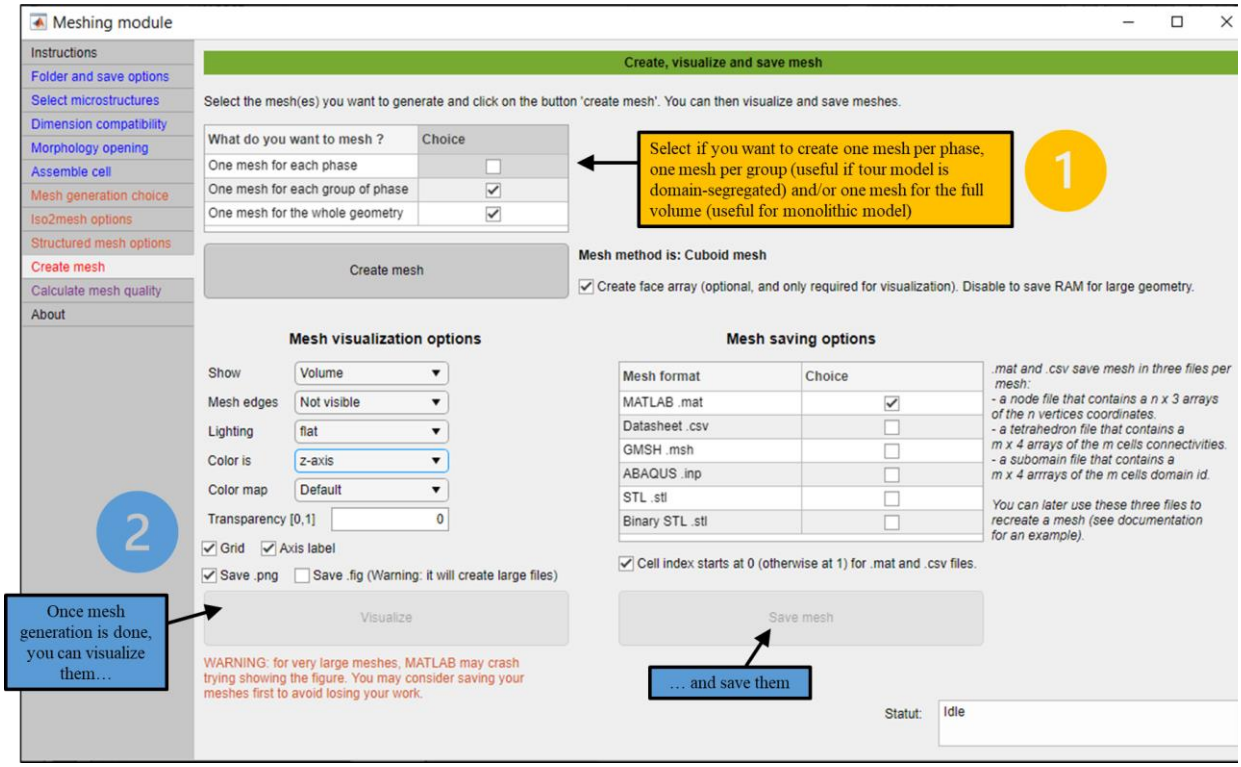


Figure IX-2l. Create, visualize and save mesh.

vii. Calculate mesh quality

Various metrics of mesh quality exist in the literature. The mean ratio η is calculated according to the definition provided by A. Liu and B. Joe [61] (cf. eq. IX-5), with *v* the cell volume and l_{ij} the cell edge length. The mean ratio ranges from 0 (needle-like cell) to 1 (regular tetrahedron) and is calculated with meshquality.m from Iso2mesh.

$$\eta = \frac{12(3v)^{2/3}}{\sum_{0\leq i<j\leq 3} l_{ij}^2} \qquad \text{[IX-5]}$$

b. Particle scale mesh

Lithium-ion battery cathode materials NMC are made of submicron primary particles (called grains) that agglomerate together to form micrometer size secondary particles, with the grain architecture strongly impacting the mechanical integrity of the particle during (de)lithiation[62–64]. NMC grain architecture (both can be imaged, quantified, and representative digital twin can be numerically generated for modeling[65,66]. Polycrhistalline architecture (cf. Fig. IX-2a, right choice) is a special case for which we assume two groups: the background (label 0) and the grains (all other labels). An additional morphology opening step, a grain size filter, is available for this case. Grains with volume (expressed in number of voxels) below the user-defined threshold will be assigned to adjacent grains. This is particularly valuable to remove small grains that may be numerical artifacts.

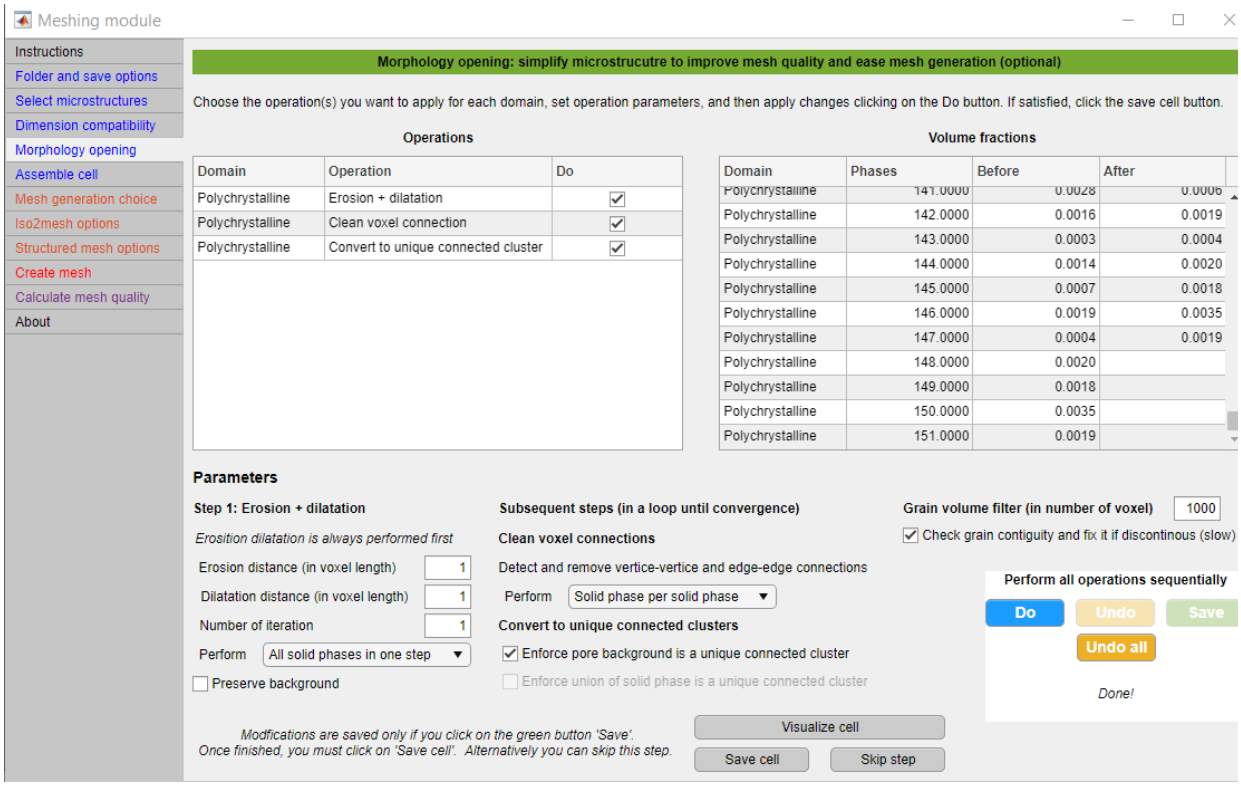


Figure IX-2m. Morphology opening applied to a polycrhistalline architecture. Note that the grain size filter led to a reduction of grain number.

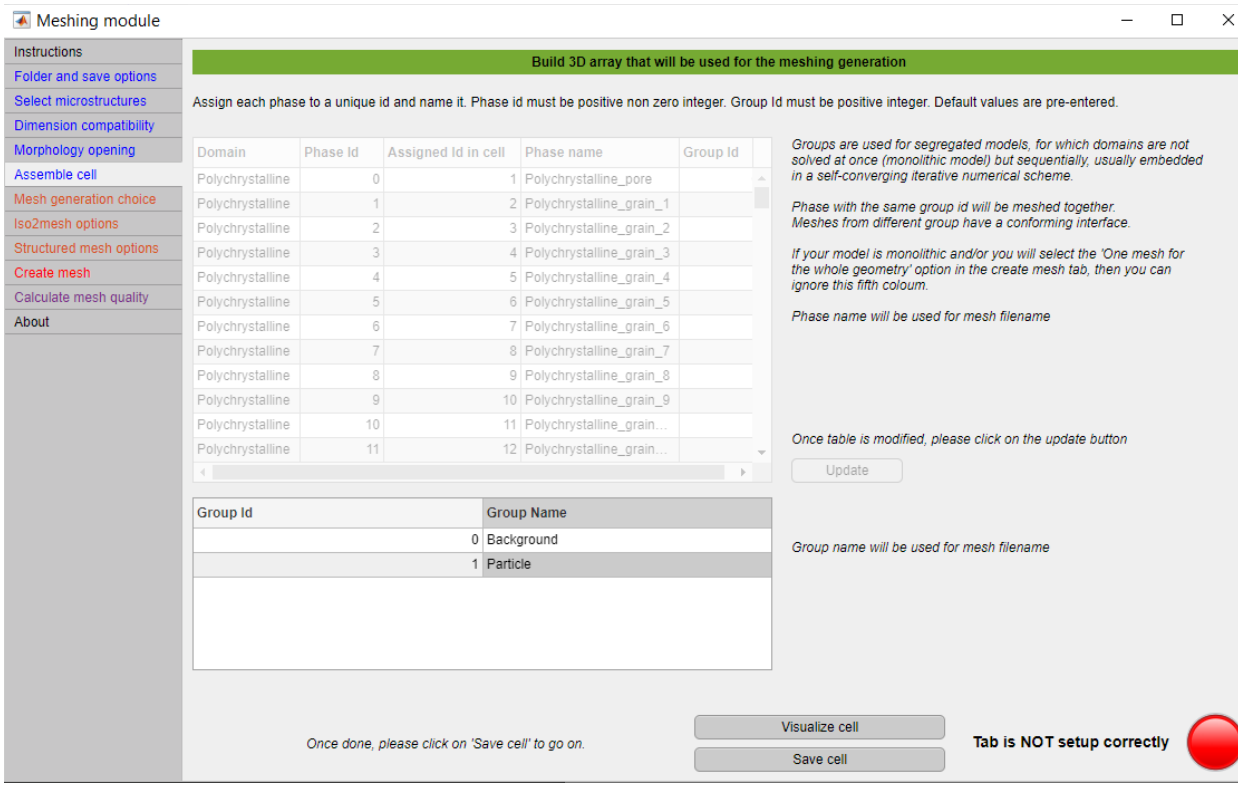


Figure IX-2n. Assigning phase id and grain id is not controllable for a polycrhistalline architecture.

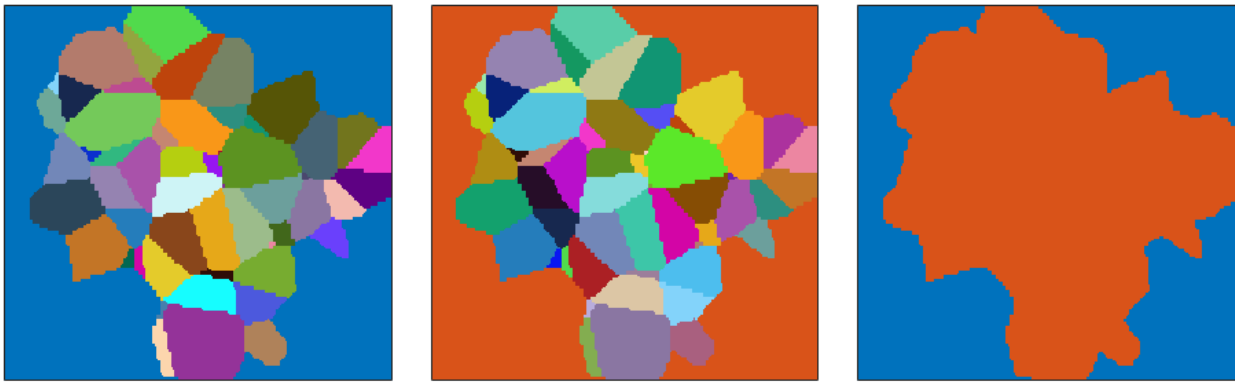


Figure IX-2o. (Left) Imported labels, (center) cell with grain id after assemble, and (right) particle with group id after assemble, with 2 groups: {background}, {union of all grains}.

3. Re-create the mesh in a dedicated FEM software

a. Vertices coordinates, cell connectivity and cell/phase label

In addition to being able to save the meshes in msh, inp, and stl format, vertices coordinates, cell connectivity, and phase label can be saved in .mat and/or .csv files, for each phase, group of phases and/or the whole volume (cf. Fig. IX-2l). Mesh information is organized in three files: "Nodes_<domain_name>", "Tetrahedron_<domain_name>", and "Subdomain_<domain_name>", with <domain_name> being the phase name, group name (as the user choose to name them, cf. Fig. IX-2f) or 'Full volume' for the whole geometry.

- "Nodes_<domain_name>" contains a N x 3 array with N being the number of vertices of the geometry, the row index being the vertices id, and the three columns being the vertices coordinates.
- "Tetrahedron_<domain_name>" contains a C x 4 array with C being the number of tetrahedron cells of the geometry, the row index being the cell id, and the four columns being the vertices ids of the vertices that make this cell (i.e., the cell connectivity). Vertices id start either at 0 or 1 (cf. Fig. IX-2l).
- "Subdomain_<domain_name>" contains a C x 1 array with C being the number of tetrahedron cells of the geometry, the row index being the cell id, and the first column being the phase id (or grain id) of the phase the cell is belonging to.

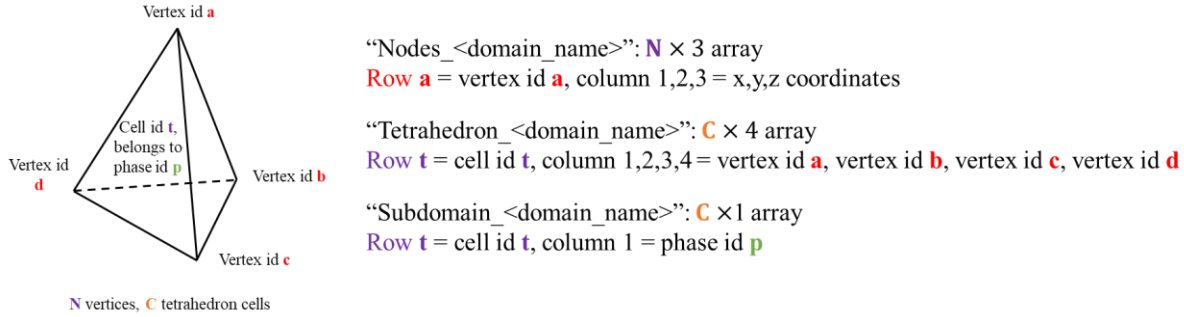


Figure IX-3. Vertices coordinates, cell connectivity, and phase id are saved in three dedicated files.

Meshes can be recreated using "Nodes_<domain_name>" and "Tetrahedron_<domain_name>" as explained in the next paragraph using FEniCS as example.

b. Creating the mesh for a FEM model using FEniCS

FEniCS[67] is an open-source (LGPLv3) computing platform for solving partial differential equations. The syntax is Python. The code below indicated how to create meshes for each phase, which is relevant for a segregated model[60], or for homogenization calculations. All meshes in this example have conforming interfaces.

First load the node and cell connectivity information generated with the mesh module (the code below import the .mat files):

```
from dolfin import * # Dolfin library
from mshr import * # Mesh library
import scipy.io as spio # Import MATLAB.mat array
import h5py # Import MATLAB.mat array
[…]
def load_mat(path,variable_name):
    try:
        res = spio.loadmat(path)[variable_name]
    except NotImplementedError:
        # raise NotImplementedError('Please use HDF reader for matlab v7.3 files')
        dict = {}
        f = h5py.File(path,'r')
        for k, v in list(f.items()):
            dict[k] = np.array(v)
        res = dict[variable_name]
        res = res.transpose()
    return res

Node_Electrolyte = load_mat(Node_Electrolyte_path,'node_')
Node_Anode = load_mat(Node_Anode_path,'node_')
Node_Cathode = load_mat(Node_Cathode_path,'node_')
Tetrahedron_Electrolyte = load_mat(Tetrahedron_Electrolyte_path,'elem_')
Tetrahedron_Anode = load_mat(Tetrahedron_Anode_path,'elem_')
Tetrahedron_Cathode = load_mat(Tetrahedron_Cathode_path,'elem_')
```

Then reconstruct the mesh, node per node, and cell per cell.

```
def create_mesh(node_,tetrahedroncell_):
        number_vertices=len(node_) # Number of vertices
        number_cells=len(tetrahedroncell_) # Number of cells
        mesh = Mesh() # Create an empty mesh object
        editor = MeshEditor() # Select the editor
        if version_check[dolfin.__version__]==version_check['2016.1.0']:
                editor.open(mesh, 3, 3) # (mesh, topological dimension, geometrical dimension)
        else:
                editor.open(mesh, 'tetrahedron', 3, 3, 1) #
                # (mesh, cell type, topological dimension, geometrical dim, polynomial degree)
        editor.init_vertices(number_vertices) # Set number of vertices
        editor.init_cells(number_cells) # Set number of cells
        # Set vertex location (vertex id,x,y)
        for vertex_id in range(number_vertices):
                x_ = node_[vertex_id][0] # Vertex coordinate 0
                y_ = node_[vertex_id][1] # Vertex coordinate 1
                z_ = node_[vertex_id][2] # Vertex coordinate 2
                vertex_id=int(vertex_id) # The vertex id must be an integer
                if version_check[dolfin.__version__]>=version_check['2018.1.0']:
                        editor.add_vertex(vertex_id, Point(x_, y_, z_)) # Add the vertex
                else:
                        editor.add_vertex(vertex_id, x_, y_, z_) # Add the vertex
        # Set cell (cell id, vertices id), for tetrahedrons
        for cell_id in range(number_cells):
                v0 = tetrahedroncell_[cell_id][0] # Vertex indice 0
                v1 = tetrahedroncell_[cell_id][1] # Vertex indice 1
                v2 = tetrahedroncell_[cell_id][2] # Vertex indice 2
                v3 = tetrahedroncell_[cell_id][3] # Vertex indice 3
                cell_id=int(cell_id); v0=int(v0); v1=int(v1); v2=int(v2); v3=int(v3)
                # The ids must be integers
                if version_check[dolfin.__version__]>=version_check['2018.1.0']:
                        editor.add_cell(cell_id, [v0, v1, v2, v3]) # Add the cell
                else:
                        editor.add_cell(cell_id, v0, v1, v2, v3) # Add the cell

        editor.close() # Close the editor
        return mesh

# Create electrolyte mesh
```

```
Mesh_Electrolyte_and_separator = create_mesh(Node_Electrolyte,Tetrahedron_Electrolyte)
# Create active material of the anode
Mesh_Anode = create_mesh(Node_Anode,Tetrahedron_Anode)
# Create active material of the cathode
Mesh_Cathode = create_mesh(Node_Cathode,Tetrahedron_Cathode)
```

The variables ‘mesh_*’ are FEniCS mesh objects. Vertices coordinates corresponds to the array indices. A rescaling is then required to match the physical dimension of the geometry:

```
def rescale_mesh(mesh,scaling_factor):
        coordinates = mesh.coordinates() # Get all the coordinates
        coordinates[:, :] *= scaling_factor # Apply scaling factor
        mesh.bounding_box_tree().build(mesh)
        # A cached search tree needs to be explicitly updated after moving/deforming a mesh
        return mesh

Mesh_Electrolyte_and_separator = rescale_mesh(Mesh_Electrolyte_and_separator,voxel_size)
Mesh_Anode = rescale_mesh(Mesh_Anode,voxel_size)
Mesh_Cathode = rescale_mesh(Mesh_Cathode,voxel_size)
```

The mesh reconstruction is done in serial. Therefore, it is recommended to perform the mesh reconstruction in a first file, called in serial, save it, and then call a second file, this time in parallel, that will import and partition the mesh and run FEM calculation on it (e.g., python3 create_mesh.py, and then mpirun -n 100 fem_calculation.py).

To save a mesh in FEniCS:

```
def save_mesh(folder, filename_withoutextension, mesh, mpi_comm, xml_format=False,
hdf5_format=False):
        if xml_format==False and hdf5_format==False:
                print('WARNING: Mesh saving error! You did not specified the mesh format!')
        elif xml_format:
                File(folder + filename_withoutextension + '.xml.gz') << mesh
        elif hdf5_format:
                hdf_ = HDF5File(mesh.mpi_comm(), folder + filename_withoutextension + '.h5', "w")
                hdf_.write(mesh, "mesh")
                hdf_.close()

save_mesh(folder_path,'mesh_electrolyte_separator',Mesh_Electrolyte_and_separator,mpi_comm,xml_fo
rmat=False,hdf5_format=True)
save_mesh(folder_path,'mesh_anode',Mesh_Anode,mpi_comm,xml_format=False,hdf5_format=True)
save_mesh(folder_path,'mesh_cathode',Mesh_Cathode,mpi_comm,xml_format=False,hdf5_format=True)
```

To import a mesh in FEniCS (mesh partitioning is automated, with load balanced equally among all processes):

```
def import_mesh(folder, filename_withoutextension, xml_format=False, hdf5_format=False):
        if xml_format==False and hdf5_format==False:
                print('WARNING: Mesh importation error! You did not specified the mesh format!')
        elif xml_format:
                mesh = Mesh(folder + filename_withoutextension + '.xml.gz')
        elif hdf5_format:
                mesh=Mesh()
                hdf5 = HDF5File(mesh.mpi_comm(), folder + filename_withoutextension + '.h5', 'r')
                hdf5.read(mesh, '/mesh', False)
        return mesh

mesh_electrolyte_separator = import_mesh(folder_path,'mesh_electrolyte_separator',
xml_format=False, hdf5_format=True)
mesh_anode = import_mesh(folder_path,'mesh_anode',xml_format=False,hdf5_format=True)
mesh_cathode = import_mesh(folder_path, 'mesh_cathode', xml_format=False, hdf5_format=True)
```

4. Example of mesh created with the module

If not say otherwise, all figures below have been obtained using the visualization options of the meshing module (cf. Fig. IX-2l).

a. Meshes of lithium-ion battery cells

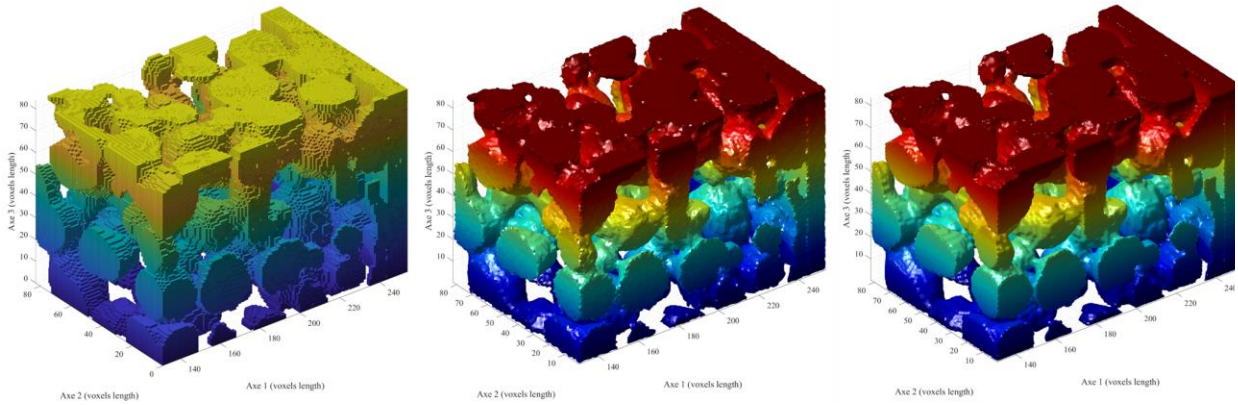


Figure IX-4a. Mesh of group {NMC cathode solid + current collector}. (left) Structured mesh, (center) unstructured mesh with Iso2mesh options: Radbound=1, distbound=1, Lowpass, iteration=1, alpha=0.5, keepratio=1, and max vol=10, and (right) unstructured mesh with Iso2mesh options: Radbound=0.75, distbound=1, Laplacian, iteration=2, alpha=0.5, keepratio=1, and max vol=5. Color is "z-axis" and mesh edges are not shown (cf. Fig. IX-2l for list of visualization options).

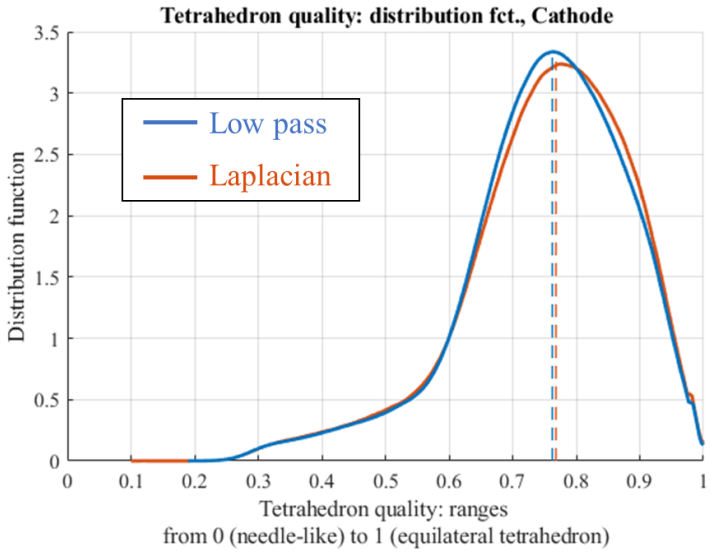


Figure IX-4b. Mesh quality distribution function for the cathode phase. Minimum cell quality is 0.102 using Laplacian smoothing, 0.187 using Low pass smoothing. Mesh quality of the structured mesh is uniform with a mean ratio η of 0.763.

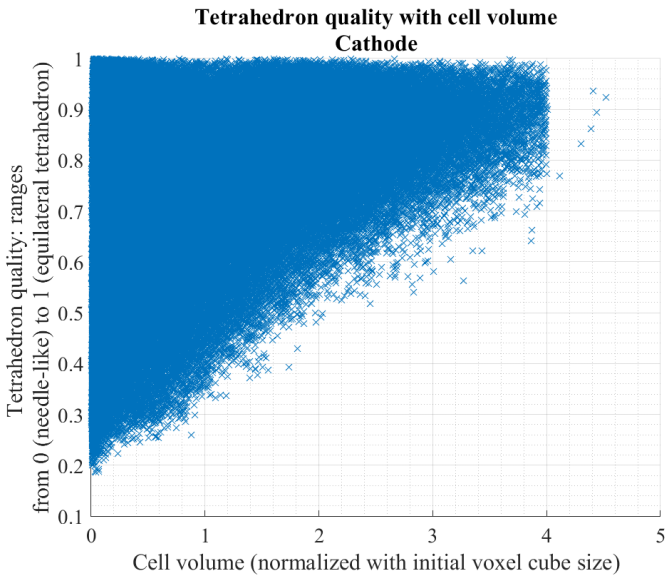


Figure IX-4c. Scatter plot of the mesh quality as function of the cell volume. The trend smaller cells have poorer mesh quality is systematic.

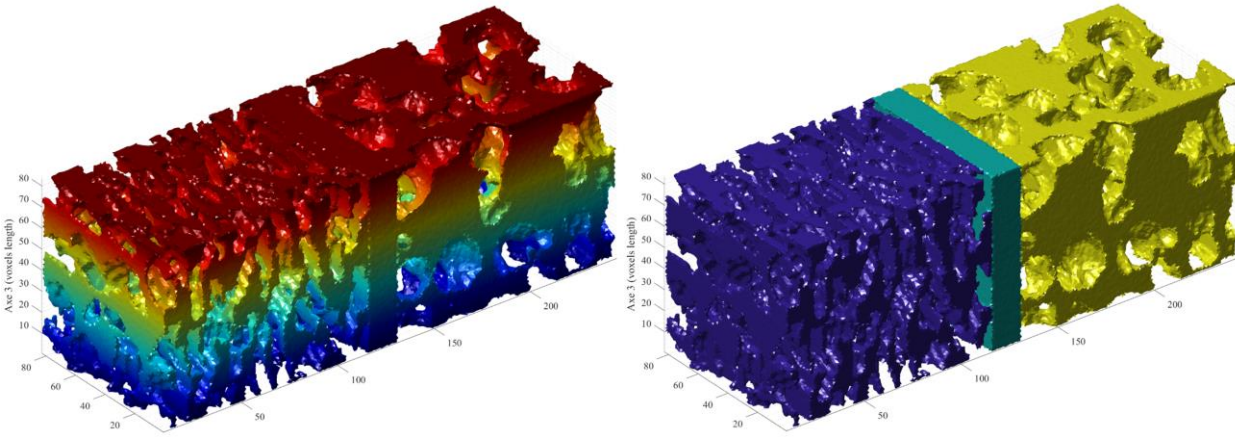


Figure IX-4d. Unstructured mesh of group {anode electrolyte + separator + cathode electrolyte} with Iso2mesh options: Radbound=1, distbound=1, Lowpass, iteration=1, alpha=0.5, keepratio=1, and max vol=10. (left) Color is z-axis, (right) color is phase id.

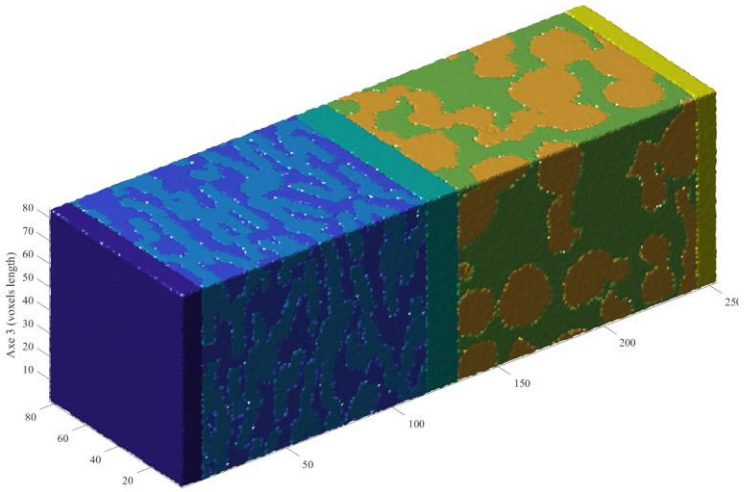


Figure IX-4e. Unstructured mesh of the whole cell volume, i.e., (from left to right) anode current collector, anode solid material and anode electrolyte, separator, cathode solid material and cathode electrolyte, and cathode current collector. Iso2mesh options: Radbound=1, distbound=1, Lowpass, iteration=1, alpha=0.5, keepratio=1, and max vol=10. Color is phase id.

Very large geometries with ~x0 millions of vertices can be achieved with very tortuous geometry:

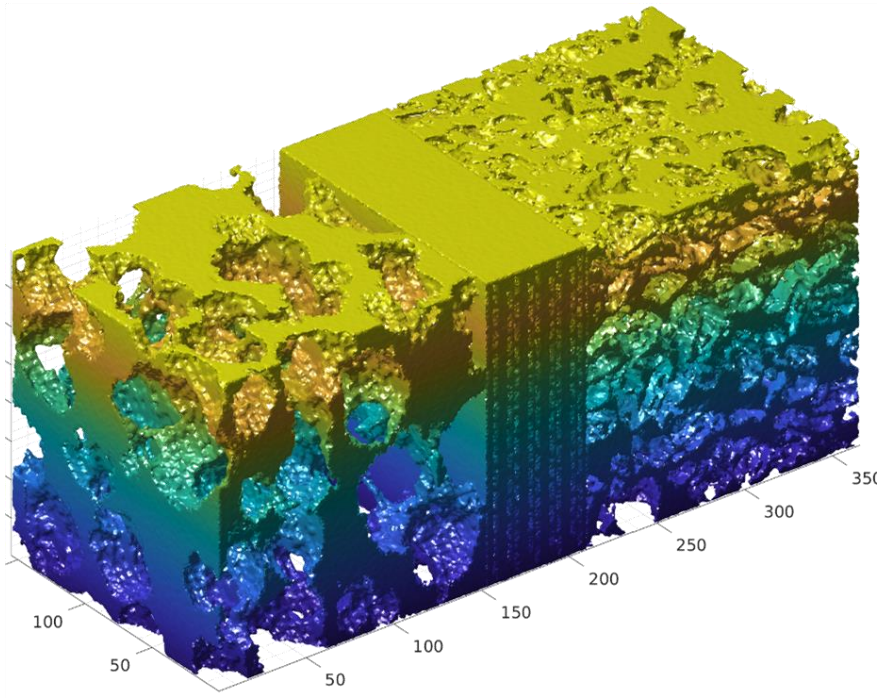


Figure IX-4f. Large full cell electrolyte geometry. From left to right: NMC cathode, separator, and graphite anode (separator is not homogenous in this example).

Geometries with a very smooth surface can be meshed too (usually though Laplacian smoothing with few smoothing iterations):

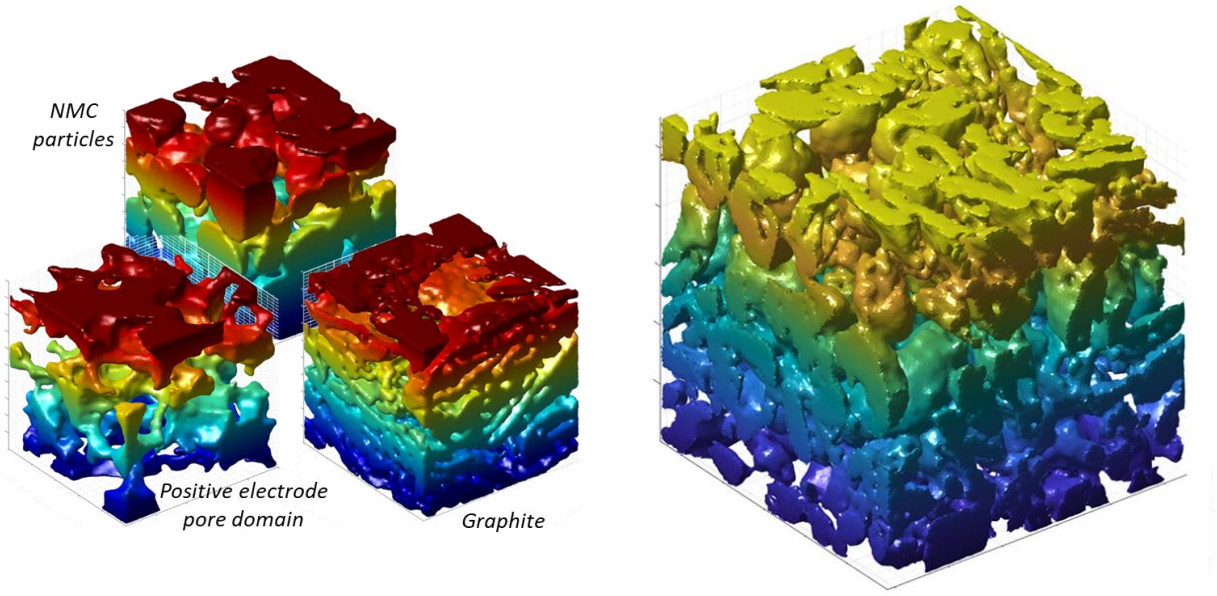


Figure IX-4g. Examples of meshed with very smooth interface.

Full cell meshes have been used in a segregated electrochemical LIB model[60] with success. Figures below corresponds to fast charge FEM simulation, images are from Paraview.

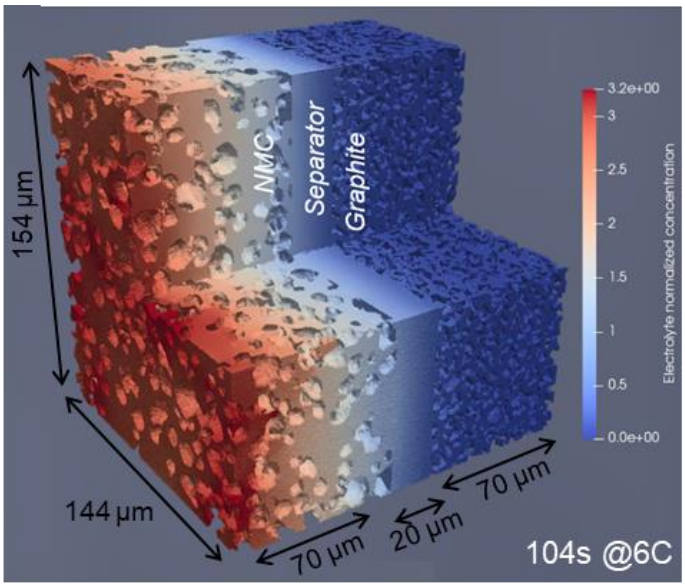


Figure IX-4h. Fast charge FEM electrochemical simulation performed on a mesh generated with this module. Electrolyte concentration is displayed. Visualization from Paraview.

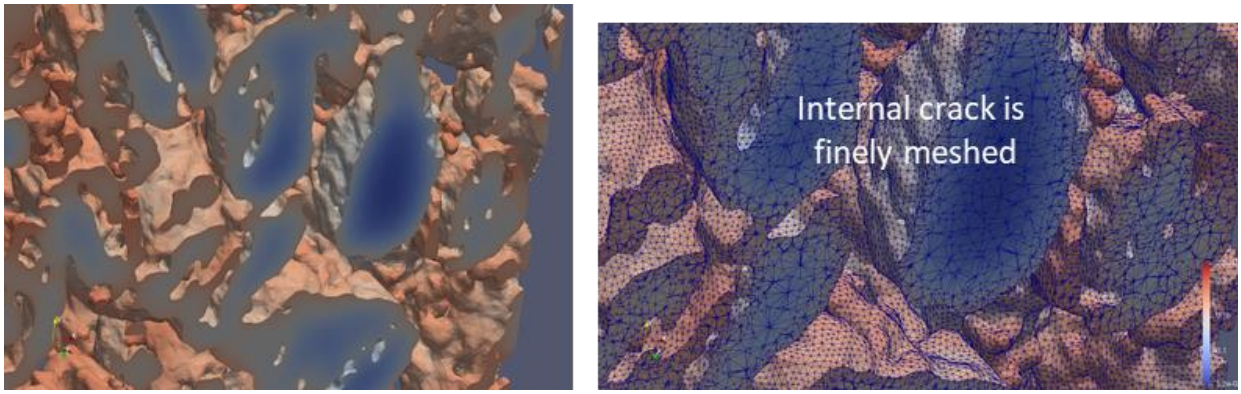


Figure IX-4i. Zoom on a FEM electrochemical simulation showing mesh details. Solid concentration is displayed. Visualization from Paraview.

b. Meshes of particle with polycrhistalline architecture (beta)

Iso2mesh can handle geometries with dozens of different label.

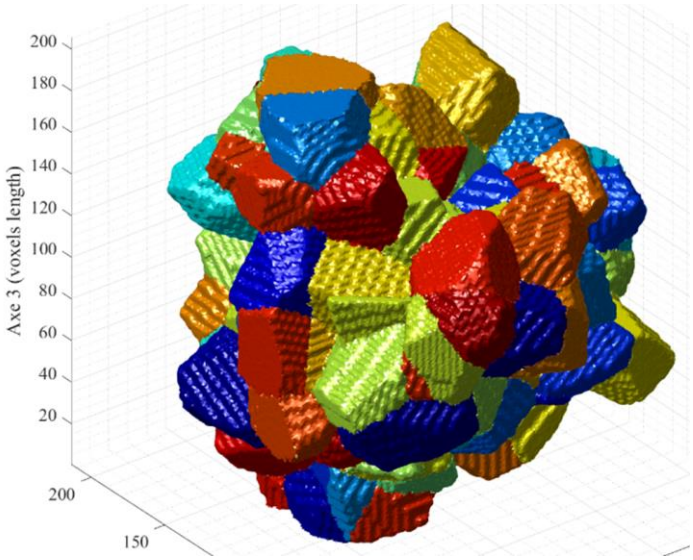


Figure IX-4j. Mesh of a polycrhistalline particle.

c. Meshes of programmatically determined geometry.

The examples below are here to illustrate the versality of Iso2mesh.

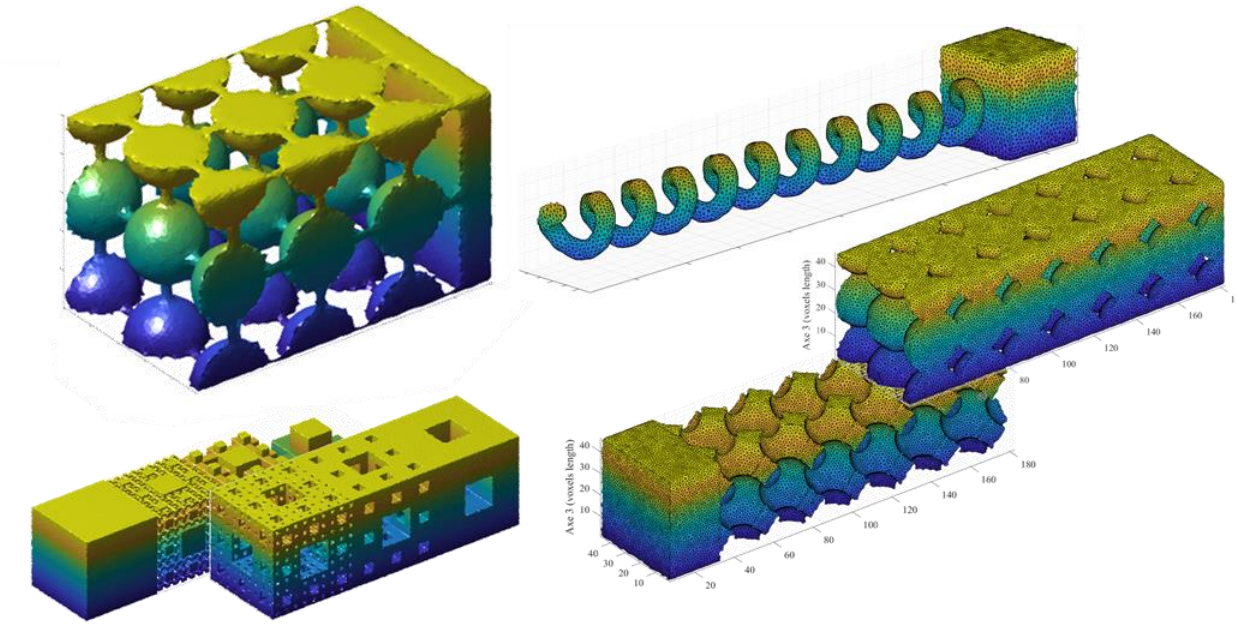


Figure IX-4k. Various test geometries meshed with the module.

5. Common Iso2mesh error and workaround

Most common error with Iso2mesh is "Two subfaces ... are found intersecting each other". Workarounds are:

- Reduce domain's size (through cropping or downscaling).
- Apply morphology opening (erosion + dilatation) to simply surface detail.
- Apply morphology opening (voxel-voxel connection) to remove ill-defined connection.
- Decrease the radbound value (e.g. 0.5 instead of default 1.0). Warning: this will result in much more refined meshes, thus it is very CPU/RAM expensive).
- Using the lowpass smoothing method with a small number of iteration (e.g., 1) and a small useralpha value (e.g., 0.5).

X. USEFUL STANDALONE FUNCTIONS

Functions presented below can be used as standalone. For more details, please open them with the MATLAB editor.

- src\Miscellaneous\function_save_tif

Save a 2D or 3D array in a tif file (one unique tif file even for a 3D volume) that you can later load in MATLAB with function_load_tif.m or in a third-party software such as ImageJ.

```
function_save_tif(array3D, pathname)
```

- src\Miscellaneous\function_load_tif

Load a tif file in a 3D MATLAB array with a user-defined format (e.g., uint8, default uint16).

```
[array,outcome] = function_load_tif(path,datatype_array)
```

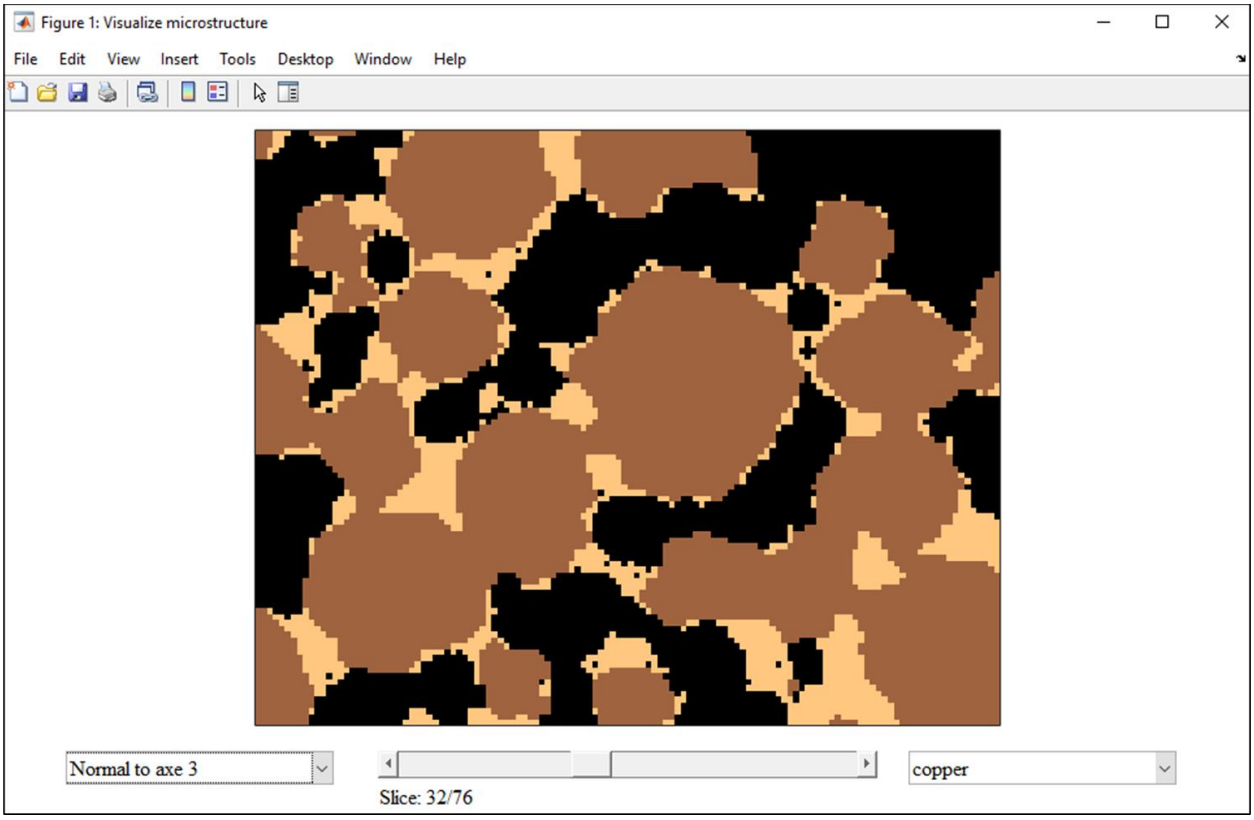
Or

```
[array,outcome] = function_load_tif(path)
```

- src\Miscellaneous\Microstructure_basic_visualization_interface.m

Use for simple 2D slice interactive visualization.

```
M=function_load_tif('C:\Users\fussegli\Desktop\Tifs\Example\nmc_bridge.tif');
Microstructure_basic_visualization_interface(M)
```



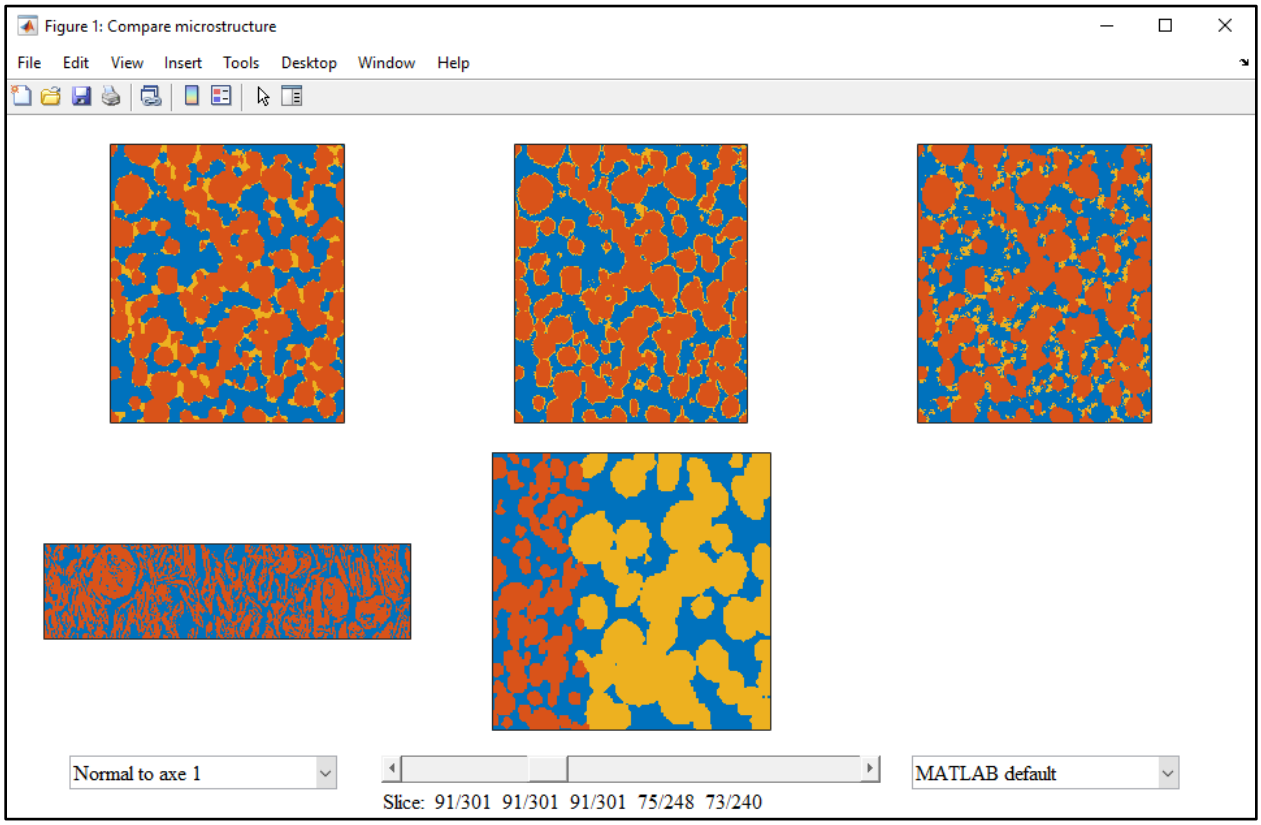
- src\Microstructure_comparison_visualization_interface.m

Based on Microstructure_basic_visualization_interface code but expanded to visualize n files side by side for comparison sake. Arrays do not need to be of the same size (the slider is based on the largest dimension among the arrays).

```
A=function_load_tif('C:\Users\fussegli\Desktop\nmc-1-cal_generatedCBD_bridge.tif');
B=function_load_tif('C:\Users\fussegli\Desktop\nmc-1-cal_generatedCBD_Energy_w0.001.tif');
C=function_load_tif('C:\Users\fussegli\Desktop\nmc-1-cal_generatedCBD_Energy_w0.999.tif');
D=function_load_tif('C:\Users\fussegli\Desktop\graphite-5-cal-blackisporepluscbd.tif');
E=function_load_tif('C:\Users\fussegli\Desktop\Bi_layer.tif');
```

S=size(*)	A	B	C	D	E
S(1)	301	301	301	248	240
S(2)	301	301	301	251	240
S(3)	251	251	251	961	240

```
Microstructure_comparison_visualization_interface(A, B, C, D, E);
```



Number of row and columns is automatically determined, spacings between axes is parametrized within the function and can be easily modified.

- src\Miscellaneous\Function_Writetable.m

  Use to save multiple MATLAB table in a csv file.

```
clear DATA_writetable
DATA_writetable.sheet(1).name='Excel_sheet1'; % Name of first excel sheet
DATA_writetable.sheet(1).table=Table_1;
% A MATLAB table (see https://www.mathworks.com/help/matlab/ref/table.html)
DATA_writetable.sheet(2).name='Whatever_name'; % Do not use empty space or too long name
DATA_writetable.sheet(2).table=Another_MATLAB_table;
[…]% Add others sheets
Folder = 'C:\Users\fussegli\Desktop\';
filename = 'example';
Function_Writetable(folder,filename,DATA_writetable)
```

- src\Miscellaneous\ function_remove_emptyandspecialcharacter_string.m

  Use to remove or replace specific characters (e.g., ' ' ',' '-') you would not like to use for filename when saving.

```
[ str_new ] = function_remove_emptyandspecialcharacter_string(str_old)
```

XI. KNOWS ISSUES/FREQUENTLY ASKED QUESTIONS

- Documentation link does not open the pdf documentation with MATLAB displaying instead a warning message.

The error could happen if you have packaged MATBOX in a MATLAP app and did not add the documentation pdf manually as indicated in the installation section (look at "shared ressources and helper files" of MATLAB package app and add the documentation pdf).

- Image is uniform after saving or while modifying it with the filtering/segmentation module:

Try changing the data type before saving/modifying the image. Recommended is uint8 or uint16. If problem persists, edit file src\Miscellaneous\function_save_tif.m.

- MATLAB does not respond or crash during microstructure characterization, with no error displayed.

You may have reached MATLAB's maximum name length of 63, which can occur if your main folder is already long. When this occurs, MATLAB, unfortunately, does not always show a warning and stalls indefinitely while trying to save a file. Try using a short name main folder (such as C:\Calc\).

- MATLAB fails to save table in an excel file:

```
Warning: File not found or permission denied
> In Function_Writetable (line 40)
  In Function_Specificinterface_direct (line 209)
  In function_microstructure_characterization_segmentedvolume (line 474)
  In microstructure_characterization_GUI/pushbutton_runallcalculations_Callback (line 2947)
Error using Interface.000208DA_0000_0000_C000_000000000046/SaveAs
Invoke Error, Dispatch Exception:
Source: Microsoft Excel
Description: Cannot access 'your_file.xls'.
Help File: xlmain11.chm
Help Context ID: 0

Error in Function_Writetable (line 44)
    ExcelWorkbook.SaveAs(full_path_xls,1); % Save it
```

Open Task Manager and kill all EXCEL process. Manually delete your_file.xls and retry.

XII. REFERENCES

1. Getting started with microstructure analysis

Algorithms presented in this toolbox have been utilized in [17,34,43]

You can find below a selection of references sorted by topics you can read to get into microstructure analysis:

- *Image segmentation.* Review of segmentation methods can be found in [24,25].
- *Microstructure numerical generation*. Generation methods for active materials are described in [14,15], and for additive phases in [8,19]
- *Representative Volume Element*. RVE size being parameter specific is discussed in [43,44].
- *Voxel size dependence analysis*. Example of voxel size dependence analysis can be found in [17,34,40]. Fractal nature of some parameter is discussed in [34,42].
- *Particle size*. Impact of particle size on battery electrode performances is discussed in [37–39].
- *Particle anisotropy/non-spherical shape*. Impact of particle shape on effective diffusion coefficient is discussed in [34,36].
- *Tortuosity factor*. Homogenization method is well explained in [40]. Impact of boundary conditions and edged effects are detailed in [43]. Finite difference method (used in this toolbox) is introduced in [2]. Comparison between finite element and finite difference calculations is discussed in [17] (see supplementary materials). Experimental determination of tortuosity factor are detailed in [50,51] Comparison between numerical and experimental determination of tortuosity factor is also discussed in [17]. Re-derivation of the Bruggeman analytical relationship along with a discussion on its limitations can be found in [48]
- *Sinuosity and constrictivity*. Geometric interpretation of tortuosity factor is discussed in length in [34,46,68]
- *Electrochemical macroscale modeling (use microstructure properties)*. Most macroscale electrochemical battery models use the pseudo-2D formulation originally proposed by Newman and coworkers [32]. A more recent writing of these equations can be found in K. Smith articles [33,69].
- *Meshing*.
- *Electrochemical microstructure scale modeling (use microstructure mesh)*. Look at [60].

2. Documentation references

1. NREL's Computer-Aided Engineering for Electric Drive Vehicle Batteries (CAEBAT). *NREL's Computer-Aided Engineering for Electric Drive Vehicle Batteries (CAEBAT)* https://www.nrel.gov/transportation/caebat-microstructure-applications.html (n.d.).

2. Cooper, S. J., Bertei, A., Shearing, P. R., Kilner, J. A. & Brandon, N. P. TauFactor: An open-source application for calculating tortuosity factors from tomographic data. *Softwarex* **5**, 203–210 (2016).

3. Fang, Q. & Boas, D. A. Tetrahedral mesh generation from volumetric binary and grayscale images. *2009 Ieee Int Symposium Biomed Imaging Nano Macro* 1142–1145 (2009) doi:10.1109/isbi.2009.5193259.

4. Crosby, K. Column Converter for Excel, MATLAB Central File Exchange. *Column Converter for Excel, MATLAB Central File Exchange* https://www.mathworks.com/matlabcentral/fileexchange/28343-column-converter-for-excel (2020).

5. Jan. DataHash, MATLAB Central File Exchange. *DataHash, MATLAB Central File Exchange* https://www.mathworks.com/matlabcentral/fileexchange/31272-datahash (n.d.).

6. Tanaka, M. Noise Level Estimation from a Single Image, MATLAB Central File Exchange. *Noise Level Estimation from a Single Image, MATLAB Central File Exchange* https://www.mathworks.com/matlabcentral/fileexchange/36921-noise-level-estimation-from-a-single-image (2020).

7. Altman, Y. ScreenCapture - get a screen-capture of a figure frame or component. *ScreenCapture - get a screen-capture of a figure frame or component* https://www.mathworks.com/matlabcentral/fileexchange/24323-screencapture-get-a-screen-capture-of-a-figure-frame-or-component (2021).

8. Mistry, A., Smith, K. & Mukherjee, P. P. Secondary Phase Stochastics in Lithium-ion Battery Electrodes. *ACS Applied Materials & Interfaces* (2018) doi:10.1021/acsami.7b17771.

9. Conder, J., Marino, C., Novák, P. & Villevieille, C. Do imaging techniques add real value to the development of better post-Li-ion batteries? *Journal of Materials Chemistry A* **6**, 3304–3327 (2018).

10. Pietsch, P. & Wood, V. X-Ray Tomography for Lithium Ion Battery Research: A Practical Guide. *Annual Review of Materials Research* **47**, 1–29 (2016).

11. Eastwood, D. S. *et al.* The application of phase contrast X-ray techniques for imaging Li-ion battery electrodes. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* **324**, 118–123 (2014).

12. Xu, C. S. *et al.* Enhanced FIB-SEM systems for large-volume 3D imaging. *Elife* **6**, e25916 (2017).

13. NREL battery Microstructures Library. https://www.nrel.gov/transportation/microstructure.html (2020).

14. Moussaoui, H. *et al.* Stochastic geometrical modeling of solid oxide cells electrodes validated on 3D reconstructions. *Comp Mater Sci* **143**, 262–276 (2018).

15. Gayon-Lombardo, A., Mosser, L., Brandon, N. P. & Cooper, S. J. Pores for thought: generative adversarial networks for stochastic reconstruction of 3D multi-phase electrode microstructures with periodic boundaries. *Npj Comput Mater* **6**, 82 (2020).

16. Robertson, D. C. *et al.* Effect of Anode Porosity and Temperature on the Performance and Lithium Plating During Fast-Charging of Lithium-Ion Cells. *Energy Technol-ger* (2020) doi:10.1002/ente.202000666.

17. Usseglio-Viretta, F. L. *et al.* Resolving the Discrepancy in Tortuosity Factor Estimation for Li-Ion Battery Electrodes through Micro-Macro Modeling and Experiment. *Journal of The Electrochemical Society* **165**, A3403–A3426 (2018).

18. Mistry, A. N., Smith, K. & Mukherjee, P. P. Electrochemistry Coupled Mesoscale Complexations in Electrodes Lead to Thermo-Electrochemical Extremes. *Acs Appl Mater Inter* **10**, 28644–28655 (2018).

19. Hein, S. *et al.* Influence of Conductive Additives and Binder on the Impedance of Lithium-Ion Battery Electrodes: Effect of Morphology. *J Electrochem Soc* **167**, 013546 (2020).

20. Song, C., Wang, P. & Makse, H. A. A phase diagram for jammed matter. *Nature* **453**, 629–632 (2008).

21. Mistry, A., Smith, K. & Mukherjee, P. P. Stochasticity at Scales Leads to Lithium Intercalation Cascade. *Acs Appl Mater Inter* **12**, 16359–16366 (2020).

22. Perona, P. & Malik, J. Scale-space and edge detection using anisotropic diffusion. **12**, (1990).

23. Buades, A., Coll, B. & Morel, J.-M. A Non-Local Algorithm for Image Denoising. (n.d.) doi:10.1109/cvpr.2005.38.

24. Khan, M. A Survey: Image Segmentation Techniques. *International Journal of Future Computer and Communication* 89–93 (2014) doi:10.7763/IJFCC.2014.V3.274.

25. Wirjadi, O. Survey of 3D Image Segmentation Methods. *Berichte des Fraunhofer ITWM Tech. Rep* (2007).

26. Julie, V. *et al.* 3D phase mapping of solid oxide fuel cell YSZ/Ni cermet at the nanoscale by holographic X-ray nanotomography. **243**, (2013).

27. Otsu, N. A Threshold Selection Method from Gray-Level Histograms. **9**, (1979).

28. Pizer, S. M. *et al.* Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing* **39**, 355–368 (1986).

29. Zuiderveld, K. *Contrast Limited Adaptive Histograph Equalization*. (1994).

30. Joos, J., Ender, M., Rotscholl, I., Menzler, N. H. & Ivers-Tiffée, E. Quantification of double-layer Ni/YSZ fuel cell anodes from focused ion beam tomography data. *Journal of Power Sources* **246**, 819–830 (2014).

31. Usseglio-Viretta, F. L. E. Optimization of the performances and the robustness of an electrolyser at high temperatures. *Grenoble Alpes University* (2015).

32. Doyle, M. Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell. *Journal of The Electrochemical Society* **140**, 1526 (1993).

33. Smith, K. & Wang, C.-Y. Power and thermal characterization of a lithium-ion battery pack for hybrid-electric vehicles. *Journal of Power Sources* **160**, (2006).

34. USSEGLIO-VIRETTA, F. L. E. *et al.* Quantitative relationships between pore tortuosity, pore topology, and solid particle morphology using a novel discrete particle size algorithm. *J Electrochem Soc* (2020) doi:10.1149/1945-7111/ab913b.

35. Singh, M., Kaiser, J. & Hahn, H. Thick Electrodes for High Energy Lithium Ion Batteries. *J Electrochem Soc* **162**, A1196–A1201 (2015).

36. Ebner, M., Chung, D., García, E. R. & Wood, V. Electrodes: Tortuosity Anisotropy in Lithium-Ion Battery Electrodes (Adv. Energy Mater. 5/2014). *Advanced Energy Materials* **4**, (2014).

37. Röder, F., Sonntag, S., Schröder, D. & Krewer, U. Simulating the Impact of Particle Size Distribution on the Performance of Graphite Electrodes in Lithium-Ion Batteries. *Energy Technol-ger* **4**, 1588–1597 (2016).

38. Meyer, M., Komsiyska, L., Lenz, B. & Agert, C. Study of the local SOC distribution in a lithium-ion battery by physical and electrochemical modeling and simulation. *Appl Math Model* **37**, 2016–2027 (2012).

39. Lee, K. & Kum, D. The Impact of Inhomogeneous Particle Size Distribution on Li-ion Cell Performance Under Galvanostatic and Transient Loads. *2016 Ieee Transp Electrification Conf Expo Asia-pacific Itec Asia-pacific* 454–459 (2016) doi:10.1109/itec-ap.2016.7512997.

40. Laurencin, J. *et al.* Characterisation of Solid Oxide Fuel Cell Ni–8YSZ substrate by synchrotron X-ray nano-tomography: from 3D reconstruction to microstructure quantification. **198**, (2012).

41. Delette, G. *et al.* Thermo-elastic properties of SOFC/SOEC electrode materials determined from three-dimensional microstructural reconstructions. *International Journal of Hydrogen Energy* **38**, 12379–12391 (2013).

42. Bertei, A. *et al.* The fractal nature of the three-phase boundary: A heuristic approach to the degradation of nanostructured solid oxide fuel cell anodes. *Nano Energy* **38**, 526–536 (2017).

43. Usseglio-Viretta, F. & Smith, K. Quantitative Microstructure Characterization of a NMC Electrode. *ECS Transactions* **77**, 1095–1118 (2017).

44. Kanit, T., Forest, S., Galliet, I., Mounoury, V. & Jeulin, D. Determination of the size of the representative volume element for random composites: statistical and numerical approach. **40**, (2003).

45. Tjaden, B., Brett, D. J. & Shearing, P. R. Tortuosity in electrochemical devices: a review of calculation approaches. *International Materials Reviews* 1–21 (2016) doi:10.1080/09506608.2016.1249995.

46. Holzer, L. *et al.* The influence of constrictivity on the effective transport properties of porous layers in electrolysis and fuel cells. *J Mater Sci* **48**, 2934–2952 (2013).

47. Bruggeman, D. Berechnung verschiedener physikalischer Konstanten von heterogenen Substanzen. I. Dielektrizitätskonstanten und Leitfähigkeiten der Mischkörper aus isotropen Substanzen. *Annalen der Physik* **416**, (1935).

48. Tjaden, B., Cooper, S. J., Brett, D. J., Kramer, D. & Shearing, P. R. On the origin and application of the Bruggeman correlation for analysing transport phenomena in electrochemical systems. *Current Opinion in Chemical Engineering* **12**, 44–51 (2016).

49. Archie, G. E. The Electrical Resistivity Log as an Aid in Determining Some Reservoir Characteristics . *Dallas Meeting* (1941).

50. Thorat, I. V. *et al.* Quantifying tortuosity in porous Li-ion battery materials. *Journal of Power Sources* **188**, 592–600 (2009).

51. Landesfeind, J., Hattendorff, J., Ehrl, A., Wall, W. A. & Gasteiger, H. A. Tortuosity Determination of Battery Electrodes and Separators by Impedance Spectroscopy. *J Electrochem Soc* **163**, A1373–A1387 (2016).

52. Usseglio-Viretta, F. *et al.* Quantitative microstructure characterization of a Ni–YSZ bi-layer coupled with simulated electrode polarisation. *Journal of Power Sources* **256**, 394–403 (2014).

53. Joos, J. *et al.* Detailed Microstructure Analysis and 3D Simulations of Porous Electrodes. **35**, (2011).

54. Rajon, D. A., Patton, P. W., Shah, A. P., Watchman, C. J. & Bolch, W. E. Surface area overestimation within three-dimensional digital images and its consequence for skeletal dosimetry. **29**, (2002).

55. Weisstein, E. Oblate Spheroid. *MathWorld--A Wolfram Web Resource* https://mathworld.wolfram.com/OblateSpheroid.html (n.d.).

56. Gelb, L. D. & Gubbins, K. Pore Size Distributions in Porous Glasses: A Computer Simulation Study. *Langmuir* **15**, 305–308 (1999).

57. Münch, B. & Holzer, L. Contradicting Geometrical Concepts in Pore Size Analysis Attained with Electron Microscopy and Mercury Intrusion. **91**, (2008).

58. Beucher, S. & Lantuejoul, C. Use of Watersheds in Contour Detection. *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France* (1979).

59. KENDALL, M. G. A NEW MEASURE OF RANK CORRELATION. *Biometrika* **30**, 81–93 (1938).

60. Allen, J. M., Chang, J., Usseglio-Viretta, F. L. E., Graf, P. & Smith, K. A Segregated Approach for Modeling the Electrochemistry in the 3-D Microstructure of Li-Ion Batteries and Its Acceleration Using Block Preconditioners. *J Sci Comput* **86**, 42 (2021).

61. Liu, A. & Joe, B. Relationship between tetrahedron shape measures. *Bit* **34**, 268–287 (1994).

62. Xu, Z., Rahman, M. M., Mu, L., Liu, Y. & Lin, F. Chemomechanical behaviors of layered cathode materials in alkali metal ion batteries. *J Mater Chem A* **6**, 21859–21884 (2018).

63. Mao, Y. *et al.* High-Voltage Charging-Induced Strain, Heterogeneity, and Micro-Cracks in Secondary Particles of a Nickel-Rich Layered Cathode Material. *Advanced Functional Materials* **29**, 1900247 (2019).

64. Yang, Y. *et al.* Quantification of Heterogeneous Degradation in Li-Ion Batteries. *Advanced Energy Materials* 1900674 (2019) doi:10.1002/aenm.201900674.

65. Quinn, A. *et al.* Electron Backscatter Diffraction for Investigating Lithium-Ion Electrode Particle Architectures. *Cell Reports Phys Sci* **1**, 100137 (2020).

66. Furat, O. *et al.* Mapping the architecture of single lithium ion electrode particles in 3D, using electron backscatter diffraction and machine learning segmentation. *J Power Sources* **483**, 229148 (2021).

67. Alnaes, M. S. *et al.* The FEniCS Project Version 1.5. *Archive of Numerical Software* **3**, (2015).

68. Holzer, L. *et al.* Redox cycling of Ni–YSZ anodes for solid oxide fuel cells: Influence of tortuosity, constriction and percolation factors on the effective transport properties. *Journal of Power Sources* **242**, 179–194 (2013).

69. Smith, K. A., Rahn, C. D. & Wang, C.-Y. Control oriented 1D electrochemical model of lithium ion battery. *Energy Conversion and Management* **48**, 2565–2578 (2007).