Deep Brain Lead Profile Script

Version 1.0

Date: 02.04.2019

Created by: Trieu

Purpose of this tutorial: To simulate the effect of DBS it is necessary to have the DBS lead geometry defined. In order to use many different types of DBS lead for the in-house developed simulation platform, the DBS lead’s profiles need to be created in a certain format. This document describes the routine that people should follow in order to create a new DBS lead profile that can be used in the simulation platform.

DBS\_lead\_position\_V10.py (manages the scripts)

DBS lead profile

File with a brain geometry

Figure 1. Components of the geometrical modelling in the simulation platform.

1. **DBS lead geometry script.**

**1.1 Definitions**

Geometry of a DBS lead profile has 2 main parts: volumes and surfaces (area)

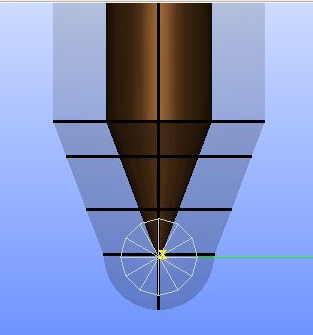
Volumes: region of interest (ROI), encapsulation layer inside ROI, encapsulation layer outside ROI, and DBS lead volume.

Surfaces correspond to contacts.

Encapsulation layer is the offset volume of the DBS lead volume with the thickness defined as variable encap\_thickness (the name is predefined to use with extra code).

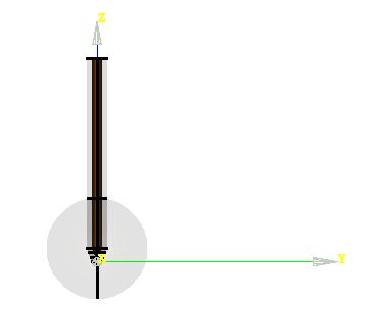
ROI: a sphere, where neuron models are placed, which radius is also defined as a variable ROI\_radial ( The name is predefined to use with extra code ) . ROI\_radial will be later assigned automatically depending on the extent of the neuron models.

Contact’s surfaces



DBS lead volume

Encapsulation layer



ROI

Encapsulated layer

outer ROI

Encapsulated layer

inner ROI

* 1. **How to define a variable in Salome Script**

The initial geometry is built with constant values. After generating python script, the constant values are replaced with predefined variables.

Example: the electrode is a cylinder of 0.5 mm in diameter. We can choose an arbitrary (but reasonable) value for the encap. layer thickness, e.g. 0.2 mm.

When building the encapsulation layer as an additional cylinder, the radius should be put to 0.5 mm + 0.2 mm

Encap\_cylinder = geompy.MakeCylinderRH(0.7, 20.0) #python equivalent

Later when a Python study is generated, the value 0.7 is replaced with the lead radius + encap\_thickness

Encap\_cylinder = geompy.MakeCylinderRH(0.5+encap\_thickness, 20.0) #python equivalent

The same procedure is applied for defining ROI, where ROI\_radial will be placed as a radius of the ROI sphere.

It is important to have the exact variable name of encap\_thickness, ROI\_radial, so they can run with extra code 1.

* 1. **DBS lead Geometry build steps (more details on the entities, how exactly they should be defined)**
* Build DBS lead volume (i.e. the electrode itself as solid)
* Build encapsulation layer offset from the DBS lead (encapsulation layer does not include the DBS lead volume!)
* Build ROI with a constant radius, naming it “ROI” (ROI=Sphere-encap. layer- DBS lead volume)
* Build encapsulation layer inside ROI naming it “encap\_inner\_ROI” (=Sphere ∩ encapsulation layer)
* Build encapsulation layer outside ROI naming it “encap\_outer\_ROI” (=encapsulation layer- Sphere)
* Fuse all volume objects (DBS lead,ROI, encapsulation layer inside ROI, encapsulation layer outside ROI) together, naming “Fuse\_all” :

Fuse\_all = geompy.MakeFuseList([ROI, encap\_in, encap\_out ,DBS\_lead ], True, True) #python equivalent

* Build surface contacts naming them “Contact\_1”, “Contact\_2”, etc.
* Dump Python study (save it as Electode\_name+”\_profile.py”)
* Replace constant values in Dump study file with encap\_thickness, ROI\_radial.

**2.Extra code Definitions:**

**2.1 DBS lead profile** is a python script with Salome commands to build the geometry of the DBS lead, ROI and encapsulation layer. This python script was generated above, and it has 3 main parts:

Import : start from beginning until “import SALOMEDS”

geometry commands : from “import SALOMEDS” until before “geompy.addToStudy( O, 'O' )”

geompy.addToStudy: from “geompy.addToStudy( O, 'O' )” until end of file

**2.2 Extra codes:** the codes are inserted in the DBS lead geometry script so that the change of position of the DBS lead in the brain can be done and the Partitions’ IDs can be identified. The codes can be found in Template\_profile.py. Extra codes consist of three parts, which start with a note like:

“ ################################extra code 1 V11 13/1/19#######################”

And

“ ################################end of extra code 1 ############################”

Extra code 1: imports the necessary python path for running the files and variables defined in the simulation platform. (insert after ### GEOM component)

Extra code 2: moves the DBS lead to a new position and finds the new partition (insert before geompy.addToStudy( O, 'O' ))

Extra code 3: contains addtostudy commands for the new geometry (insert after the last geompy.addToStudy() command)

1. **Interface between Geometry and extra codes:**
   1. **Interface with Extra code 1 and geometry script**

Extra code 1 and extra code 2 have predefined variable name: encap\_thickness, ROI\_radial. It is important that in the DBS lead profile script these variables have exactly the same names.

**3.2 volume objects and surface objects.**

The extra codes can be reused for many different geometry scripts. The interface code in extra code 1 **needs to be modified** depending on the number of electrode’s contacts.

############## Geometry and extra code interface #################################

Change according to the number of contacts!

ContactObject1 = [Contact\_1,Contact\_2] ## the name as previously defined in GUI

Contact\_name1 = ['Contact1\_1','Contact1\_2'] ## fixed notation!

E.g, if we have three contacts, then

ContactObject1 = [Contact\_1,Contact\_2,Contact\_3] ## the name as previously defined in GUI

Contact\_name1 = ['Contact1\_1','Contact1\_2', 'Contact1\_3'] ## fixed notation!

if(Lead2nd\_Enable): ################## 2nd LEAD ##########################

Change according to the number of contacts!

Contact\_name2 = ['Contact2\_1','Contact2\_2']

##############################################################################

**3.4 Generate DBS lead position.**

To generate the position file of DBS lead, the python script “DBS\_lead\_position\_V10.py” is run in the same directory with the profile file. “DBS\_lead\_position\_V10.py” will insert the position variable and other parameters value to the DBS lead profile file to generate DBS lead position in the same folder. Every times, the position of the DBS lead changes in the simulation platform, a new position file is generated and rewritten in the folder.

If you want to test your script manually, run “python DBS\_lead\_position\_V10.py” in terminal to create a position file. You can also change the position and angles of implantation, encapsulation layer thickness, etc. in the DBS\_lead\_position\_V10.py (the last line in the file)

After the Electrode\_name+”\_position.py” script is generated, we can manually run it with “salome Electrode\_name+\_position.py”

**3.5 files needed to run the position script:**

- DBS\_lead\_position\_V10.py (insert position parameters to profile file and generate DBS lead position file)

- Profile\_Process\_V6.py (used by DBS\_lead\_position\_V10.py to process python script )

- PrintLog.py ( used to record log file of Salome)

- brain file (formats: BREP, STEP, IGES, STL). Brain\_dummy.brep can be used to test the script

- DBS lead profile script should have the format : “xxxxxx\_profile.py' so the DBS\_lead\_position\_V10.py can detect it.

1. **DBS lead Profile build steps**

* Build geometry script ( part 1)
* Insert extra code 1 , extra code 2, extra code 3 in the positions (part 3)
* Change the interface variables according to number of contacts
* Test the profile by changing with different.

1. **Meshing**

Copy everything after extra code 3 from the Template\_profile.py to your profile file and follow the instructions in the code (given as comments). Test the script repeating **3.4**