Instruction

This instruction will guide you to install STEPS as well as the geometry preparation and visualization toolkits described in “Python-Based Geometry Preparation and Simulation Visualization Toolkits for STEPS”. It will also demonstrate the examples described in the paper.

Prerequisites

STEPS compilation requires C/C++ compiler such as gcc/g++. For some platforms it may require to rebuild the swig files, in which case SWIG (http://www.swig.org/) is needed.

The Geometry Preparation toolkit requires CUBIT, a trial version can be requested via http://www.csimsoft.com/

The visualization toolkit requires Numpy (http://www.numpy.org/), PyQt4 (http://www.riverbankcomputing.com/software/pyqt/download) and PyQtGraph(http://www.pyqtgraph.org/).

Installation

1. STEPS

STEPS can be installed using the standard Python distutils approach. Here we described the installation under Mac OSX. More information can be found in (http://docs.python.org/2/library/distutils.html)

1.1 Download STEPS from http://steps.sourceforge.net
1.2 Open a terminal, unzip STEPS by typing in **tar -xzf STEPS-2.2.0.tar.gz**
1.3 type in **cd STEPS-2.2.0**
1.4 (optional, only needed if 1.5 fails) type in **./runswig**
1.5 type in **python setup.py build**
1.6 type in **python setup.py install**, you may need to add **sudo** to gain permission.

To check if STEPS is installed successfully, start python and type in **import steps**, a license message should appear.

2. Geometry Preparation Toolkit

The geometry preparation toolkit can be installed with or without STEPS integration with CUBIT, if it is installed without integration, preparation data is exported to a file and can be import to STEPS mesh using STEPS importing utilities. If it is installed with integration, preparation data can be directly written to Tetmesh object in CUBIT.

2.1 In CUBIT’s script window, check the PYTHONPATH and Python executable it uses by typing in

import sys
print sys.path

2.2 With/Without integration:

copy the steps-cubit folder under STEPS-2.2.0/independent to CUBIT’s PYTHONPATH directory.

2.3 (optional) With integration:

install STEPS using the same Python executable CUBIT uses.

3. Visualization Toolkit

If all prerequisites and STEPS are installed correctly the visualization toolkit can be import using **import steps.visual** in Python.

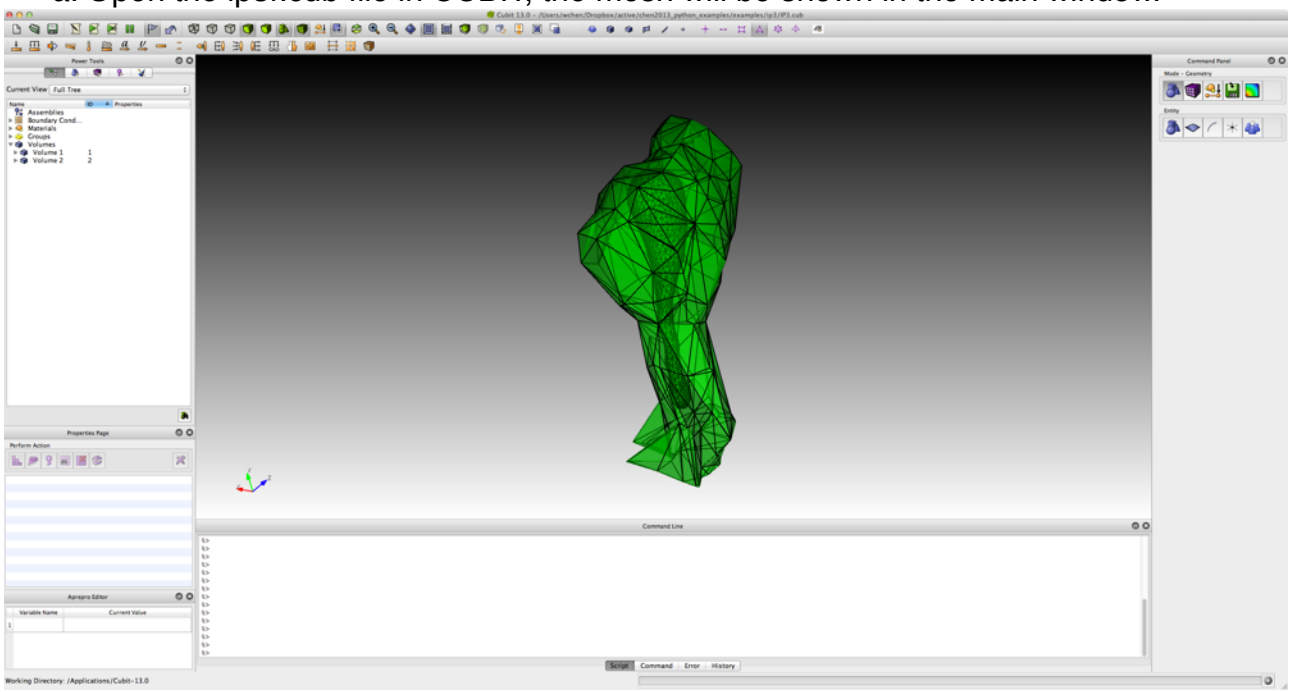
Examples

1. IP3 Receptor Model

Files for the example are located in examples/ip3r.

Mesh Preparation:

a. Open the ip3r.cub file in CUBIT, the mesh will be shown in the main window.



b. In the script panel, type in

import os

os.chdir(PATH_TO_THE_EXAMPLE_FOLDER)

To change the working path, then type in

import steps_cubit as scubit

This will import the Geometry Preparation toolkit as scubit, then type in

shadow_mesh = scubit.ShadowMesh()

To create a ShadowMesh object for the preparation.

c. Select Volume 1 from the left hand side panel, then in the script panel type in

shadow_cyt = scubit.selectedVolumesAsComp("cyt", shadow_mesh, ['vsys'])

This will assign the outer volume (volume 1) as the cytosol and associates it with volume system “vsys”.

d. Select Volume 2 and type in

shadow_ER = scubit.selectedVolumesAsComp("ER", shadow_mesh, ['vsys'])

The command assign Volume 2 as ER and associates it with volume system “vsys”.

e. Expend Volume 2 and select Surface 21, then type in

shadow_memb = scubit.selectedSurfacesAsPatch("memb", shadow_mesh, ['ssys'], shadow_ER, shadow_cyt)

this create a membrane using Triangles on Surface 21 and associates it with surface system “ssys”.

f.

f1. (With STEPS integration) type in

import steps.utilities.meshio as meshio

tetmesh = meshio.importAbaqus2('tets.inp', 'tris.inp', 1e-6, shadow_mesh)[0]

meshio.saveMesh("ip3r_mesh", tetmesh)

This will save the tetmesh object with compartment and patch definitions in shadow_mesh to the ip3r_mesh.xml and ip3r_mesh.txt files.

f2. (Without STEPS integration) type in

shadow_mesh.exportTo("mesh_conf")

This will export the mesh configuration in shadow_mesh to mesh_conf file.
The visualization of this example assumes that you follow f1.

Visualization:

In terminal change the working directory to the ip3r example folder and type in

python ip3r_sim.py

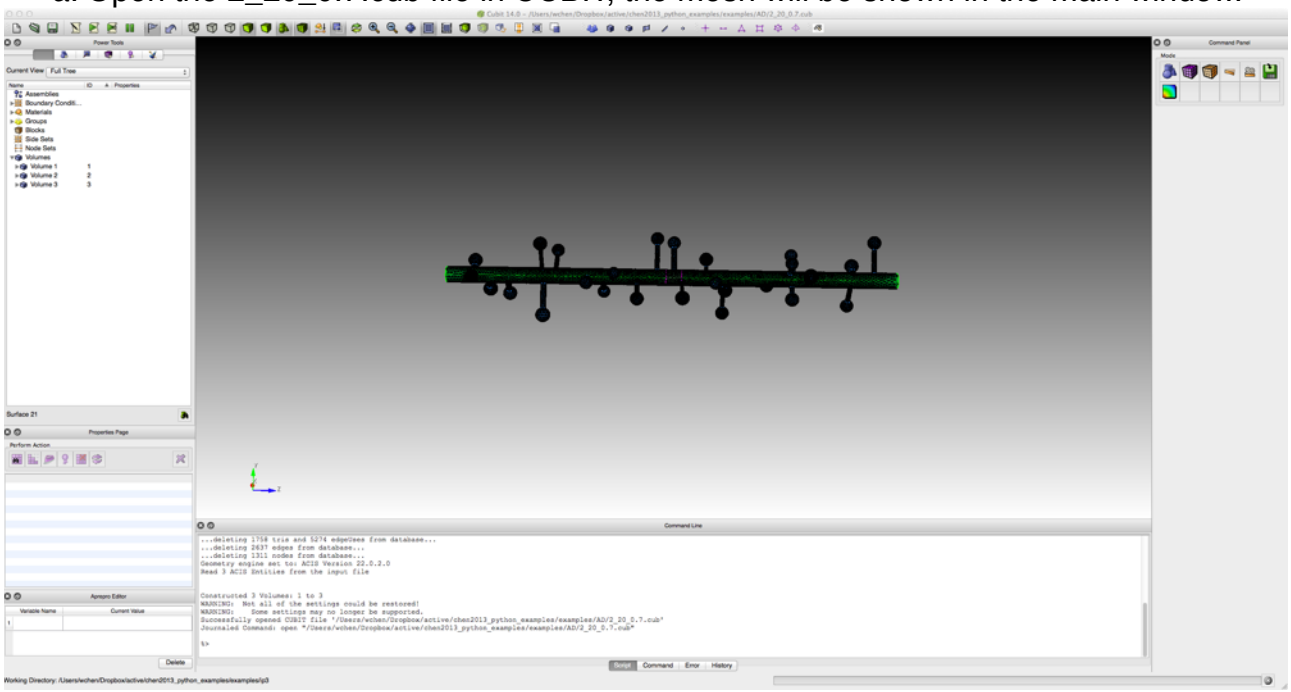
The displays and control will be displayed after loaded. Hit run/stop to run the simulation, or reset to reset the simulation. On the displays, use left mouse button to rotate the views, middle mouse button for panning, and wheel for rotation. Different species are explained in the manuscript.

The ip3r_model.py contains the ip3r model, while the ip3r_sim.py showcases the construction of components in the visualization toolkit.

2. Anomalous Diffusion Model
Files for the example are located in examples/AD.

Mesh Preparation:
As the mesh preparation process is the same for 4 meshes, we use 2_20_0.7.cub as example.

a. Open the 2_20_0.7.cub file in CUBIT, the mesh will be shown in the main window.



b. In the script panel, type in

import os
os.chdir(PATH_TO_THE_EXAMPLE_FOLDER)

To change the working path, then type in

import steps_cubit as scubit

This will import the Geometry Preparation toolkit as scubit, then type in

shadow_mesh = scubit.ShadowMesh()

To create a ShadowMesh object for the preparation.

c. Select Volume 1 from the left hand side panel, then in the script panel type in

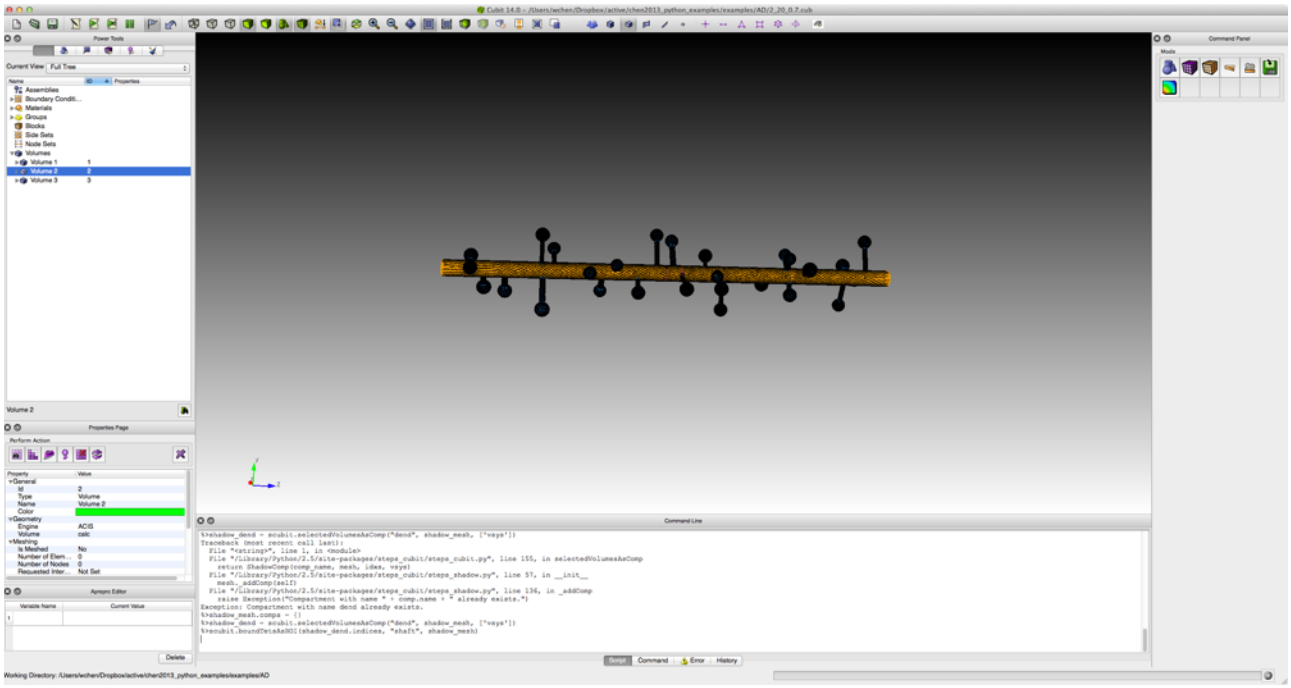
shadow_dend = scubit.selectedVolumesAsComp("dend", shadow_mesh, ['vsys'])

This will make Volume 1 as a compartment named “dend” and associates it with volume system “vsys”.

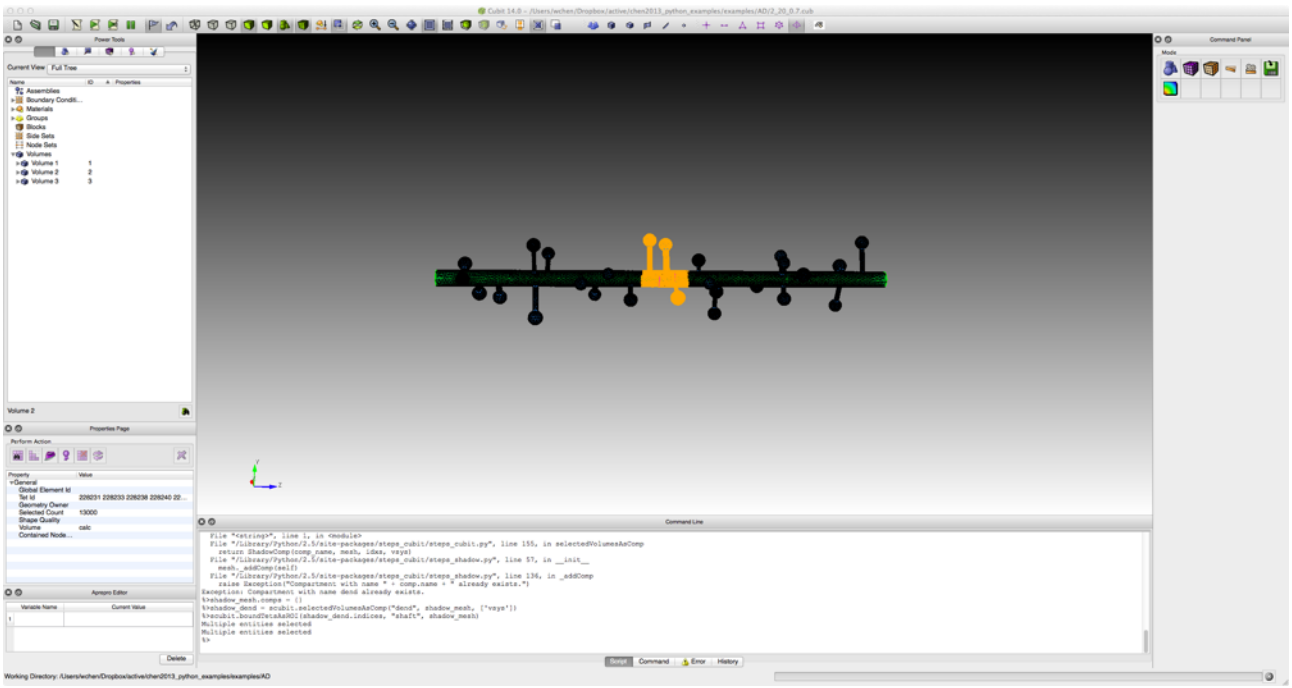
d. select Volume 2 and type in

scubit.boundTetsAsROI(shadow_dend.indices, "shaft", shadow_mesh)

This function assigns tetrahedrons in shadow_dend that are bound by Volume2 as the dendritic shaft. Note that the search may take a long time for mesh with high spine densities.



e. On the top panel enable x-ray and tet-selection panel options, cmd+left mouse button (key combination may change for different platform, check CUBIT's user manual for details) to select the tets around the center of the mesh, the selection should cover Volume 3 completely.



f. Keep the selection, in the script panel type in

reduced_list = scubit.getSelectedTets()

The purpose of the reduced_list is to reduce the search range for injection zone ROI.

g. Deselect the tets, select Volume 3 and type in

scubit.boundTetsAsROI(reduced_list, "injection", shadow_mesh)

This create another ROI as the injection zone using volume 3.

h. In terminal type in

cubit.cmd('export Abaqus "2_20_0.7.inp" dimension 3 overwrite cubitids ')

This exports the mesh as Abaqus inp file which can be imported in STEPS.

i.

i1. (With STEPS integration) type in

import steps.utilities.meshio as meshio

tetmesh = meshio.importAbaqus('2_20_0.7.inp', 1e-6, None, shadow_mesh)[0]

meshio.saveMesh("2_20_0.7", tetmesh)

This will save the tetmesh object with compartment and patch definitions in shadow_mesh to the 2_20_0.7.xml and 2_20_0.7_mesh.txt files.

i2. (Without STEPS integration) type in

shadow_mesh.exportTo("2_20_0.7_conf")

This will export the mesh configuration in shadow_mesh to 2_20_0.7_conf file.

The visualization of this example assumes that you follow i2.

Repeat these procedures for other meshes.

Visualization:

Two scripts are provided for the example. AD_single.py generates visualization for a single simulation (2_20_0.7, with/without filtering species in spines), and AD_all.py generates visualization for all 4 simulations. Note that due to the sizes of the meshes it may take some time to process the initialization. To execute the script, in terminal change the working directory to the example folder and type in

python AD_single.py or **python AD_all.py**