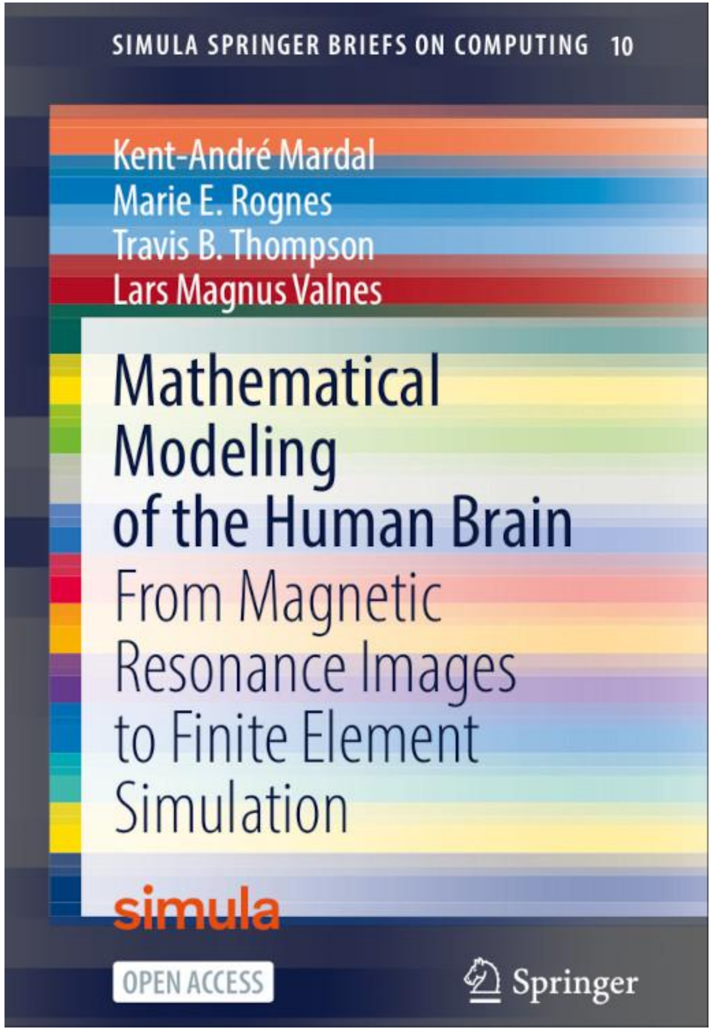
IGCT FEM Workshop
Meshing the human brain

Travis Thompson, Texas Tech University
March 5th 2023

Mathematical Modeling of the Human Brain



SIMULA SPRINGER BRIEFS ON COMPUTING 10

Kent-André Mardal
Marie E. Rognes
Travis B. Thompson
Lars Magnus Valnes

Mathematical Modeling of the Human Brain

From Magnetic Resonance Images to Finite Element Simulation

simula

OPEN ACCESS

Springer

Mathematical Modeling of the Human Brain

PDF version or order a hardcopy at
https://link.springer.com/book/10.1007/978-3-030-95136-8

Updateded versions can be obtained at
https://github.com/kent-and/mri2fem

The FEniCS numerical software can be downloaded at
https://fenicsproject.org/download/archive/

Everything starts with a DICOM file

- Install DicomBrowser (or equivalent)
  - Book pg. 14
- Open DicomBrowser

```
$ DicomBrowser &
```

- Extract T1, T2 and DTI sequences.
- The sequences are used as follows
  - T1: primary mesh generation
  - T2: (optional) improve segmentation
  - DTI: Extract a diffusion tensor

The MRI FEM book comes with a dataset that includes DICOM files with T1, T2 and DTI sequences included

https://zenodo.org/record/4899120

Segment a T1 sequence

- Use the FreeSurfer command recon-all to segment a T1 sequence

```
$ cd dicom-files/ernie/T13D
$ recon-all subjid ernie -i IM_0161 -all
```

The recon-all command can be augmented to use a T2 sequence to improve the segmentation if needed.

FreeSurfer recon-all documentation:

https://surfer.nmr.mgh.harvard.edu/fswiki/recon-all

Select a set of presegmented data

- For this tutorial, MRI data has been pre segmented for you using FreeSurfer

```
$ ls mri-data/data/freesurfer-files
abby ernie
```

- Abby is an anonymized dataset from a healthy female patient (late 30s at time of scan)
- Ernie is an anonymized dataset from a healthy male patient (mid 40s at time of scan)

```
$ ls mri-data/data/freesurfer-files/abby
label mri scripts stats surf tmp touch trash
```

- These directories are created by the FreeSurfer recon-all segmentation. They typically reside in your FreeSurfer/subjects directory
- We will primarily use the surf subdirectory, above
- Select the data set that you want to use. These slides will use ernie

An overview of the tutorial environment

- The tutorial takes place in a directory called runtime-env.
- This environment has been set up for you by the system administrators for this tutorial. The book tells you how to set up your own system.
- Let's look at the contents of runtime-env

```
$ ls
mri-data startfenics startfreesurfer startsvmtk
```

- mri-data: contains the dicom, FreeSurfer files
- startfenics: start an isolate FEniCS finite element solver environment
- startfreesurfer: start an isolated FreeSurfer environment
- startsvmtk: start an isolated instance of a linux operating system with the **S**urface **V**olume **M**apping **T**ool**k**it pre-installed.

https://github.com/SVMTK/SVMTK

Starting up the software environments

- Start the FEniCS software environment by

```
$ source startfenics

# FEniCS stable version image

Welcome to FEniCS/stable !

[ .. additional text .. ]

fenics@3b6caf02059f:~/shared$
```

- Exit the FEniCS environment by

```
fenics@3b6caf02059f:~/shared$ exit
exit

$
```

Working with shared folders

- After exiting the FEniCS software environment, let’s have a look at the directory structure of runtime-env

```
$ ls
fenics freesurfer svmtk mri-data startfenics
startfreesurfer startsvmtk
```

- The fenics directory is shared between the computer that you are working on and the isolated FEniCS application.
- Like a mailbox, any files that you copy into the fenics directory, from the outside, will be “sent” inside the isolated FEniCS application and vice-versa.
- The freesurfer and svmtk directories act similarly

A simple mesh, a simple model

A simple left hemisphere mesh

A simple isotropic diffusion model

Converting a surface to an STL

- When FreeSurfer performs its segmentation, it creates a large number of binary files that describe different brain surfaces.
- We will first convert one of these surfaces to a file format called STL or **S**tandard **T**riangle **L**anguage format.
- We will first **copy** a **left pial surface** to our FreeSurfer directory

Male brain

```
$ cp ../mri-data/data/freesurfer-files/ernie/surf/lh.pial ./freesurfer
```

Female brain

```
$ cp ../mri-data/data/freesurfer-files/abby/surf/lh.pial ./freesurfer
```

Converting a surface to an STL

- We are going to use FreeSurfer's **mris_convert** command to convert the FreeSurfer surface to an STL file.
- For this tutorial, we have to initialize the FreeSurfer environment manually. If you install FreeSurfer on your own machine, you could skip this initialization step.

```
$ source startfreesurfer
```

- Now lets verify that our lh.pial file is here

```
$ ls
lh.pial readme.txt
```

- Convert lh.pial to an STL and exit

```
$ mris_convert ./lh.pial ./pial.stl
Saving ./pial.stl as a surface
$ exit
```

Viewing the STL in ParaView

- ParaView can open **STL** files
- ParaView is not included in your demo software kit but is freely available from the following URL

https://www.paraview.org/download/

- Use the slice tool in Paraview on the STL file. What do you notice?
- The STL file only describes a surface. This is evident with visualization.
- We need a volume mesh to solve mathematical problems

Starting up the SVMTk

- The pial STL we created encodes information about the (left) pial surface. It is not yet a mesh.
- The SVMTk was designed to make brain meshes from STL files an **approachable** and **scriptable** process

Move pial.stl into the SVMTk directory

```
$ mv ./freesurfer/pial.stl ./svmtk
```

Start the SVMTk software and verify pial.stl is there

```
$ source startsvmtk
$ ls
pial.stl readme.txt
```

Start the vim text editor

```
$ vim makemesh.py
```

Your first SVMTk script: Make a mesh from the pial surface STL

Basic Vim commands

i	Enter write mode
:w	Save the file
:q	Exit the editor

Type the following Python code and save the script

```
import SVMTK as svmtk

# 1. load the STL file
surface = svmtk.Surface("pial.stl")

# 2. Generate the volume mesh from the surface
domain = svmtk.Domain(surface)
#     (specify the mesh resolution: 8, 16, 32, 64, etc)
domain.create_mesh(32)

# 3. Save the output
domain.save("lh.mesh")
```

Run your script and convert the result to XDMF format

Run the makemesh.py script and verify the output

```
$ python3 makemesh.py
Cell size 2.62271
Start meshing
(( The script will take about 90 seconds to finish ))
Number of isolated vertices removed: 0
Done meshing
$ ls
lh.mesh makemesh.py pial.stl readme.txt
```

- The FEniCS software can read meshes in a file format called XDMF

Convert the mesh to FEniCS XDMF using a script

```
$ cp ../SVMTk-tet-mesh-to-FEniCS-XDMF.py .
$ python3 SVMTk-tet-mesh-to-FEniCS-XDMF.py lh.mesh
Converting lh.mesh to FEniCS XDMF format
Writing FEniCS compatible mesh files: lh.xdmf and lh.h5
```

Viewing your mesh in ParaView

- ParaView can open **XDMF** files
- ParaView is not included in your demo software kit but is freely available from the following URL

https://www.paraview.org/download/

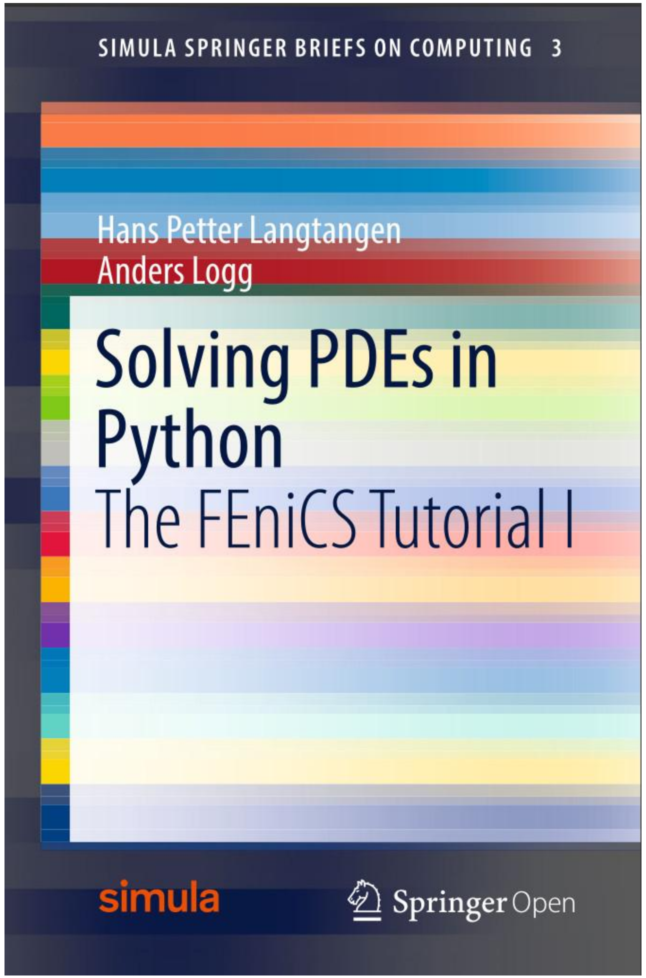
- Use the slice tool in Paraview on the **XDMF** file. What do you notice?
- The SVMTk has created a tetrahedral volume mesh from the STL file

Copy the mesh files and to the FEniCS shared directory

- Lets copy the mesh files from the shared directory of the SVMTk environment to the shared directory of the FEniCS environment

```
$ cp ./svmtk/lh.h5 ./fenics
$ cp ./svmtk/lh.xdmf ./fenics
```

Solving Partial Differential Equations with FEniCS


SIMULA SPRINGER BRIEFS ON COMPUTING 3

Hans Petter Langtangen
Anders Logg

Solving PDEs in Python
The FEniCS Tutorial I

simula
Springer Open

https://fenicsproject.org/tutorial/

An overly simple model problem of Gadobutrol penetration into the brain parenchyma

- Gadobutrol is continuously delivered by intrathecal injection over the course of 6 hours.
- After 6 hours, the concentration of CSF gadobutrol is kept constant
- (simplifying assumption) the extraventricular concentration of CSF gadobutrol is uniform
- (simplifying assumption) extracellular diffusion in the parenchyma is isotropic

A simple mathematical model of Gadobutrol delivery

$$\begin{aligned} u_t - \kappa \Delta u &= f && \text{in } \Omega \times (0,T] \\ u &= u_D && \text{on } \Omega \times (0,T] \\ u(0,x) &= u_0(x) && \text{in } \Omega \end{aligned}$$

$$u_0(x) = 0 \quad u_D(t,x) = \begin{cases} (tc/3600) & 0 \leq t \leq 3600 \\ c & t > 3600 \end{cases}$$

$$\kappa = 7.2 \times 10^{-3} \, \frac{\text{mm}^2}{\text{min}} \qquad c = 2.813 \times 10^{-3} \, \frac{\mu\text{mol}}{\text{mm}^3}$$

Setting up to solve the model with FEniCS

Start the FEniCS environment

```
$ source startfenics
```

- FEniCS translates partial differential equations expressed in a ***variational formulation*** to sequence of linear problems (problems of the form **Ax** = **b**).
- This process has a slight learning curve. We have included the code needed to solve the mathematical model (from the previous slide) in ***diffusion.py***.

```
$ ls
diffusion.py lh.h5 lh.xdmf pre-made results
```

- The ***diffusion.py*** script will be looking for an XDMF file named *ernie.xdmf* so lets rename our file

```
$ mv lh.xdmf ernie.xdmf
$ ls
diffusion.py ernie.xdmf lh.h5 pre-made results
```

Solving the mathematical model of Gadobutrol delivery with FEniCS

Run ***diffusion.py*** and solve the model with FEniCS

```
$ python3 diffusion.py

Calling FFC just-in-time (JIT) compiler, this may take some time.
Storing at n = 10 (of 1440), t = 30 (min)
Storing at n = 20 (of 1440), t = 60 (min)
Storing at n = 30 (of 1440), t = 90 (min)
Storing at n = 40 (of 1440), t = 120 (min)
Storing at n = 50 (of 1440), t = 150 (min)
...
Storing at n = 1400 (of 1440), t = 4200 (min)
Storing at n = 1410 (of 1440), t = 4230 (min)
Storing at n = 1420 (of 1440), t = 4260 (min)
Storing at n = 1430 (of 1440), t = 4290 (min)
Storing at n = 1440 (of 1440), t = 4320 (min)

$ ls ./results/
amounts.csv  u000002.vtu  u000011.vtu  u000020.vtu
u000029.vtu  u000038.vtu  ...

$ exit
```

View simulation results in ParaView

- ParaView can open **pvd, pvu** and **vtk** FEniCS solution output files
- ParaView is not included in your demo software kit but is freely available from the following URL

https://www.paraview.org/download/

- Use the 2D or 3D slice tool in Paraview on the solution sequence to see the simulated gadobutrol penetration into the brain parenchyma

Intermediate meshing

A two-domain, left-hemisphere mesh

Working with the ventricles

A whole brain mesh

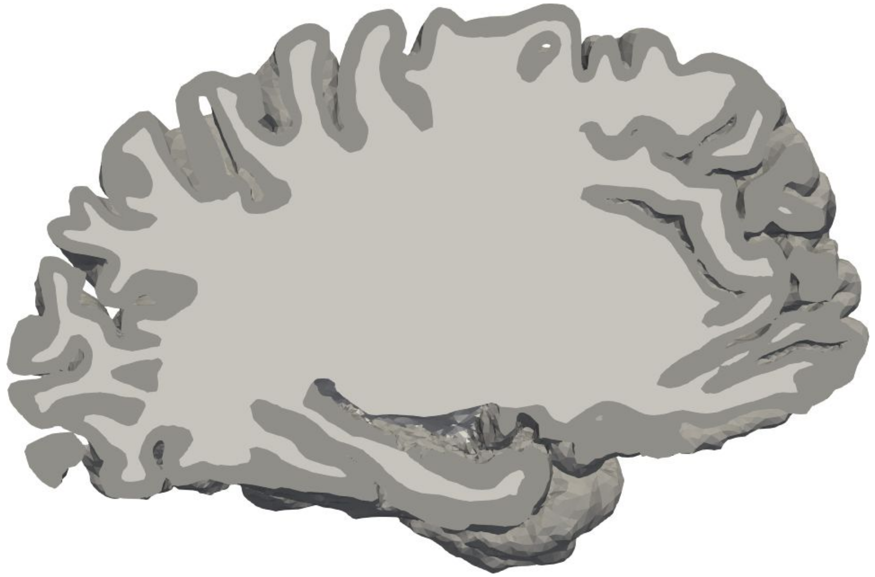
Advanced topics covered in the book

- Tagging regions from a full parcellation
- Adding anisotropic diffusion

A two-domain left hemisphere mesh

This part of the tutorial demonstrates

- The basic use of the SVMTk SubdomainMap class to create a mesh with distinct subdomains and corresponding subdomain tags



Create surface files for the left hemisphere gray and white matter

- In the previous example, we meshed the pial surface and created a volume mesh.
- The resulting volume mesh was unable to differentiate between **gray** and **white** matter
- Gray and white matter have different material properties important to mathematical modeling

Copy the left pial and white matter surfaces to FreeSurfer

```
$ cp mri-data/data/freesurfer-files/ernie/surf/lh.pial ./freesurfer
$ cp mri-data/data/freesurfer-files/ernie/surf/lh.white ./freesurfer
```

- You can copy from either the **ernie** (male patient) or **abby** (female patient) in the commands above

Start FreeSurfer and create STLs with mris_convert

```
$ source startfreesurfer
$ mris_convert lh.pial lh.pial.stl
$ mris_convert lh.white lh.white.stl
$ exit
```

Combining domains using SVMTk Subdomain Maps

Copy the STL files from FreeSurfer and start the SVMTk

```
$ mv ./freesurfer/lh.pial.stl ./svmtk
$ mv ./freesurfer/lh.white.stl ./svmtk
$ source startsvmtk
```

Open vim and create two-domain.py

```
import SVMTK as svmtk

pial = svmtk.Surface("lh.pial.stl")
white = svmtk.Surface("lh.white.stl")

# an ordered list of surfaces
surfaces = [pial,white]
smap = svmtk.SubdomainMap()

# Relative position and numeric IDs create surface “tags”
smap.add(“10”, 1)
smap.add(“11”, 2)

# create a domain from the surface list and the surface (tag) map
domain = svmtk.Domain(surfaces, smap)
domain.create_mesh(32)
domain.save("ernie-gw.mesh")
```

Running and a quick view of the two-domain map

Run the two-domain script to create the tagged mesh

```
$ python3 two-domain.py
Cell size: 2.8096
Start meshing
Number of isolated vertices removed: 0
Done meshing
```

Convert the result using meshio for easy viewing in ParaView

```
$ meshio convert ernie-gw.mesh ernie-gw.xdmf
```

View the result in ParaView

https://www.paraview.org/download/

Additional post processing for use with FEniCS

- Further post processing is needed to use the mesh file in FEniCS
- Supplementary source code is available for use alongside the book

https://github.com/kent-and/mri2fem

To convert an SVMTk mesh, complete with subdomain tags, to a set of XDMF files for use with FEniCS, see:

https://github.com/kent-and/mri2fem/blob/master/mri2fem/mri2fem/chp4/convert_to_dolfin_mesh.py

The Ventricular region

- Thus far, the ventricles have not been included in our left hemisphere model
- A model that incorporates the ventricles as a fluid domain may be important for considering drug delivery applications.
- If the ventricles are not needed, we can reduce the complexity of the computations by removing them from the mesh.
- Freesurfer includes the ***mri_binarize*** command to extract specific surfaces from parcellated volumes

Copy the white matter volume to the FreeSurfer environment

```
$ cp mri-data/data/freesurfer-files/ernie/mri/wmparc.mgz ./freesurfer
```

- You can choose **ernie** (male) or **abby** (female)

Start FreeSurfer

```
$ source startfreesurfer
```

The mri_binarize FreeSurfer command

- FreeSurfer's ***mri_binarize*** command is a tool that marks and extracts voxels from a volume (.mgz) file
- Depending on the volume file, the voxels contained within it may be marked with various types of data. Important examples include **signal intensity** and **segmentation tags**.
- The file **wmparc.mgz** is a parcellation volume. That means that the voxels of wmparc.mgz are labelled with integer values corresponding to a specific location code
- FreeSurfer comes with a utility called ***freeview*** that can be used to visualize, or correct, the volume and surface files created by FreeSurfer's recon-all segmentation.

FreeSurfer's mri_binarize has over 40 options. We will use:

```
$ mri_binarize –i [input file] –ventricles –match [value]
–surf-smooth [value] –surf [output file]
```

Extracting the ventricles from wmparc.mgz

In the FreeSurfer environment, enter the command

```
$ mri_binarize –i wmparc.mgz –ventricles –match 15 –surf
ventricles-nosmooth.stl
input      wmparc.mgz
frame      0
nErode3d   0
nErode2d   0
…
Found 16080 voxels in final mask
MRIStessellate: nvertices = 16469, nfaces = 16562
Count: 16080 16080.000000 16777216 0.095844
```

Enter the same command but add 5 smoothing iterations

```
$ mri_binarize –i wmparc.mgz –ventricles –match 15 –surf-smooth
5 –surf ventricles-smooth.stl
```

Inspecting the extracted ventricular surfaces

- The extracted STL files can be viewed in ParaView

View the result in ParaView

https://www.paraview.org/download/

- Both the raw binary ventricular volume and the smoothed volume show disconnected regions.
- The segmentation has resolution issues
  - The cerebral aqueduct is not well defined
  - The 4th ventricle is disconnected from the 3rd and lateral ventricles
- Segmentations can be improved by hand using FreeSurfer’s ***freeview*** tool

FreeSurfer postprocessing tools for surfaces

- FreeSurfer comes with a number of tools that can be used to postprocess surface files. We will use two.
- ***mri_volcluster*** : identifies clusters in a volume and allows for the elimination of clusters with volumes below a certain threshold. We will use mri_volcluster to get rid of the detached cerebral aqueduct and 4th ventricle
- **mri_morphology** : performs various operations on volumes such as opening, closing, dilating, eroding and filling holes
- The average adult CSF volume is approx 150 cubed mm. We propose the following postprocessing

Ventricle postprocessing for use with meshing

1. Binarize the lateral and third ventricles to a volume
2. Cluster with a minimum size of 100 cubed mm. Voxels in the largest cluster will be assigned a value of 1.
3. Binarize again to select the largest cluster
4. Close holes in the ventricular volume
5. Export the resulting ventricular surface

Postprocess the ventricles

Extract the lateral and 3rd ventricles to a temporary volume

```
$ mri_binarize –i wmparc.mgz –ventricles –o “tmp.mgz”
```

Cluster the volume, set a 100mm cubed threshold

```
$mri_volcluster –in “tmp.mgz” –thmin 1 –minsize 100 –ocn
“tmp-ocn.mgz”
```

Extract the largest cluster (labeled with a 1)

```
$ mri_binarize –i “tmp-ocn.mgz” –match 1 –o “tmp.mgz”
```

Run two iterations of gap closing

```
$ mri_morphology “tmp.mgz” close 2 “tmp.mgz”
```

Extract the final surface, 3 smoothing iterations

```
$ mri_binarize –i “tmp.mgz” –match 1 –surf-smooth 3
–surf ventricles-final.stl
```

Working with the ventricles and the brain mesh

- Copy **lh.ventricles-final.stl** to the SVMTk shared directory and check that **lh.pial.stl** and **lh.white.stl** are still there as well

Start the SVMTk

```
$ source startsvmtk
```

- Using Vim, create “**ventricle-mesh.py**” with

```
$ vim ventricle-mesh.py
```

- The code for **ventricles-mesh.py** is on the next slide

Working with the ventricles and the brain mesh

ventricle-mesh.py

```
import SVMTK as svmtk

#----
pial   = svmtk.Surface("lh.pial.stl")
white= svmtk.Surface("lh.white.stl")
ventricles=svmtk.Surface("lh.ventricles-final.stl”)
surfaces = [pial,white,ventricles]

#----
smap   = svmtk.SubdomainMap()
smap.add(“100”,1) #tag the pial surface
smap.add(“110”,2) #tag the white surface
smap.add(“111”,3) #tag the ventricle surface
domain = svmtk.Domain(surfaces, smap)

#----
domain.create_mesh(32)
domain.save("lh.ventricle.mesh")
```

A quick view of the tagged domain in paraview

- We can quickly view the result using Paraview

```
$ meshio convert lh.ventricle.mesh lh.ventricle.xdmf
```

View the result in ParaView

https://www.paraview.org/download/

- Further postprocessing is required to use this mesh, and its subdomain tags, with FEniCS for mathematical modeling.

A slight modification allows ventricle removal

ventricle-removed-mesh.py

```
import SVMTK as svmtk

#----
pial   = svmtk.Surface("lh.pial.stl")
white= svmtk.Surface("lh.white.stl")
ventricles=svmtk.Surface("lh.ventricles-final.stl”)
surfaces = [pial,white,ventricles]

#----
smap   = svmtk.SubdomainMap()
smap.add(“100”,1) #tag the pial surface
smap.add(“110”,2) #tag the white surface
smap.add(“111”,3) #tag the ventricle surface
domain = svmtk.Domain(surfaces, smap)

#----
domain.create_mesh(32)
domain.remove_subdomain(3) # Remove the ventricles
domain.save("lh.no-ventricle.mesh")
```

A quick view of the no-ventricle domain in paraview

- We can quickly view the result using Paraview

```
$ meshio convert lh.no-ventricle.mesh lh.no-ventricle.xdmf
```

View the result in ParaView

https://www.paraview.org/download/

- Further postprocessing is required to use this mesh, and its subdomain tags, with FEniCS for mathematical modeling.

A full brain mesh

- So far we have seen simple examples of the following:
  - How to convert surface files to STL files
  - How a surface can be used to construct a volume mesh using the SVMTk
  - How surfaces can be combined using SVMTk subdomains
  - How surfaces can be tagged and used to exclude parts of a volume mesh
- We have also seen that post processing can help turn .mesh files, produced by the SVMTk, into .xdmf files that can be used by FEniCS

As a final exercise, we put all of the above simple steps together to make a mesh of the full brain from the gray and white matter and the ventricles.

- You will need to convert the following FreeSurfer surfaces to STL files
  - lh.pial, lh.white
  - rh.pial, rh.white
  - You will need the ventricle STL from the previous step

A full brain mesh

full-brain-ventricle-mesh.py

```
import SVMTK as svmtk

#----
stlfiles = ("lh.pial.stl", "rh.pial.stl", "lh.white.stl", "rh.white.stl",
“lh.ventricles-final.stl”)
surfaces = [svmtk.Surface(stl) for stl in stls]

#---Combine the white matter surfaces---
surfaces[2].union(surfaces[3])
#---Remove the unnecessary rh.white.stl---
surfaces.pop(3)

#--Build the surface map--
smap   = svmtk.SubdomainMap()
smap.add(“1000”,1) #left pial surface
smap.add(“0100”,1) #right pial surface
smap.add(“1010”,2) #white matter on left side
smap.add(“0110”,2) #white matter on right side
smap.add(“1110”,2) #white matter on both sides
smap.add(“1011”,3) #ventricles on left side
smap.add(“0111”,3) #ventricles on right side
smap.add(“1111”,3) #ventricles on both sides

domain = svmtk.Domain(surfaces, smap)

#----
domain.create_mesh(32)
domain.save("fullbrain-ventricle.mesh")
```

A quick view of the full brain domain in paraview

```
$ meshio convert fullbrain-ventricle.mesh fullbrain-ventricle.xdmf
```

- We can quickly view the result using Paraview

View the result in ParaView

https://www.paraview.org/download/

- Further postprocessing is required to use this mesh, and its subdomain tags, with FEniCS for mathematical modeling.

A full brain mesh without the ventricles

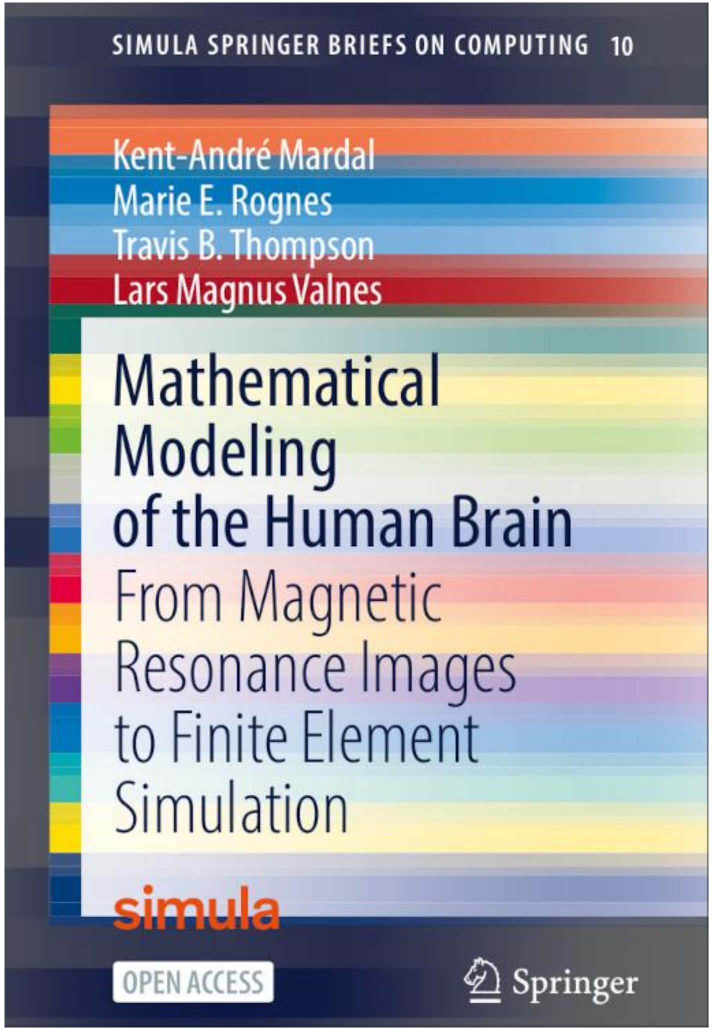
full-brain-no-ventricle-mesh.py

```
import SVMTK as svmtk

#----
stlfiles = ("lh.pial.stl", "rh.pial.stl", "lh.white.stl", "rh.white.stl",
“lh.ventricles-final.stl”)
surfaces = [svmtk.Surface(stl) for stl in stls]

#---Combine the white matter surfaces---
surfaces[2].union(surfaces[3])
#---Remove the unnecessary rh.white.stl---
surfaces.pop(3)

#--Build the surface map--
smap   = svmtk.SubdomainMap()
smap.add(“1000”,1) #left pial surface
smap.add(“0100”,1) #right pial surface
smap.add(“1010”,2) #white matter on left side
smap.add(“0110”,2) #white matter on right side
smap.add(“1110”,2) #white matter on both sides
smap.add(“1011”,3) #ventricles on left side
smap.add(“0111”,3) #ventricles on right side
smap.add(“1111”,3) #ventricles on both sides

domain = svmtk.Domain(surfaces, smap)

#----
domain.create_mesh(32)
domain.remove_subdomain(3) # Remove the ventricles
domain.save("fullbrain-no-ventricle.mesh")
```

Many more topics

- Additional mesh operations
  - Unions of surfaces
  - Fixing overlapping surfaces
  - Separating close surfaces
  - Global mesh refinement
  - Mesh refinement by region / parcellation label
  - Mesh smoothing
- Adding detailed parcellation data to the mesh
  - Add mesh labels to identify cortical regions such as the entorhinal cortex, the parahippocampal gyrus, etc
  - Add mesh labels to Identify subcortical regions such as the hippocampus, the amygdala, etc.
- Building anisotropic diffusion tensors from DTI images
- Defining and running more complex, larger scale mathematical models in FEniCS using these assets

Mathematical Modeling of the Human Brain



SIMULA SPRINGER BRIEFS ON COMPUTING 10

Kent-André Mardal
Marie E. Rognes
Travis B. Thompson
Lars Magnus Valnes

Mathematical Modeling of the Human Brain

From Magnetic Resonance Images to Finite Element Simulation

simula

OPEN ACCESS

Springer

Mathematical Modeling of the Human Brain

PDF version or order a hardcopy at
https://link.springer.com/book/10.1007/978-3-030-95136-8

Updateded versions can be obtained at
https://github.com/kent-and/mri2fem

The FEniCS numerical software can be downloaded at
https://fenicsproject.org/download/archive/

Thank you

Special thanks to

Dr. David Fuentes

Dr. Kari Brewer Savannah

SVMTk and book coauthors

Kent-Andre Mardal

Marie E. Rognes

Lars-Magnes Valnes