ldrb Documentation

Release 0.1

Henrik Finsberg

Dec 15, 2021

Demos

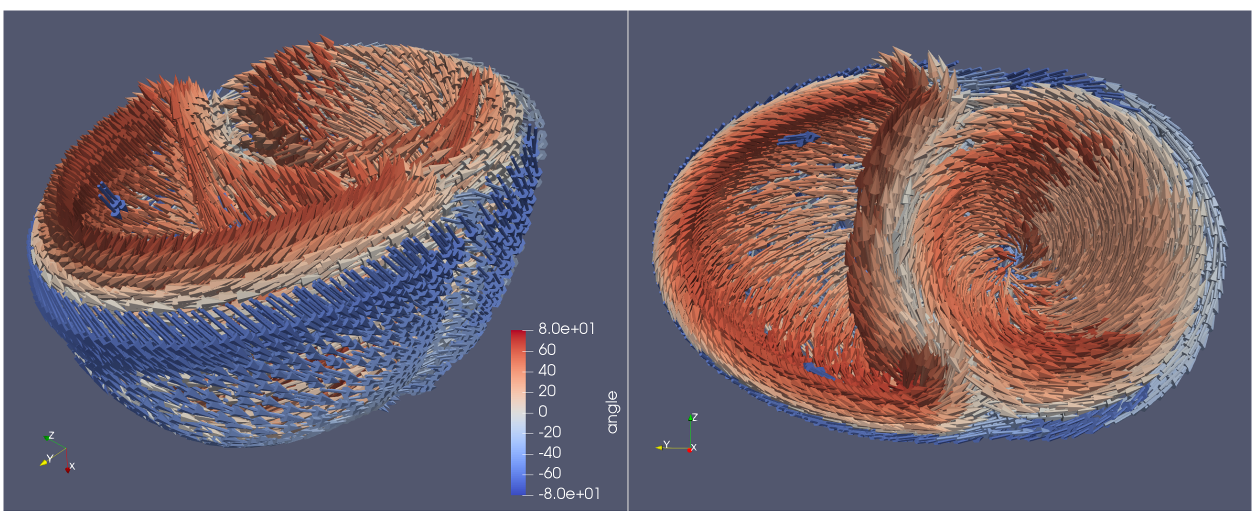
A software for assigning myocardial fiber orientations based on the Laplace Dirichlet Ruled-Based algorithm.

> Bayer, J.D., Blake, R.C., Plank, G. and Trayanova, N.A., 2012. A novel rule-based algorithm for assigning myocardial fiber orientation to computational heart models. Annals of biomedical engineering, 40(10), pp.2243-2254.(https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3518842/)

```
# Decide on the angles you want to use
angles = dict(alpha_endo_lv=30,     # Fiber angle on the LV endocardium
              alpha_epi_lv=-30,     # Fiber angle on the LV epicardium
              beta_endo_lv=0,       # Sheet angle on the LV endocardium
              beta_epi_lv=0,        # Sheet angle on the LV epicardium
              alpha_endo_sept=60,   # Fiber angle on the Septum endocardium
              alpha_epi_sept=-60,   # Fiber angle on the Septum epicardium
              beta_endo_sept=0,    # Sheet angle on the Septum endocardium
              beta_epi_sept=0,    # Sheet angle on the Septum epicardium
              alpha_endo_rv=80,     # Fiber angle on the RV endocardium
              alpha_epi_rv=-80,     # Fiber angle on the RV epicardium
              beta_endo_rv=0,       # Sheet angle on the RV endocardium
              beta_epi_rv=0)         # Sheet angle on the RV epicardium

# Choose space for the fiber fields
# This is a string on the form {family}_{degree}
fiber_space = 'Quadrature_2'

# Compute the microstructure
fiber, sheet, sheet_normal = ldrb.dolfin_ldrb(mesh=mesh,
                                              fiber_space=fiber_space,
                                              ffun=ffun,
                                              markers=markers,
                                              **angles)
# Store files using a built in xdmf viewer that also works for functions
# defined in quadrature spaces
ldrb.fiber_to_xdmf(fiber, 'fiber')
# And visualize it in Paraview
```



CHAPTER 1

Installation

1.1 Install with pip

In order to install the software you need to have installed FEniCS (versions older than 2016 are not supprted)

The package can be installed with pip.

```
python -m pip install ldrb
```

or if you need the most recent version you can install the source

```
python -m pip install git+https://github.com/finsberg/ldrb.git
```

1.2 Install with conda

Alternatively you can install with conda

```
conda install -c finsberg ldrb
```

CHAPTER 2

Documetation

Documentation is hosted at https://ldrb.readthedocs.io.

CHAPTER 3

Getting started

Check out the demos and the documentation

CHAPTER 4

Known issues

If you encounter the following error:

```
ImportError: numpy.core.multiarray failed to import
```

see https://github.com/moble/quaternion/issues/72 for how to troubleshoot.

CHAPTER 5

License

`ldrb` is licensed under the GNU LGPL, version 3 or (at your option) any later version. `ldrb` is Copyright (2011-2019) by the authors and Simula Research Laboratory.

CHAPTER 6

Contributors

Henrik Finsberg (henriknf@simula.no)

6.1 Demos

6.1.1 LV Geometry

In this demo we will show how you can generate fibers using the ldrb algorithm on a LV mesh. In order to run this demo you also need to install mshr if you haven't already.

To run the demo in serial do

```
python demo_lv.py
```

If you want to run the demo in parallel you should first comment out the lines that don't work in serial. Say you want to run on 4 cpu's, you run the command:

```
mpirun -n 4 python demo_lv.py
```

```
import dolfin as df
import ldrb

# Here we just create a lv mesh. Here you can use yor own mesh instead.
geometry = ldrb.create_lv_mesh()

# The mesh
mesh = geometry.mesh
# The facet function (function with marking for the boundaries)
ffun = geometry.ffun
# A dictionary with keys and values for the markers
markers = geometry.markers
```

(continues on next page)

(continued from previous page)

```
# Also if you want to to this demo in parallell you should create the mesh
# in serial and save it to e.g xml
# df.File('lv_mesh.xml') << mesh

# And when you run the code in paralall you should load the mesh from the file.
# mesh = df.Mesh('lv_mesh.xml')

# Since the markers are the default markers and the facet function is
# stored within the mesh itself, you can just set
# markers = None
# ffun = None


# Decide on the angles you want to use
angles = dict(alpha_endo_lv=60,  # Fiber angle on the endocardium
              alpha_epi_lv=-60,  # Fiber angle on the epicardium
              beta_endo_lv=0,    # Sheet angle on the endocardium
              beta_epi_lv=0)     # Sheet angle on the epicardium

# Choose space for the fiber fields
# This is a string on the form {family}_{degree}
fiber_space = 'Quadrature_2'
# fiber_space = 'Lagrange_1'

# Compute the microstructure
fiber, sheet, sheet_normal = ldrb.dolfin_ldrb(mesh=mesh,
                                              fiber_space=fiber_space,
                                              ffun=ffun,
                                              markers=markers,
                                              **angles)

# Store the results
df.File('lv_fiber.xml') << fiber
df.File('lv_sheet.xml') << sheet
df.File('lv_sheet_normal.xml') << sheet_normal

# If you run in parallel you should skip the visualisation step and do that in
# serial in stead. In that case you can read the the functions from the xml
# Using the following code
# V = ldrb.space_from_string(fiber_space, mesh, dim=3)
# fiber = df.Function(V, 'lv_fiber.xml')
# sheet = df.Function(V, 'lv_sheet.xml')
# sheet_normal = df.Function(V, 'lv_sheet_normal.xml')

# Store files in XDMF to be visualised in Paraview
# (These function are not tested in parallel)
ldrb.fiber_to_xdmf(fiber, 'lv_fiber')
ldrb.fiber_to_xdmf(sheet, 'lv_sheet')
ldrb.fiber_to_xdmf(sheet_normal, 'lv_sheet_normal')
```

6.1.2 BiV Geometry

In this demo we will show how you can generate fibers using the ldrb algorithm on a BiV mesh. In order to run this demo you also need to install mshr if you haven't already.

To run the demo in serial do

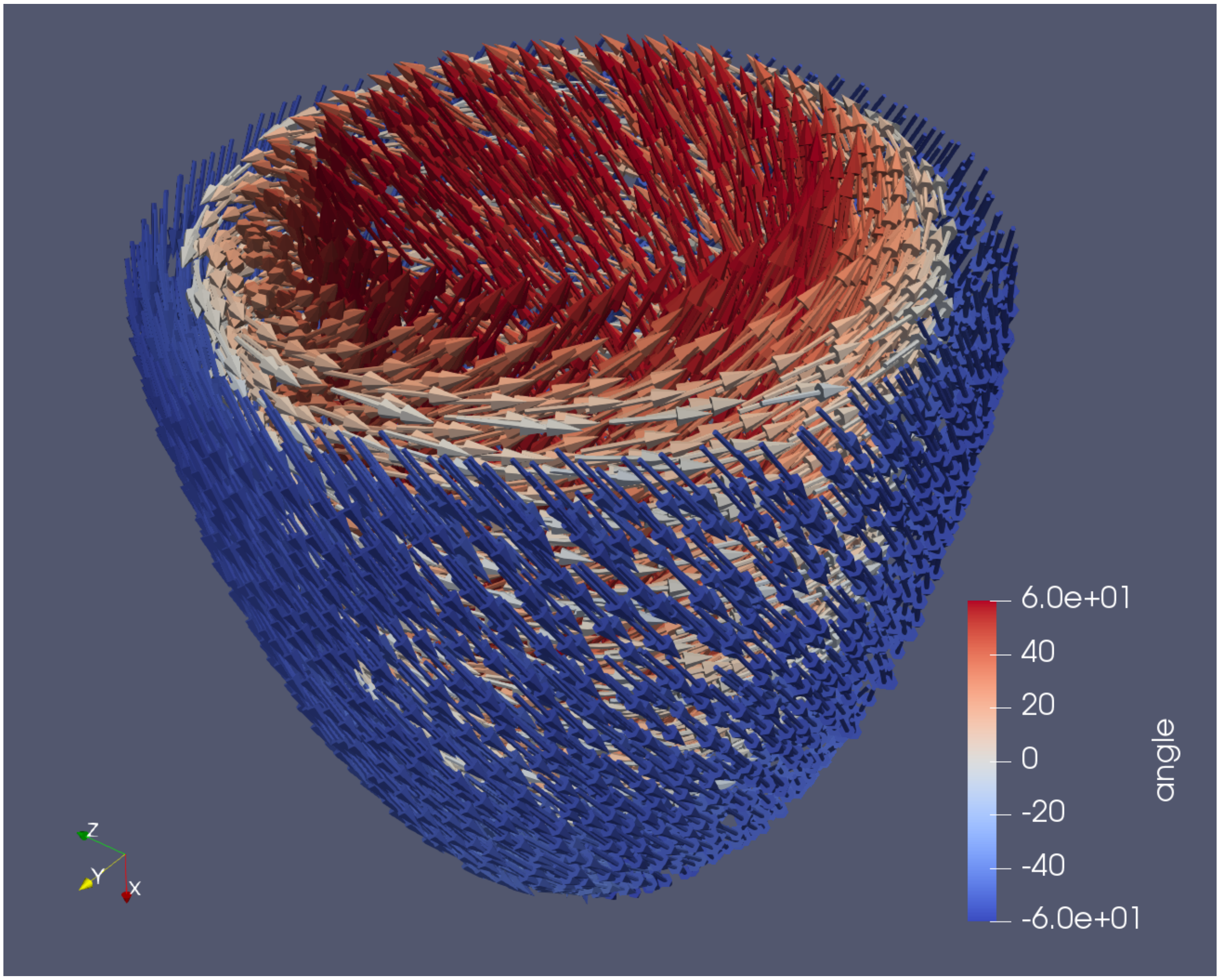


Fig. 1: LV Fiber

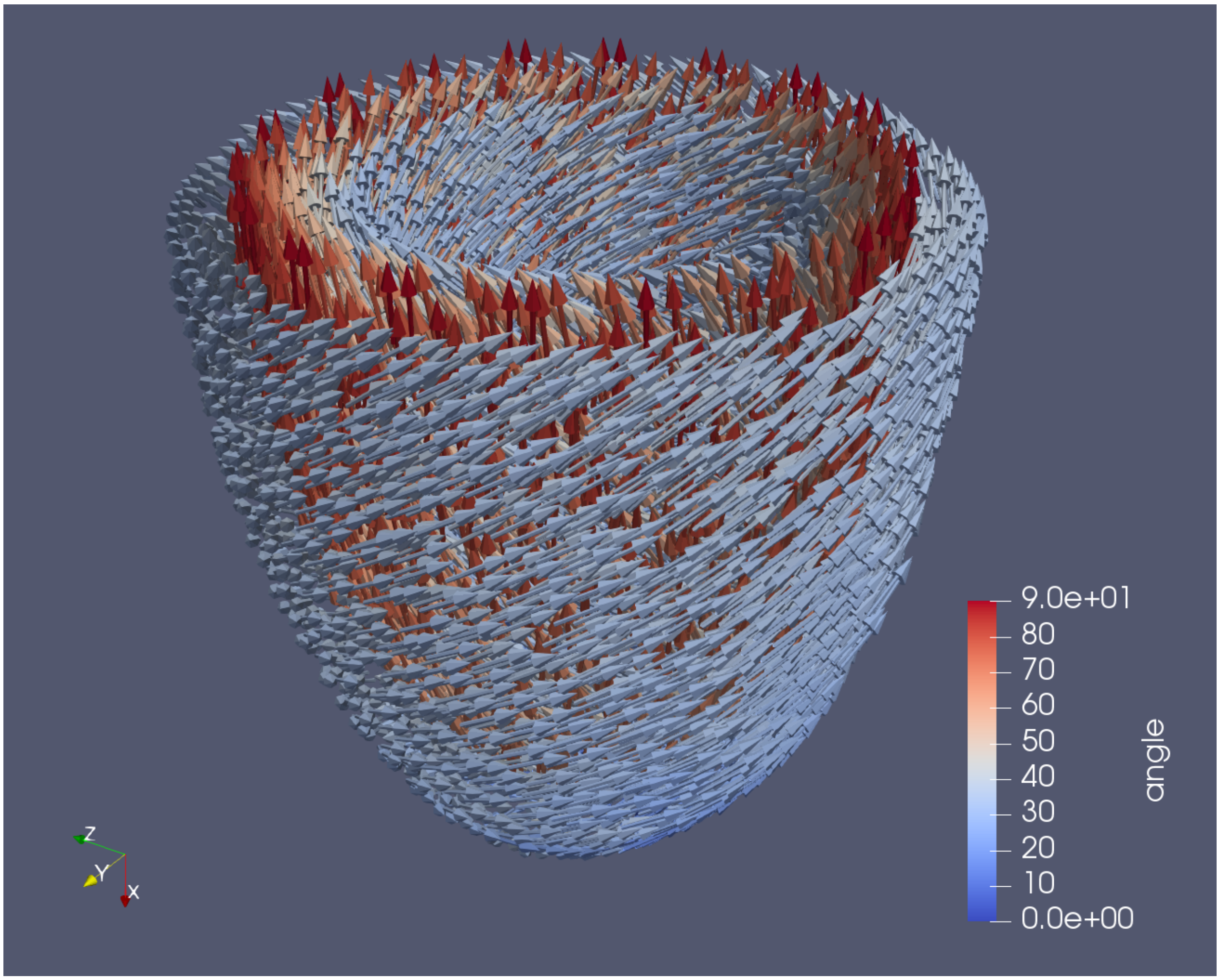


Fig. 2: LV Sheet

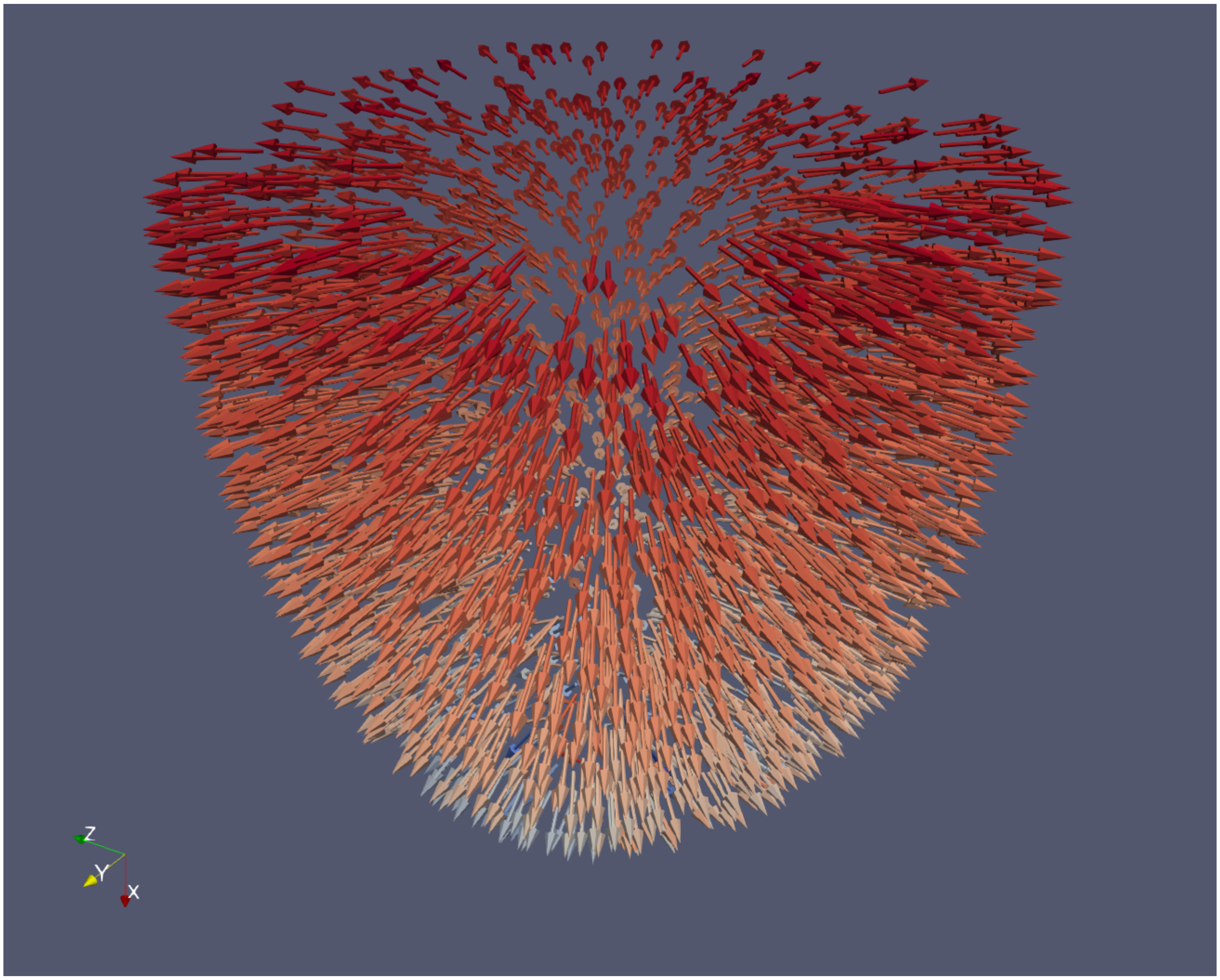


Fig. 3: LV Sheet-normal

```
python demo_biv.py
```

If you want to run the demo in parallel you should first comment out the lines that don't work in serial. Say you want to run on 4 cpu's, you run the command:

```
mpirun -n 4 python demo_biv.py
```

```
import dolfin as df
import ldrb

# Here we just create a lv mesh. Here you can use yor own mesh instead.
geometry = ldrb.create_biv_mesh()
#
# The mesh
mesh = geometry.mesh
# The facet function (function with marking for the boundaries)
ffun = geometry.ffun
# A dictionary with keys and values for the markers
markers = geometry.markers

# Also if you want to to this demo in parallell you should create the mesh
# in serial and save it to e.g xml
# df.File('biv_mesh.xml') << mesh


# And when you run the code in paralall you should load the mesh from the file.
# mesh = df.Mesh('biv_mesh.xml')

# Since the markers are the default markers and the facet function is
# stored within the mesh itself, you can just set
# markers = None
# ffun = None


# Decide on the angles you want to use
angles = dict(alpha_endo_lv=30,     # Fiber angle on the LV endocardium
              alpha_epi_lv=-30,     # Fiber angle on the LV epicardium
              beta_endo_lv=0,       # Sheet angle on the LV endocardium
              beta_epi_lv=0,        # Sheet angle on the LV epicardium
              alpha_endo_sept=60,   # Fiber angle on the Septum endocardium
              alpha_epi_sept=-60,   # Fiber angle on the Septum epicardium
              beta_endo_sept=0,   # Sheet angle on the Septum endocardium
              beta_epi_sept=0,    # Sheet angle on the Septum epicardium
              alpha_endo_rv=80,     # Fiber angle on the RV endocardium
              alpha_epi_rv=-80,     # Fiber angle on the RV epicardium
              beta_endo_rv=0,       # Sheet angle on the RV endocardium
              beta_epi_rv=0)         # Sheet angle on the RV epicardium

# Choose space for the fiber fields.
# This is a string on the form {family}_{degree}
fiber_space = 'Quadrature_2'
# fiber_space = 'Lagrange_1'

# Compute the microstructure
fiber, sheet, sheet_normal = ldrb.dolfin_ldrb(mesh=mesh,
                                              fiber_space=fiber_space,
                                              ffun=ffun,
```

(continues on next page)

(continued from previous page)

```
                                                             markers=markers,
                                                             log_level=df.debug,
                                                             **angles)

# # Store the results
df.File('biv_fiber.xml') << fiber
df.File('biv_sheet.xml') << sheet
df.File('biv_sheet_normal.xml') << sheet_normal

# If you run in parallel you should skip the visualization step and do that in
# serial in stead. In that case you can read the the functions from the xml
# Using the following code
# V = ldrb.space_from_string(fiber_space, mesh, dim=3)
# fiber = df.Function(V, 'biv_fiber.xml')
# sheet = df.Function(V, 'biv_sheet.xml')
# sheet_normal = df.Function(V, 'biv_sheet_normal.xml')

# Store files in XDMF to be visualized in Paraview
# (These function are not tested in parallel)
ldrb.fiber_to_xdmf(fiber, 'biv_fiber')
ldrb.fiber_to_xdmf(sheet, 'biv_sheet')
ldrb.fiber_to_xdmf(sheet_normal, 'biv_sheet_normal')
```

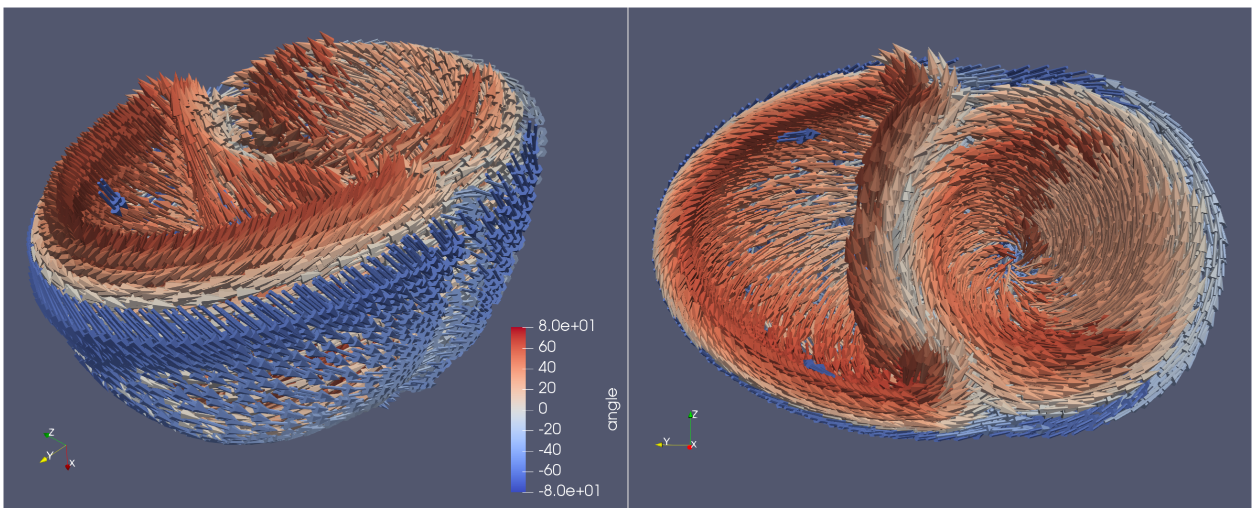


Fig. 4: BiV Fiber

6.2 ldrb package

6.2.1 Submodules

6.2.2 ldrb.ldrb module

6.2.3 ldrb.save module

6.2.4 ldrb.utils module

6.2.5 Module contents

6.3 Source code

Source code is avaible at GitHub https://github.com/finsberg/ldrb

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

7.1 Source code

Source code is avaible at GitHub https://github.com/finsberg/ldrb

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

8.1 Source code

Source code is avaible at GitHub https://github.com/finsberg/ldrb

CHAPTER 9

Indices and tables

- genindex
- modindex
- search