SimVascular Lumped Parameter Modeller

Krishna Patel and Senkai Hsia

Under the Direction of

Dr Hongzhi Lan
Justin Tran
Gabriel Maher

Professor Alison L. Marsden

Cardiovascular Biomechanics Computation Laboratory
Stanford University

August 2017

Abstract

SimVascular is an open source software package that enables a complete pipeline from patient medical image data to a computational fluid dynamics (CFD) simulation of patient blood flow. The CFD boundary conditions that are required to allow SimVascular to achieve accurate results utilise lumped parameter circuit models (LPM); where electrical circuit components are used as analogies for the cardiovascular system. Under the status quo, users of SimVascular will need to manually generate the source code to solve the associated ordinary differential equations (ODE) for each circuit model, a process that is time consuming and inflexible to editing.

The SimVascular Lumped Parameter Modeller (svLPM) is a Python based program to facilitate the creation and solving of lumped parameter circuit models for use in cardiovascular computational fluid dynamics simulations and reduced-order validation. svLPM is composed of an intuitive and simplistic User Interface (GUI) that allows users to create circuit models and a solver which enables users to generate and solve the ODE source code for LPMs.

Installation and Running As of 2017-08-09

Please first download and unpackage the svLPMDeveloper Zip File

Mac OSX:
1) Install Python 2.7.13 from https://www.python.org/downloads/
2) Install Homebrew from https://brew.sh
3) When Homebrew has been installed, in the terminal window input:

a) brew install sip
b) brew install cartr/qt4/pyqt

4) The svLPM.py file should now be runnable from IDLE in Python 2.7.13

User Guide

As of 2017-08-09

Introduction

The svLPM User Interface (GUI) is composed of a white scene which functions as the circuit design area, a sidebar of LPM component icons and a status bar. A full list of mouse and keyboard inputs is available on page 8.

As of 2017-08-09, the svLPM GUI is in its first iteration and thus only supports basic functionality for creating LPM.

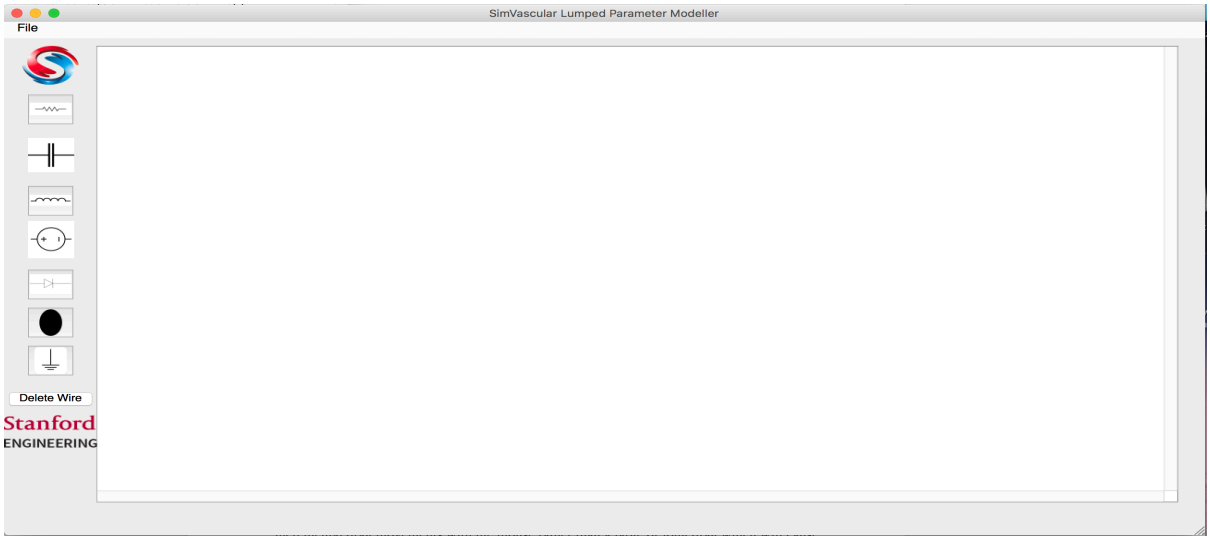

Figure 1: The svLPM GUI

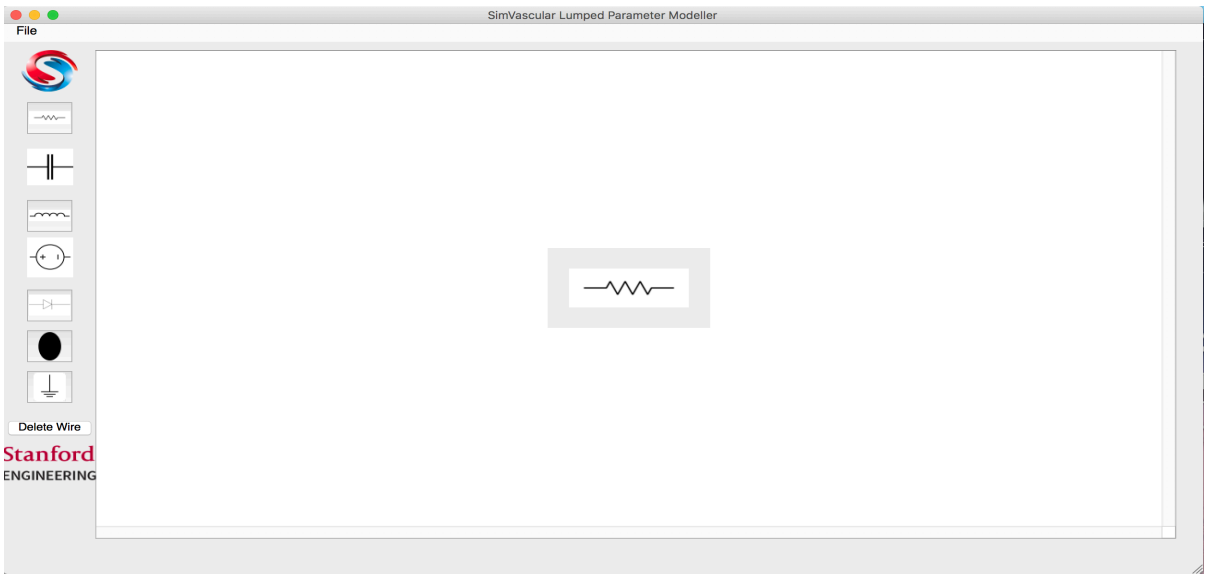

Figure 2: Adding Components will place them at the centre of the scene.

Building LPM Circuits

To add a component to the design area, click on the respective icon to add that type of component widget to the scene. All added component widgets will be created at co-ordinates (0,0) at the centre of the scene as shown in Figure 2. If you have added two components in succession, then dragging the top component will reveal the second component underneath.

To create the circuit model, left press and drag the component widget to a desired position on the scene. Note that the scene will automatically expand when you drag a component to another position. Use the sidebars of the scene to reposition the circuit design area.

As of 2017-08-09 the drag and drop functionality is best realised through incremental drag movements with the mouse rather than a large or long drag which will cause the widget to "shake". For best performance, rotated components should be first moved to the desired position while horizontal, then rotated.

To draw wires between components, first use the middle button on the mouse to click on a component widget. svLPM assigns the start of the wire to a (virtual) node depending on the position of your mouse click. For **non-rotated** components, the node assignment will depend on whether you click on the right or left-hand side of the component widget. For **rotated** components, the node assignment will depend on whether you click on the top or bottom half of the component widget.

To draw wires between the components, right click on the second component that you wish to connect the first component to. The same node assignment functionality applies to the second component with the end of the wire being drawn to the respective node nearest to the position of your right click.

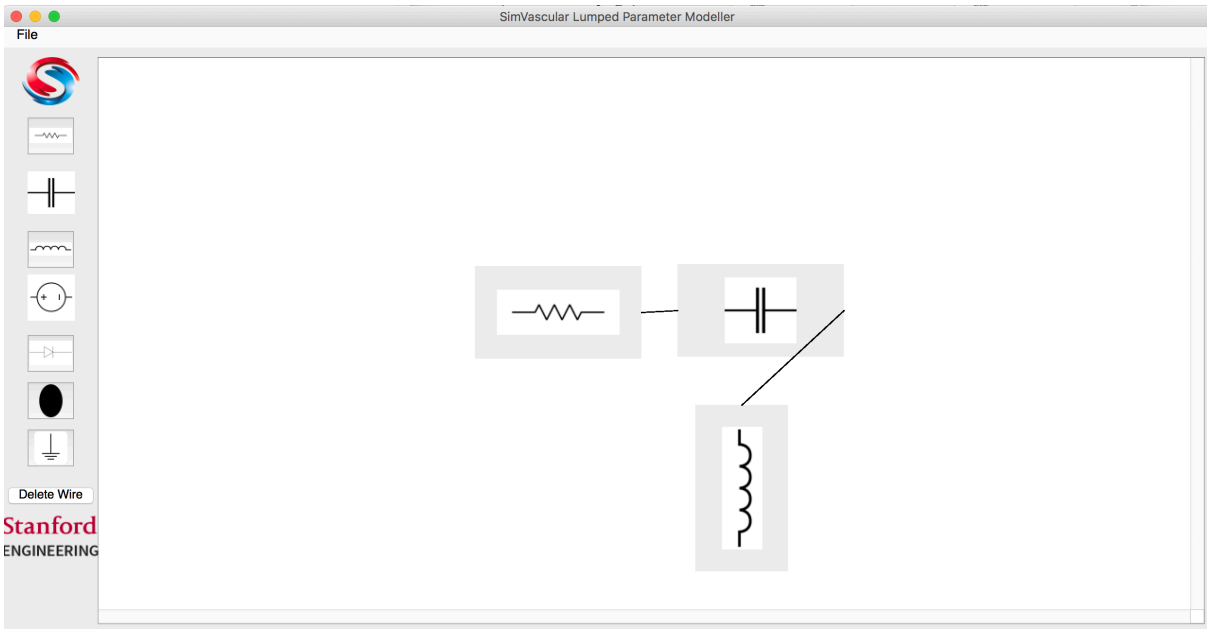

Figure 3: Drawing wires between components: Note that the wire's start and end positions are assigned relative to the position of the click on the component widget.

To delete the previous wire you have added, click the "Delete Wire" button in the sidebar. Each successive click will delete the previous added wire until no wires remain in the scene.

In its first iteration, svLPM is only able to draw lines directly between components in a straight line. This means that circuits in the program will need to be drawn differently from those in a circuit diagram. Therefore, instead of having additional wires that connect to or branch out of a single wire, you will need to connect each component directly to the preceding component in the circuit. Thus, the following RCR block in Figure 4 would look like Figure 5:

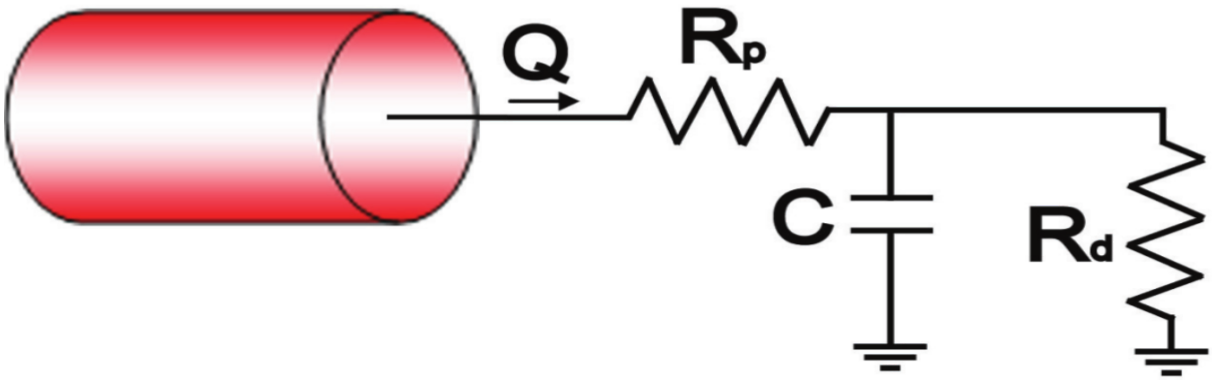

Figure 4: RCR Block for downstream vasculature boundary condition.

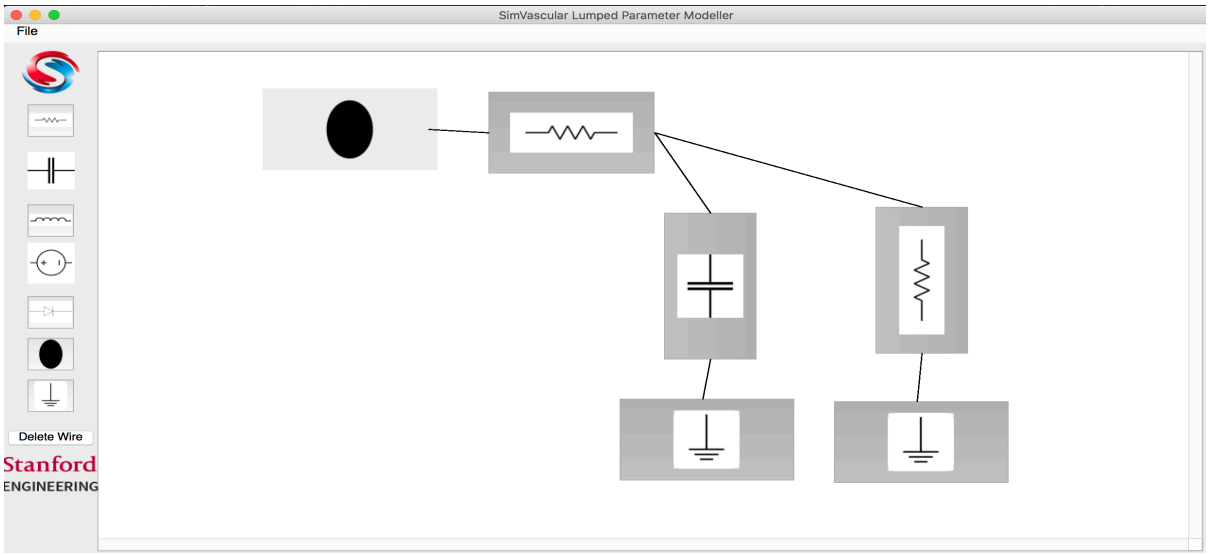

Figure 5: Interpreted svLPM circuit diagram from Figure 4.

To rotate a component, first select the component by middle clicking anywhere on the component. Then left click on the component **whilst** holding down the option key for OSX, or the alt key for Windows. Once a component has been rotated, you will be unable to rotate it back to horizontal without deleting the component and re-adding it. Additionally, to avoid the difficulty of dragging a rotated component, while horizontal drag the component to the desired position and then rotate the widget.

To delete a component, first middle click anywhere on the component. Then left click on the component **whilst** holding down the command key for OSX, or the control key for Windows. The component will be deleted from the scene along with any assigned settings.

Assigning Component Settings

To assign settings to components, left click on anywhere on a component **whilst** holding down the shift key. This will then launch the Component Settings Dialog as shown in Figure 6. In this dialog, the user is able to assign the following:

1) A desired name for the clicked component to differentiate the component from other components of the same type.

2) The component's value.

3) The value's metric prefix: the value of the component will automatically be assigned the appropriate unit based on the component type when the program is run. Therefore, the metric prefix option allows the user to specify the SI prefix of the unit (and thus the respective index power).

When saved, the inputted settings will remain attributes of the component until re-assigned or the component is deleted. When the dialog is launched on the component the next time, the assigned settings will automatically be displayed in the dialog and can be modified.

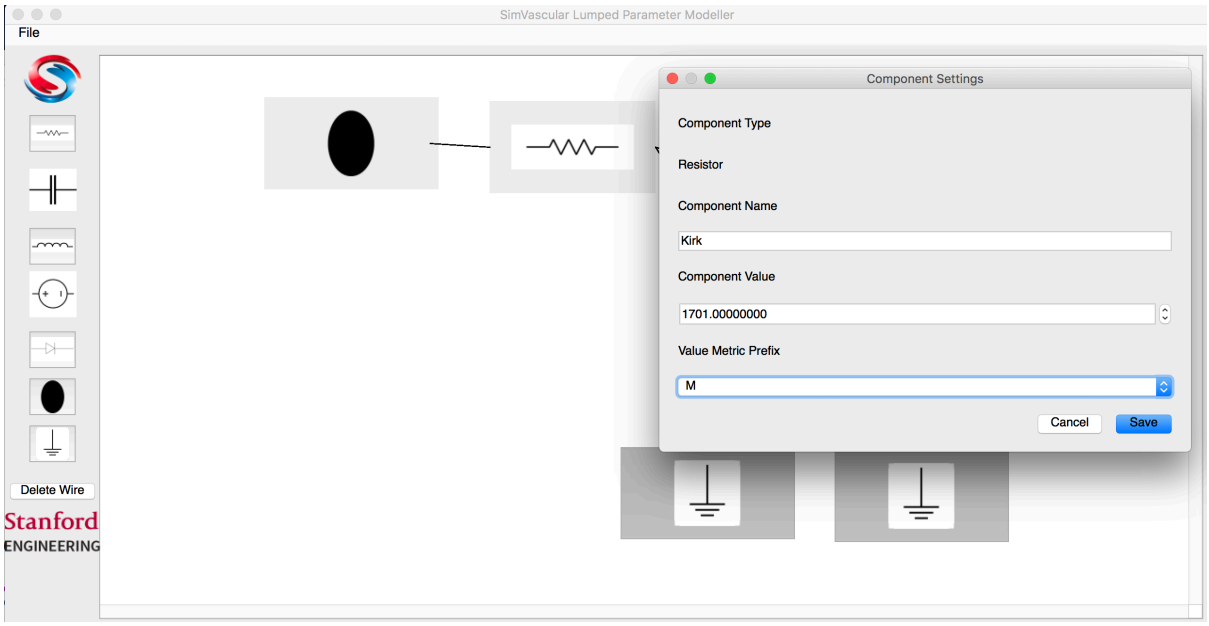

Figure 6: Component Settings Dialog with a resistor named "Kirk" being assigned a value of 1701 Mega Ohms.

Clearing and Quitting

To clear the scene of all components, go to File and click Clear. As the warning dialog which appears suggests, you will lose any unsaved work or changes.

Saving and Opening XML Files

To export a newly created LPM to an XML file to generate the corresponding source code, go to File in the top left-hand corner of the GUI, and click Save As…. Input the desired file name and press save to create the XML file as shown in Figure 7. Note: Do not type .xml after the file name, the program will automatically do this for you.

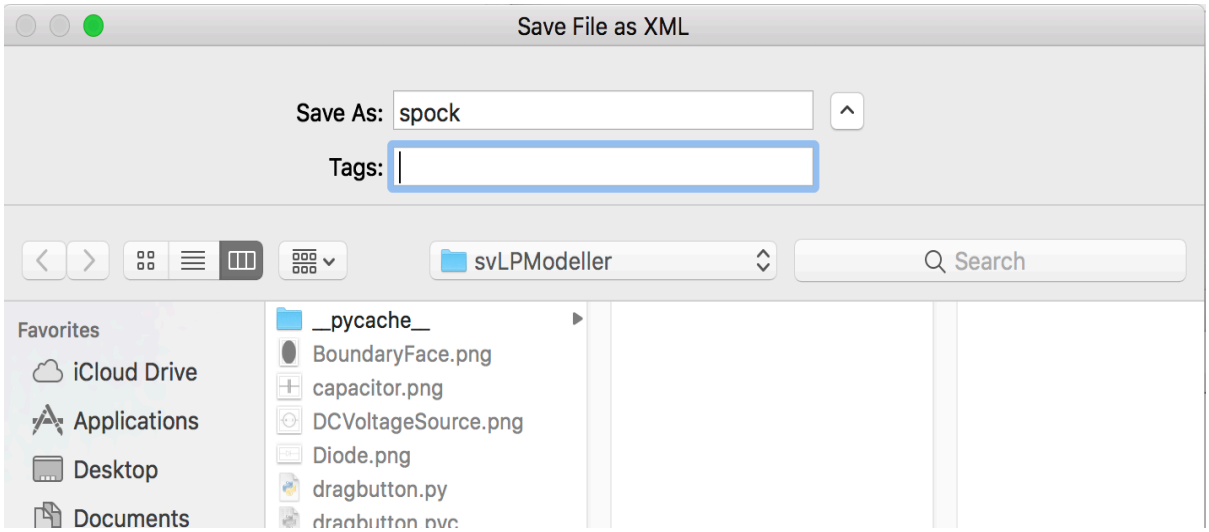

Figure 7: Save As… Functionality. Note: Do not add .xml to the file name

To open a previously created or saved XML file, go to File and click Open and the program will re-create the LPM from the file. To save changes, go to File and click Save or Save As… to create a new XML file.

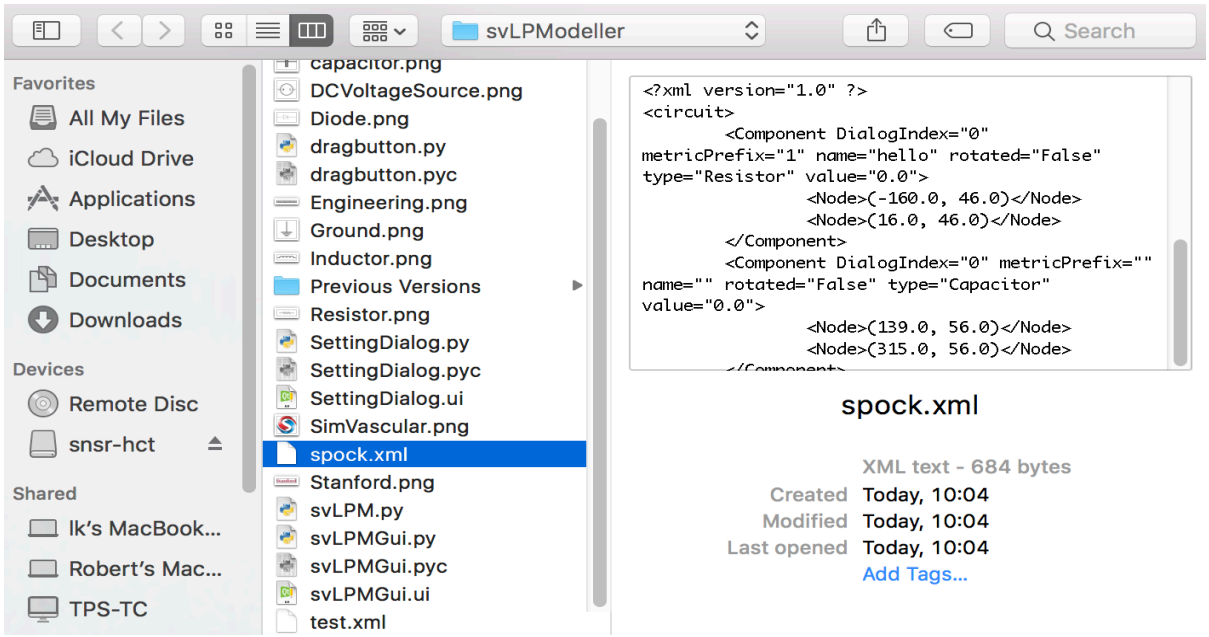

Figure 8: The file will be added to the folder of the svLPM python program.

Summary of GUI Mouse and Keyboard Controls

Building LPM Circuits

Adding Components to Scene: Left Click on Icon

Moving Components on Scene: Left Press and Drag on component

Rotate Component: 1) Middle Click on component
2) Left Click + OPTION/ALT on component

Delete Component: 1) Middle Click on Component
2) Left Click + COMMAND/CONTROL on component

Drawing Wires between Components: 1) Middle Click on Component 1
2) Right Click on Component 2
Note: Node Assignment based on position of Click

Deleting Wire: Left Click on Delete Wire button

Assigning Component Settings

Launch Component Settings Dialog: Left Click + SHIFT on component

Saving and Opening XML Files

Export LPM as XML File: File>Save As…

Open and Load XML File: File>Open

Save changes: File>Save

Clearing and Quitting

Clear Scene/Delete All Components: File>Clear

Quit Application: File>Quit or X in window

Developer Documentation

As of 2017-08-09

svLPM GUI

Developer Background

N.B. The GUI portion of svLPM was initially developed by a novice programmer and thus the python code is likely to not be optimally designed or laid out in the easiest of fashions.

In its original instance, the svLPM GUI was written using:

OS: OSX 10.12.5 Sierra

Language: Python 2.7.12

Frameworks: PyQt4 v4.12.1 GUI development framework from River Bank Computing

Additional Applications: Qt Designer from Qt Creator 4.3

Additional native Python modules imported: xml.dom, xml.etree, os, sys

Additional python files imported: dragbutton.py, SettingDialog.py

Installation and Running

Mac OSX:
1) Install Python 2.7.13 from https://www.python.org/downloads/
2) Install Homebrew from https://brew.sh
3) When Homebrew has been installed, in the terminal window input:

c) brew install sip
d) brew install cartr/qt4/pyqt

4) The svLPM.py file should now be runnable from IDLE in Python 2.7.13

Design and Implementation

The GUI layout was first designed using Qt Designer, an application from Qt that facilitates GUI design. This then generates a corresponding .ui file with the layout which should be saved in the same file as the svLPM.py file. In order to convert the .ui file to an importable python code (Note for OSX but similar process should be applicable in Windows/Linux:

1) In Terminal navigate to the folder containing the .ui file. E.g. for OSX

   cd /users/Picard/Desktop/svLPModeller

2) Convert the .ui file to a .py file using the following command:

   pyuic4 svLPMGui.ui -o svLPMGui.py

3) This process will need to be repeated after each edit of the .ui GUI layout file in Qt Designer to have the changes implemented in the python code and in the GUI.

The svLPMGui.py file will now be importable to the main svLPM code. Note that you should not edit the svLPMGui.py code generated from the .ui file as you will lose any changes made when overwriting the gui.py file after an edit in Qt Designer.

Python Code Design

Please refer to the svLPMDeveloper.py file for a fully commented version of the svLPM.py file explaining method implementation and design choices.

Useful Resources

The following were invaluable in constructing svLPM:

1) Qt4 Documentation http://doc.qt.io/qt-4.8/
2) PyQt4 Documentation http://pyqt.sourceforge.net/Docs/PyQt4/classes.html
3) Introductory pyqt4 tutorial series: https://www.youtube.com/watch?v=JBME1ZyHiP8&list=PLQVvvaa0QuDdVpDFNq4FwY9APZPGSUyR4
4) Stack Overflow pyqt community : https://stackoverflow.com/questions/tagged/pyqt
5) SimVascular: http://simvascular.github.io

Setting up svLPM.py as an OSX Application

Note that currently the application is not distributable

For information regarding APIs other than the below mentioned for deploying the python code, please refer to https://wiki.python.org/moin/PyQt/Deploying_PyQt_Applications

The svLPM Application for Mac OSX was created using the py2app API which allows a user to run the program without the need to install python 2.7, PyQt4 or Qt themselves.
See the following for more information: https://py2app.readthedocs.io/en/latest/

To deploy the svLPM.py file to a distributable Mac OSX Application first use the following commands in the Terminal:

1) **pip install -U py2app**

2) In Terminal navigate to the svLPMModeller folder.

   cd /users/Picard/Desktop/svLPModeller

3) **Py2applet --make-setup svLPM.py Resistor.png, Capacitor.png DCVoltageSource.png Diode.png Ground.png Engineering.png Stanford.png SimVascular.png Inductor.png BoundaryFace.png**

After the setup.py file is generated in the svLPModeller folder edit it with the following, save and close the file:

1) Set **OPTIONS = {'iconfile':'LPM.png.icns','argv_emulation': False, 'includes': ['sip', 'PyQt4', 'PyQt4.QtCore', 'PyQt4.QtGui', 'PyQt4._qt']}**

Return to the Terminal and input the following commands after navigating back to the svLPModeller file :

1) **rm -rf build dist**

2) **python setup.py py2app**

In Finder do the following:

1) Go to the svLPModeller folder and enter the dict folder

2) In dict, right click on the svLPM App and click Show Package Contents

3) Navigate to the following folder and make it copy of it :
   Contents>Frameworks>QtGui.Framework>Versions>4>Resources>qt_menu.nib

4) Paste the qt_menu.nib folder to:
   svLPM>Show Package Contents>Contents>Resources

5) Return to the dict folder in the svLPModeller folder: double click to run svLPM