Biophysical Journal
Computational Tool

Biophysical Society

Microvessel Chaste: An Open Library for Spatial Modeling of Vascularized Tissues

James A. Grogan,[1,*] Anthony J. Connor,[1,2] Bostjan Markelc,[3] Ruth J. Muschel,[3] Philip K. Maini,[1] Helen M. Byrne,[1] and Joe M. Pitt-Francis[2]

[1]Wolfson Centre for Mathematical Biology, Mathematical Institute, [2]Department of Computer Science, and [3]CRUK/MRC Oxford Institute for Radiation Oncology, University of Oxford, Oxford, United Kingdom

ABSTRACT Spatial models of vascularized tissues are widely used in computational physiology. We introduce a software library for composing multiscale, multiphysics models for applications including tumor growth, angiogenesis, osteogenesis, coronary perfusion, and oxygen delivery. Composition of such models is time consuming, with many researchers writing custom software. Recent advances in imaging have produced detailed three-dimensional (3D) datasets of vascularized tissues at the scale of individual cells. To fully exploit such data there is an increasing need for software that allows user-friendly composition of efficient, 3D models of vascularized tissues, and comparison of predictions with in vivo or in vitro experiments and alternative computational formulations. Microvessel Chaste can be used to build simulations of vessel growth and adaptation in response to mechanical and chemical stimuli; intra- and extravascular transport of nutrients, growth factors and drugs; and cell proliferation in complex 3D geometries. In addition, it can be used to develop custom software for integrating modeling with experimental data processing workflows, facilitated by a comprehensive Python interface to solvers implemented in C++. This article links to two reproducible example problems, showing how the library can be used to build simulations of tumor growth and angiogenesis with realistic vessel networks.

INTRODUCTION

Spatial models of vascularized tissue are used to study the growth and response to treatment of tumors (1), angiogenesis (2), osteogenesis (3), coronary perfusion (4), and tissue oxygenation (5). Such models typically comprise a combination of the following: 1) agent-based or continuum representations of migrating and proliferating cells; 2) line-based or spatially resolved representations of microvessels; 3) the solution of blood, nutrient, growth factor, and drug transport problems in vessel networks whose geometry and connectivity may evolve; 4) the solution of growth factor and drug transport problems in the evolving extravascular space; and 5) vessel formation and endothelial tip cell migration in response to mechanical and chemical cues.

Composition of spatial models of vascularized tissues is a time-consuming process, with most researchers writing custom software. Examples of such multiscale/hybrid models include those developed by Anderson and Chaplain (6), Alarcón et al. (7), Frieboes et al. (8), Shirinifard et al. (9), Owen et al. (1), Perfahl et al. (10), Welter and Rieger (11), Secomb et al. (2), and Boas and Merks (12). The need to account for many biological phenomena, including multiscale spatial processes occurring over different timescales, has made the development of more general software frameworks challenging (13,14). Tools for integration with experimental imaging data and model benchmarking and cross comparison are also important (14,15). Several groups, including Liu et al. (16), Secomb et al. (2), and Beard et al. (17), have produced more general software that focuses on modeling and integration with imaging data. Notable efforts in the development of vascularized tissue models have been undertaken as part of the broader European Union's Virtual Physiological Human Project (http://www.vph-institute.org/) and the National Institute of General Medical Sciences' Virtual Physiological Rat (http://www.virtualrat.org/) program.

In this article, we introduce Microvessel Chaste, an open-source Python/C++ library for composing spatial models of vascularized tissues. It is designed so that users can assemble their own models from a collection of building blocks, based on established numerical libraries. It is a plug-in for the Chaste C++ library for problems in computational physiology and biology (18), adding functionality

Submitted December 21, 2016, and accepted for publication March 27, 2017.

*Correspondence: grogan@maths.ox.ac.uk

Editor: Fazoil Ataullakhanov.

http://dx.doi.org/10.1016/j.bpj.2017.03.036

for modeling microvessels and complex, evolving three-dimensional (3D) tissue domains. Development has been motivated by the above-mentioned models and software, following similar goals in the development of detailed models of vascularized tissues and integration of experimental data. However, there is an additional focus on providing a user-friendly, general framework for model composition, in a manner similar to that in which Chaste (18), CompuCell3D (19), EPISIM (20), and PhysiCell (21) can be used to compose tissue models with agent-based representations of cells. A novel (to our knowledge) and important feature of Microvessel Chaste is its comprehensive Python interface, which facilitates integration with a growing collection of scientific Python software for image processing, statistical analysis, and visualization. The library has facilitated the integration of modeling with high-resolution 3D imaging data, as shown in Grogan et al. (22) and below, and will be useful in future model cross-comparison studies.

The remainder of the article is structured as follows. The next section introduces the main components of the Chaste and Microvessel Chaste libraries. We then present examples showing how these components can be combined to generate numerical solutions of biophysical models of interest, focusing on tumor growth and angiogenesis. Integration with experimental imaging data is also demonstrated.

MATERIALS AND METHODS

Fig. 1 shows the main components of Chaste and Microvessel Chaste and how they build on well-known numerical libraries. Mirams et al. (18) and Osborne et al. (15) should be consulted for a detailed description of the Chaste library and its use in individual cell-based models, respectively. However, a brief summary is given here. These components can be used to assemble custom models for a broad variety of vascularized tissue applications in a flexible and modular way.

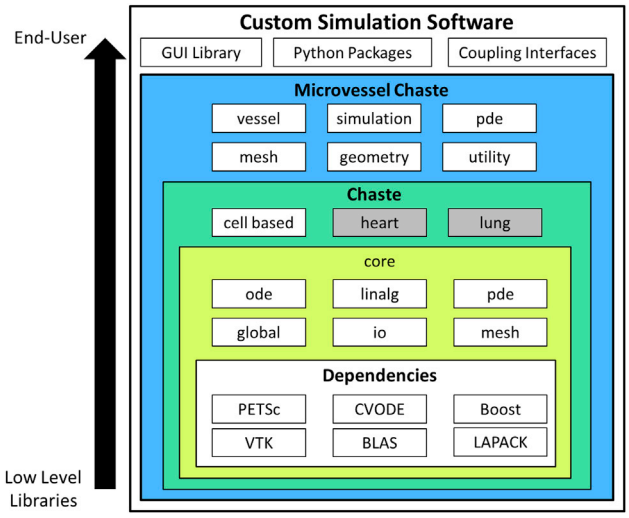


FIGURE 1 Given here is a schematic showing the components used in the Microvessel Chaste library, and how they can be used to generate custom simulation software. Shaded components are not currently used in the library. To see this figure in color, go online.

Chaste is a collection of C++ classes, interfacing established parallel linear algebra, ordinary differential equation (ODE) solver, and visualization libraries. The core component has low-level functionality for solving ODE systems, constructing finite-element solvers, reading and writing meshes, and general linear algebra operations. Interfaces to underlying libraries provide full access to solution controls, such as time stepping and tolerances. The component also has a selection of prebuilt finite-element solvers for elliptic and parabolic partial differential equations (PDEs), coupled ODE and PDE systems, and nonlinear mechanics problems, allowing users to construct detailed and bespoke biophysical tissue models (18). The core functionality is extended by the cell-based component, allowing modeling of tissues with agent-based descriptions of cells. Functionality includes cell cycle modeling, based on the solution of ODE systems, and intra- and extracellular chemical transport modeling, based on the solution of PDEs with cells acting as discrete sinks or sources of the chemicals. Various discrete on- and off-lattice cell representations and mechanical models are available, including cellular Potts, vertex-based methods, and center-based methods (15).

The Microvessel Chaste library uses the Chaste components through their C++ API. It adds functionality for modeling microvessels using line- or surface-based descriptions, allowing a wide range of existing models to be implemented or extended (1,10,22). The geometry component is a collection of tools for construction and manipulation of 3D volumes and surfaces, which is useful for detailed modeling of microvessel walls or anatomical features such as the cornea, demonstrated below. The vessel component has tools for artificial vessel network construction, reading real networks from file, and network characterization (such as the construction of line or branch density maps). The simulation component contains many well-known submodels of vessel network blood flow, including nonlinear blood rheology (23), plasma splitting (24), structural adaptation in response the chemical and mechanical cues (25), and sprouting angiogenesis (1). The Microvessel Chaste library also extends existing Chaste components. The extended PDE component contains PETSc-based finite difference solvers, allowing chemical transport PDEs to be solved in a manner typical of the literature (1). Chemical release and uptake from vessels can be modeled by including their action as discrete sink or source terms in chemical transport PDEs (1). The mesh component has tools for automatic meshing of complex 3D geometries, facilitating construction of detailed models of growing tissues.

The reader is referred to Owen et al. (1), Perfahl et al. (10), and Grogan et al. (22) for detailed descriptions of the biological background of incorporated submodels, how they can be coupled, and how they are parameterized using experimental data. Because Microvessel Chaste is a library, users can build their own models, choosing suitable submodels, coupling schemes, and time-stepping strategies. The aforementioned publications (1,10,22), web-based tutorials, and software API documentation provide guidance in this regard. Most default experimental parameters in the software are tagged with a literature source and typed over unit, while API documentation and solver names indicate the original literature sources for submodels.

RESULTS AND DISCUSSION

In this section, tumor growth and angiogenesis problems are demonstrated; they are available for reproduction at https://jmsgrogan.github.io/MicrovesselChaste. A collection of additional, simpler examples is also available at this location. The examples are simplified to facilitate the tutorial format and reproduction by the reader. Careful parameterization is required before they can be used to gain new biological insights. The examples cover only a small selection of available functionality, and readers are

encouraged to consult the software's web page and API documentation for further details.

A 3D tumor growth simulation

The first example, shown in Fig. 2, is a 3D simulation of tumor growth in a vessel network geometry obtained using multiphoton imaging after implantation of MC38 tumor cells in a mouse (22). This hybrid, multiscale tumor growth model is similar to many in the literature (6,8,10,26), and in particular, uses submodels and parameter values described in Owen et al. (1). However, the use of a large, realistic and evolving tumor vessel network distinguishes this example from previous studies. The simulation is facilitated by recent advances in intravital imaging, which allow in vivo observation of tumor growth at the scale of individual cells, and the new preprocessing and modeling functionality in Microvessel Chaste. The vessel network preprocessing time was 6 s, and 25 simulated hours of tumor growth took 25 min on a standard desktop PC.

The problem is initialized with a 3D region of the tumor vessel network. A regular lattice with spacing of 40 μm is generated in the bounding box of the network geometry, using the Microvessel Chaste mesh and vessel components shown in Fig. 1. A cellular automaton-based cell population fills all lattice sites, including those occupied by vessels, using the Chaste cell-based component. "Tumor" cell types are assigned to a 300-μm-diameter central cylindrical region and "Normal" types to the remainder. Cell cycling is represented by a subcellular model described in Owen et al. (1), which leads to oxygen-dependent proliferation rates and vascular endothelial growth factor (VEGF) release rates that differ across cell populations. The cycle model is solved as a system of ODEs using the Chaste ODE component, with a time step of 5 min. Cells far from oxygen-rich vessels experience low oxygen levels and, as a result, become hypoxic and release VEGF. Oxygen and VEGF transport in the domain are each described using a typical steady-state reaction diffusion PDE of the following form (1):

$$D\nabla^2 c + \rho(c_b - c) + kc - \lambda c = 0, \qquad (1)$$

where c is the extravascular concentration of oxygen or VEGF; D is an effective diffusivity; c_b is the vascular concentration; ρ is an effective vascular permeability that is nonzero and positive only on lattice sites occupied by vessels; k is a cell consumption or release rate, depending on species, and is nonzero only on lattice sites occupied by cells; and λ is a positive rate of natural decay, only relevant for VEGF. PDEs are solved on the same regular grid as the cellular automaton, using the finite difference solver in the Microvessel Chaste PDE component. Many alternative PDEs can be constructed using the software, including the addition of general nonlinear sink and source terms and relaxation of the steady-state assumption. The solver is constructed in a standard way using available PETSc features. Transport PDEs are solved and cell positions are updated at each 30-min global time step. Global time stepping is managed by the Chaste cell-based component. VEGF stimulates the sprouting and chemotactic migration of new vessels from the existing vasculature. Sprouts form at a rate dependent on VEGF concentration and perform a lattice-free persistent random walk biased toward nearby vessels

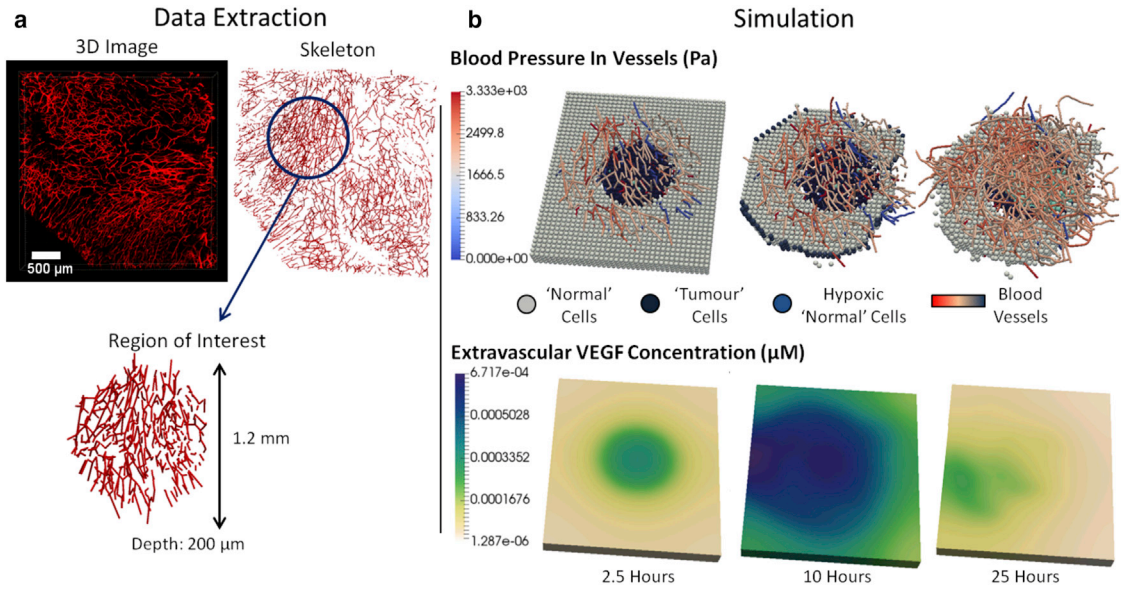


FIGURE 2 (*a*) A 3D intravital image of a tumor microvessel network (*red*) is obtained and a skeleton extracted as described in Grogan et al. (22). A cylindrical region of interest with diameter 1.2 mm is extracted for the example simulation. (*b*) A tumor growth simulation using the extracted microvessel network and Microvessel Chaste is given. The predicted evolution of the tumor over 25 h is shown, including blood pressure in growing vessels, VEGF concentrations in the extravascular space, and discrete cells. To see this figure in color, go online.

and positive VEGF gradients, using functionality in the simulation component. Off- and on-lattice tip migration submodels are easily switched, allowing model comparisons. Vessel movement occurs once per global time increment. Further details on modeling, coupling, and time stepping can be found in Owen et al. (1).

Blood flow is modeled in the branching vessel network using a typical 1D simplification (23). A pressure difference is applied across the network with a drop of 1.33 kPa through the 200-μm depth. Only perfused vessels deliver oxygen. Vessel radii can change in response to mechanical stimuli following typical models in the literature (25). Blood flow and structural adaptation models were used from the simulation component. At later times, cells far from the vessels become apoptotic. The surviving tumor cells gradually invade the domain at the expense of the normal cells. This process is similar to those observed in the simulations of Perfahl et al. (10), Anderson and Chaplain (6), and others. There are many potential extensions to models of this type, including simulated administration of chemotherapeutic and antiangiogenic drugs (7) and radiotherapy (22). These cases can be simulated using the Microvessel Chaste library.

A 3D angiogenesis simulation in a curved domain

Our second example is a 3D, off-lattice simulation of angiogenesis in a curved geometry (see Fig. 3). This example demonstrates geometry manipulation and the solution of PDEs on 3D domains. Again, the example is simplified for the purposes of a tutorial and requires careful parameterization before it can be used to gain biological insights. The application is appropriate for the corneal micropocket assay that is widely used to study angiogenesis (26). Typical experimental results are shown in Fig. 3 *a*. The simulation time was 10 s on a standard desktop PC.

In this experimental assay a pellet containing an angiogenic growth factor (for example, VEGF) is implanted in the cornea. VEGF diffuses from the pellet into the corneal tissue and stimulates endothelial cells lining existing vessels at the base to form sprouts. The sprouts then migrate toward the pellet along spatial gradients in VEGF. This example follows a common modeling paradigm where agent-based representations of cells are not included, but individual vessels are (2). As shown in Fig. 3 *b*, the cornea is represented as a hemispherical domain of radius 1.4 mm and thickness 0.1 mm, generated using the geometry component and meshed with linear tetrahedra using the Microvessel Chaste mesh component. The pellet is a cuboid with side length 0.3 mm and depth 0.1 mm, and a prescribed VEGF concentration of 3.0 nM on the boundaries. In practice, the VEGF in the pellet will deplete. Over time, vessels sprout from a large preexisting vessel at the base, or limbus, at a rate dependent on VEGF concentration. Then, they follow a persistent random walk, biased toward other vessels and positive VEGF gradients, as previously described. The VEGF distribution is obtained by solving the reaction-diffusion PDE in Eq. 1 on the cornea at the start of the simulation, in the absence of vessel or cell terms, with a fixed concentration maintained on the pellet by means of a Dirichlet boundary condition. For simplicity, the PDE solution is then fixed for the remainder of the simulation. In this case, a finite-element solver is used, available in the Chaste PDE component.

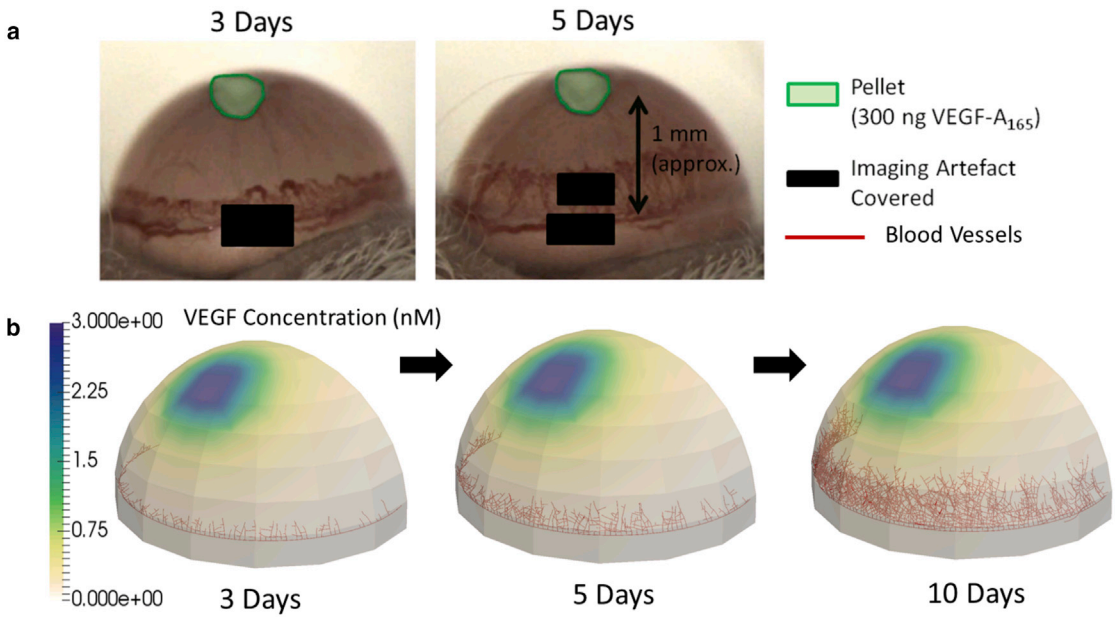


FIGURE 3 (*a*) Given here are images from a cornea micropocket experiment showing microvessels (*dark red*) at 3–5 days postpellet implantation (26). (*b*) Shown here is application of the Microvessel Chaste library in modeling a similar experiment. To see this figure in color, go online.

Vessels migrate toward the pellet, remaining within the volume of the 3D cornea geometry, with this boundary condition facilitated by tools in the geometry component. Possible extensions to this simple model include the addition of discrete stromal cells, distinction between perfused and unperfused vessels, subcellular signaling, VEGF depletion and consumption by cells, and the use of multiple vessel growth factors, as per Connor et al. (26).

CONCLUSIONS

Microvessel Chaste, a library for composing multiphysical, multiscale spatial models of vascularized tissues, has been demonstrated, and two reproducible sample problems in the areas of tumor growth and angiogenesis presented. Familiarity with object-oriented programming, C++ or Python, and standard numerical methods are recommended. Users can develop their own tissue models, vascularized or otherwise, using a variety of discrete or spatially resolved representations of cells and blood vessels. Combination with existing Python packages for image processing and statistical analysis further enhances integration into experimental data processing workflows.

As shown in Fig. 1, the library can also be used to construct more general custom simulation software, for subsequent use by end-users less familiar with programming. GUIs can be created for specific tasks using wxPython, for example, and the 3D rendering back-end included in the library. Given these possibilities, we believe that the library will be useful for detailed model cross comparisons, closer integration of modeling and experimental data, and exploration of suitable coupling and time-stepping schemes for multiscale phenomena. All of these are major challenges in biophysical modeling.

There are some limitations. Windows and MacOS support are currently only available through Docker images, but work is ongoing to develop releases for these platforms. At present, most algorithms have not been parallelized, but all PDE and flow solvers are based on PETSc structures and the vessel network components may be communicated across processors using existing serialization functionality in Chaste (27). Additional functionality for semiautomated 2D and 3D image segmentation and meshing is under development. This includes the ability to generate finite-element meshes from 2D and 3D images automatically. It is envisaged that this will further aid integration with experimental studies such as those shown in Fig. 2 *a*.

The library is available under a permissive BSD license, with source files and documentation available via the project Github page https://jmsgrogan.github.io/MicrovesselChaste/. Contributions are welcome via Github pull requests and issues can be reported via the Github issue tracker. The latest release, version 3.4.2, is archived at http://dx.doi.org/10.5281/zenodo.213148.

SUPPORTING MATERIAL

Supporting Materials and Methods is available at http://www.biophysj.org/biophysj/supplemental/S0006-3495(17)30384-3.

AUTHOR CONTRIBUTIONS

J.A.G., A.J.C., P.K.M., H.M.B., and J.M.P.-F. designed the models and software. J.A.G., A.J.C., and J.M.P.-F. developed the software. B.M. and R.J.M. designed the experimental imaging. B.M. performed the experimental imaging. J.A.G., A.J.C., B.M., P.K.M., H.M.B., and J.M.P.-F. drafted and edited the article. All authors read and approved the final article.

ACKNOWLEDGMENTS

The authors acknowledge helpful input from the Chaste development team, in particular Jonathan Cooper, Alex Fletcher, James Osborne, Gary Mirams, and Martin Robinson.

The research leading to these results has received funding from the People Program (Marie Curie Actions) of the European Union's Seventh Framework Program (FP7/2007-2013) under REA grant No 625631 (to B.M.) and the European Union's Seventh Framework Program for Research, Technological Development, and Demonstration under grant No. 600841 (to J.A.G., A.J.C., H.M.B., and J.M.P.-F.). B.M. and R.J.M. acknowledge that this work was also supported by Cancer Research UK (CRUK) grant No. C5255/A18085, the CRUK Oxford Centre under CRUK grant No. C5255/A15935, and the CRUK/EPSRC Oxford Cancer Imaging Centre under grant No. C5255/A16466.

REFERENCES

1. Owen, M. R., I. J. Stamper, …, H. M. Byrne. 2011. Mathematical modeling predicts synergistic antitumor effects of combining a macrophage-based, hypoxia-targeted gene therapy with chemotherapy. *Cancer Res.* 71:2826–2837.
2. Secomb, T. W., J. P. Alberding, …, A. R. Pries. 2013. Angiogenesis: an adaptive dynamic biological patterning problem. *PLoS Comput. Biol.* 9:e1002983.
3. Carlier, A., L. Geris, …, H. Van Oosterwyck. 2012. MOSAIC: a multiscale model of osteogenesis and sprouting angiogenesis with lateral inhibition of endothelial cells. *PLoS Comput. Biol.* 8:e1002724.
4. Smith, A. F., R. J. Shipley, …, N. P. Smith. 2014. Transmural variation and anisotropy of microvascular flow conductivity in the rat myocardium. *Ann. Biomed. Eng.* 42:1966–1977.
5. Beard, D. A., and J. B. Bassingthwaighte. 2001. Modeling advection and diffusion of oxygen in complex vascular networks. *Ann. Biomed. Eng.* 29:298–310.
6. Anderson, A. R. A., and M. A. J. Chaplain. 1998. Continuous and discrete mathematical models of tumor-induced angiogenesis. *Bull. Math. Biol.* 60:857–899.
7. Alarcón, T., M. R. Owen, …, P. K. Maini. 2006. Multiscale modelling of tumour growth and therapy: the influence of vessel normalisation on chemotherapy. *Comput. Math. Methods Med.* 7:85–119.
8. Frieboes, H. B., J. S. Lowengrub, …, V. Cristini. 2007. Computer simulation of glioma growth and morphology. *Neuroimage.* 37 (*Suppl 1*):S59–S70.
9. Shirinifard, A., J. S. Gens, …, J. A. Glazier. 2009. 3D multi-cell simulation of tumor growth and angiogenesis. *PLoS One.* 4:e7190.
10. Perfahl, H., H. M. Byrne, …, M. R. Owen. 2011. Multiscale modelling of vascular tumour growth in 3D: the roles of domain size and boundary conditions. *PLoS One.* 6:e14790.

11. Welter, M., and H. Rieger. 2013. Interstitial fluid flow and drug delivery in vascularized tumors: a computational model. *PLoS One.* 8:e70395.
12. Boas, S. E. M., and R. M. H. Merks. 2015. Tip cell overtaking occurs as a side effect of sprouting in computational models of angiogenesis. *BMC Syst. Biol.* 9:86.
13. Connor, A. J., J. Cooper, …, S. McKeever. 2012. Object-oriented paradigms for modelling vascular tumor growth: a case study. *In* The Fourth International Conference on Advances in Systems Simulation. Iaria, Lisbon, Portugal, pp. 74–83.
14. Rieger, H., and M. Welter. 2015. Integrative models of vascular remodeling during tumor growth. *Wiley Interdiscip. Rev. Syst. Biol. Med.* 7:113–129.
15. Osborne, J. M., A. G. Fletcher, …, D. J. Gavaghan. 2017. Comparing individual-based approaches to modelling the self-organization of multicellular tissues. *PLoS Comput. Biol*. 13:e1005387.
16. Liu, G., A. A. Qutub, …, A. S. Popel. 2011. Module-based multiscale simulation of angiogenesis in skeletal muscle. *Theor. Biol. Med. Model.* 8:6.
17. Beard, D. A., M. L. Neal, …, B. E. Carlson. 2012. Multiscale modeling and data integration in the virtual physiological rat project. *Ann. Biomed. Eng.* 40:2365–2378.
18. Mirams, G. R., C. J. Arthurs, …, D. J. Gavaghan. 2013. Chaste: an open source C++ library for computational physiology and biology. *PLoS Comput. Biol.* 9:e1002970.
19. Swat, M. H., G. L. Thomas, …, J. A. Glazier. 2012. Multi-scale modeling of tissues using CompuCell3D. *Methods Cell Biol.* 110:325–366.
20. Sütterlin, T., C. Kolb, …, N. Grabe. 2013. Bridging the scales: semantic integration of quantitative SBML in graphical multi-cellular models and simulations with EPISIM and COPASI. *Bioinformatics.* 29: 223–229.
21. Macklin, P., M. E. Edgerton, …, V. Cristini. 2012. Patient-calibrated agent-based modelling of ductal carcinoma in situ (DCIS): from microscopic measurements to macroscopic predictions of clinical progression. *J. Theor. Biol.* 301:122–140.
22. Grogan, J. A., B. Markelc, …, H. M. Byrne. 2017. Predicting the influence of microvascular structure on tumour response to radiotherapy. *IEEE Trans. Biomed. Eng.* 64:504–511.
23. Alarcón, T., H. M. Byrne, and P. K. Maini. 2005. A design principle for vascular beds: the effects of complex blood rheology. *Microvasc. Res.* 69:156–172.
24. Pries, A. R., K. Ley, …, P. Gaehtgens. 1989. Red cell distribution at microvascular bifurcations. *Microvasc. Res.* 38:81–101.
25. Pries, A. R., T. W. Secomb, and P. Gaehtgens. 1998. Structural adaptation and stability of microvascular networks: theory and simulations. *Am. J. Physiol.* 275:H349–H360.
26. Connor, A. J., R. P. Nowak, …, H. M. Byrne. 2015. An integrated approach to quantitative modelling in angiogenesis research. *J. R. Soc. Interface.* 12:0546.
27. Harvey, D. G., A. G. Fletcher, …, J. M. Pitt-Francis. 2015. A parallel implementation of an off-lattice individual-based model of multicellular populations. *Comput. Phys. Commun.* 192:130–137.

Biophysical Journal, Volume *112*

Supplemental Information

Microvessel Chaste: An Open Library for Spatial Modeling of Vascularized Tissues

James A. Grogan, Anthony J. Connor, Bostjan Markelc, Ruth J. Muschel, Philip K. Maini, Helen M. Byrne, and Joe M. Pitt-Francis

This tutorial is automatically generated from the file test/python/tutorials//TestPythonBiologicalNetworkLiteratePaper.py.

In [1]:
```
# Jupyter notebook specific imports
import matplotlib as mpl
from IPython import display
%matplotlib inline
```

A Tumour Growth Tutorial With A Real Network

This tutorial is designed to introduce a tumour growth problem based on a simplified version of the vascular tumour application described in Owen et al. 2011 (http://www.ncbi.nlm.nih.gov/pubmed/21363914).

It is a 3D simulation using cellular automaton for cells, lattice free migration for vessel movement and a regular grid for the solution of partial differential equations for oxygen and VEGF transport using the finite difference method.

The Test

In [2]:
```
import chaste # Core Chaste functionality
import chaste.cell_based # Chaste Cell Populations
chaste.init() # Initialize MPI and PETSc
import microvessel_chaste # Core Microvessel Chaste functionality
import microvessel_chaste.geometry # Geometry tools
import microvessel_chaste.mesh # Meshing
import microvessel_chaste.population.vessel # Vessel tools
import microvessel_chaste.pde # PDE and solvers
import microvessel_chaste.simulation # Flow and angiogenesis solvers
import microvessel_chaste.visualization # Visualization
from microvessel_chaste.utility import * # Dimensional analysis: bring in all units for convenience
# Set up the test
chaste.cell_based.SetupNotebookTest()
```

Set up output file management and seed the random number generator.

In [3]:
```
file_handler = chaste.core.OutputFileHandler("Python/TestBiologicalNetworkLiteratePaper")
chaste.core.RandomNumberGenerator.Instance().Reseed(12345)
```

This component uses explicit dimensions for all quantities, but interfaces with solvers which take non-dimensional inputs. The BaseUnits singleton takes time, length and mass reference scales to allow non-dimensionalisation when sending quantities to external solvers and re-dimensionalisation of results. For our purposes microns for length and hours for time are suitable base units.

In [4]:
```
reference_length = 1.e-6*metre()
reference_time = 3600.0*second()
reference_concentration = 1.e-6*mole_per_metre_cubed()
BaseUnits.Instance().SetReferenceLengthScale(reference_length)
BaseUnits.Instance().SetReferenceTimeScale(reference_time)
BaseUnits.Instance().SetReferenceConcentrationScale(reference_concentration)
```

Read a vessel network derived from biological images from file

In [5]:
```
vessel_reader = microvessel_chaste.population.vessel.VesselNetworkReader3()
vessel_reader.SetFileName("bio_original.vtp")
vessel_reader.SetMergeCoincidentPoints(True)
vessel_reader.SetTargetSegmentLength(40.0e-6*metre())
network = vessel_reader.Read()
```

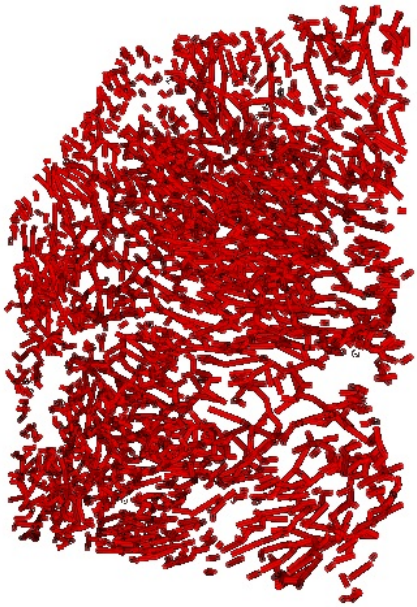
The vessel network may contain short vessels due to image processing artifacts, we remove any vessels that are on the order of a single cell length and are not connected to other vessels at both ends. Note that units are explicitly specified for all quantities. It is ok to allow some small disconnected regions to remain for our purposes. The network is large, this can take up to 30 seconds.

In [6]:
```
short_vessel_cutoff = 40.0e-6 * metre()
remove_end_vessels_only = True
network.RemoveShortVessels(short_vessel_cutoff, remove_end_vessels_only)
network.UpdateAll()
network.MergeCoincidentNodes()
network.UpdateAll()
```

Write the modified network to file for inspection and visualize it.

In [7]:
```
network.Write(file_handler.GetOutputDirectoryFullPath() + "cleaned_network.vtp")
scene = microvessel_chaste.visualization.MicrovesselVtkScene3()
scene.SetVesselNetwork(network)
scene.GetVesselNetworkActorGenerator().SetEdgeSize(20.0)
nb_manager = microvessel_chaste.visualization.JupyterNotebookManager()
nb_manager.vtk_show(scene, height=600, width = 1000)
```

Out[7]:



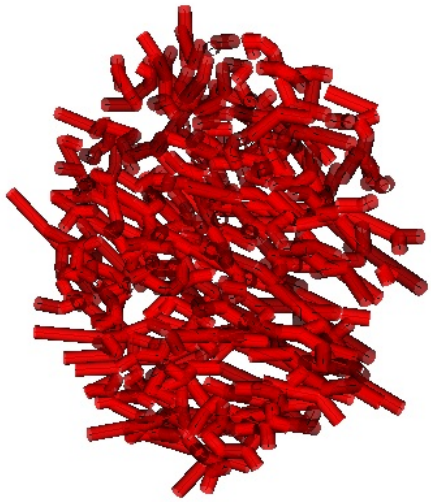
Simulating tumour growth for the entire network would be prohibitive for this tutorial, so we sample a small region. We can use some geometry tools to help.

In [8]:
```
cylinder = microvessel_chaste.geometry.Part3()
centre = microvessel_chaste.mesh.DimensionalChastePoint3(2300.0, 2300.0, -5.0, 1.e-6*metre())
radius = 600.0e-6*metre()
depth = 205.e-6*metre()
cylinder.AddCylinder(radius, depth, centre, 24)
cylinder.BooleanWithNetwork(network)
```

We visualize the smaller region

In [9]:
```
network.Write(file_handler.GetOutputDirectoryFullPath() + "cleaned_cut_network.vtp")
nb_manager.vtk_show(scene, height=600, width = 1000)
```

Out[9]:



We are ready to simulate tumour growth and angiogenesis. We will use a regular lattice for this purpose. We size and position the lattice according to the bounds of the vessel network.

In [10]:
```
network_bounding_box = [microvessel_chaste.mesh.DimensionalChastePoint3(1500.0, 1600.0, -10.0, 1.e-6*metre()),
                        microvessel_chaste.mesh.DimensionalChastePoint3(3100.0, 3000.0, 300.0, 1.e-6*metre())]
grid = microvessel_chaste.mesh.RegularGrid3()
grid_spacing = 40.0e-6* metre()
grid.SetSpacing(grid_spacing)
```

We can use the built-in dimensional analysis functionality to get the network extents in terms of grid units

In [11]:
```
botom_front_left =  network_bounding_box[0].GetLocation(grid_spacing)
top_back_right =  network_bounding_box[1].GetLocation(grid_spacing)
extents = top_back_right - botom_front_left
extents = [int(x)+1 for x in extents] # snap to the nearest unit, overestimate size if needed
grid.SetExtents(extents)
network.Translate(microvessel_chaste.mesh.DimensionalChastePoint3(-1500.0, -1600.0, +10.0, 1.e-6*metre()))
```

Next we set the inflow and outflow boundary conditions for blood flow. Because the network connectivity is relatively low we assign all vessels near the top of the domain (z coord) as inflows and the bottom as outflows.

In [12]:
```
for eachNode in network.GetNodes():
    if eachNode.GetNumberOfSegments() == 1:
        if abs(eachNode.rGetLocation().GetLocation(1.e-6*metre())[2] -
               network_bounding_box[1].GetLocation(1.e-6*metre())[2]) < 80.0:
            eachNode.GetFlowProperties().SetIsInputNode(True)
            eachNode.GetFlowProperties().SetPressure(Owen11Parameters.mpInletPressure.GetValue("User"))
        elif abs(eachNode.rGetLocation().GetLocation(1.e-6*metre())[2] -
                 network_bounding_box[0].GetLocation(1.e-6*metre())[2]) < 80.0:
            eachNode.GetFlowProperties().SetIsOutputNode(True);
            eachNode.GetFlowProperties().SetPressure(Owen11Parameters.mpOutletPressure.GetValue("User"))
```

Again, we can write the network to file for visualization

In [13]:
```
network.Write(file_handler.GetOutputDirectoryFullPath() + "flow_boundary_labelled_network.vtp")
```

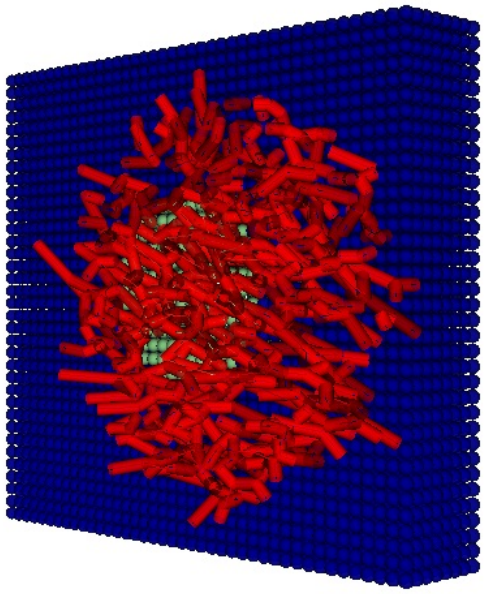
Next, set up the cell populations. We will setup up a population similar to that used in the Owen et al., 2011 paper. That is, a grid filled with normal cells and a tumour spheroid in the middle. We can use a generator for this purpose. The generator simply sets up the population using conventional Cell Based Chaste methods. It can take a few seconds to set up the population.

In [14]:
```
cell_population_genenerator = microvessel_chaste.population.cell.Owen11CellPopulationGenerator3()
cell_population_genenerator.SetRegularGrid(grid)
cell_population_genenerator.SetVesselNetwork(network)
tumour_radius = 300.0 * 1.e-6 * metre()
cell_population_genenerator.SetTumourRadius(tumour_radius)
cell_population = cell_population_genenerator.Update()
```

We can visualize the population. Note that we are reaching the limits of the browser based visualization at this point. The model can be better visualized in Paraview using the files we have been writing.

In [15]:
```
scene.SetCellPopulation(cell_population)
scene.GetCellPopulationActorGenerator().GetDiscreteColorTransferFunction().AddRGBPoint(1.0, 0.0, 0.0, 0.6)
scene.GetCellPopulationActorGenerator().SetPointSize(20)
scene.GetCellPopulationActorGenerator().SetColorByCellMutationState(True)
scene.ResetRenderer()
nb_manager.vtk_show(scene, height=600, width = 1000)
```

Out[15]:



Next set up the PDEs for oxygen and VEGF. Cells will act as discrete oxygen sinks and discrete vegf sources.

In [16]:
```
oxygen_pde = microvessel_chaste.pde.LinearSteadyStateDiffusionReactionPde3_3()
oxygen_pde.SetIsotropicDiffusionConstant(Owen11Parameters.mpOxygenDiffusivity.GetValue("User"))
cell_oxygen_sink = microvessel_chaste.pde.CellBasedDiscreteSource3()
cell_oxygen_sink.SetLinearInUConsumptionRatePerCell(Owen11Parameters.mpCellOxygenConsumptionRate.GetValue("User"))
oxygen_pde.AddDiscreteSource(cell_oxygen_sink)
```

Vessels release oxygen depending on their haematocrit levels

In [17]:
```
vessel_oxygen_source = microvessel_chaste.pde.VesselBasedDiscreteSource3()
#oxygen_solubility_at_stp = Secomb04Parameters.mpOxygenVolumetricSolubility.GetValue("User") * GenericParameters.mpGasConcentrationAtStp.GetValue("User")
#vessel_oxygen_concentration = oxygen_solubility_at_stp * Owen11Parameters.mpReferencePartialPressure.GetValue("User")
vessel_oxygen_concentration = 0.02768 * mole_per_metre_cubed()
vessel_oxygen_source.SetReferenceConcentration(vessel_oxygen_concentration)
vessel_oxygen_source.SetVesselPermeability(Owen11Parameters.mpVesselOxygenPermeability.GetValue("User"))
vessel_oxygen_source.SetReferenceHaematocrit(Owen11Parameters.mpInflowHaematocrit.GetValue("User"))
oxygen_pde.AddDiscreteSource(vessel_oxygen_source);
```

Set up a finite difference solver and pass it the pde and grid.

In [18]:
```
oxygen_solver = microvessel_chaste.pde.FiniteDifferenceSolver3()
oxygen_solver.SetPde(oxygen_pde)
oxygen_solver.SetLabel("oxygen")
oxygen_solver.SetGrid(grid)
```

The rate of VEGF release depends on the cell type and intracellular VEGF levels, so we need a more detailed type of discrete source.

In [19]:
```
vegf_pde = microvessel_chaste.pde.LinearSteadyStateDiffusionReactionPde3_3()
vegf_pde.SetIsotropicDiffusionConstant(Owen11Parameters.mpVegfDiffusivity.GetValue("User"))
vegf_pde.SetContinuumLinearInUTerm(-1.0 * Owen11Parameters.mpVegfDecayRate.GetValue("User"))
```

Set up a map for different release rates depending on cell type. Also include a threshold intracellular VEGF below which there is no release.

In [20]:
```
normal_and_quiescent_cell_source = microvessel_chaste.pde.CellStateDependentDiscreteSource3()
normal_and_quiescent_cell_rates = microvessel_chaste.pde.MapUnsigned_ConcentrationFlowRate()
normal_and_quiescent_cell_rate_thresholds = microvessel_chaste.pde.MapUnsigned_Concentration()
quiescent_cancer_state = microvessel_chaste.population.cell.QuiescentCancerCellMutationState()
normal_cell_state = chaste.cell_based.WildTypeCellMutationState()
normal_and_quiescent_cell_rates[normal_cell_state.GetColour()] = Owen11Parameters.mpCellVegfSecretionRate.GetValue("User")
normal_and_quiescent_cell_rate_thresholds[normal_cell_state.GetColour()] = 0.27*mole_per_metre_cubed()
normal_and_quiescent_cell_rates[quiescent_cancer_state.GetColour()] = Owen11Parameters.mpCellVegfSecretionRate.GetValue("User")
normal_and_quiescent_cell_rate_thresholds[quiescent_cancer_state.GetColour()] = 0.0*mole_per_metre_cubed()
normal_and_quiescent_cell_source.SetStateRateMap(normal_and_quiescent_cell_rates)
normal_and_quiescent_cell_source.SetLabelName("VEGF")
normal_and_quiescent_cell_source.SetStateRateThresholdMap(normal_and_quiescent_cell_rate_thresholds)
vegf_pde.AddDiscreteSource(normal_and_quiescent_cell_source)
```

Add a vessel related VEGF sink

In [21]:
```
vessel_vegf_sink = microvessel_chaste.pde.VesselBasedDiscreteSource3()
vessel_vegf_sink.SetReferenceConcentration(0.0*mole_per_metre_cubed())
vessel_vegf_sink.SetVesselPermeability(Owen11Parameters.mpVesselVegfPermeability.GetValue("User"))
vegf_pde.AddDiscreteSource(vessel_vegf_sink)
```

Set up a finite difference solver as before.

In [22]:
```
vegf_solver = microvessel_chaste.pde.FiniteDifferenceSolver3()
vegf_solver.SetPde(vegf_pde)
vegf_solver.SetLabel("VEGF_Extracellular")
vegf_solver.SetGrid(grid)
```

Next set up the flow problem. Assign a blood plasma viscosity to the vessels. The actual viscosity will depend on haematocrit and diameter. This solver manages growth and shrinkage of vessels in response to flow related stimuli.

In [23]:
```
large_vessel_radius = 25.0e-6 * metre()
network.SetSegmentRadii(large_vessel_radius)
viscosity = Owen11Parameters.mpPlasmaViscosity.GetValue("User")
network.SetSegmentViscosity(viscosity);
```

Set up the pre- and post flow calculators.

In [24]:
```
impedance_calculator = microvessel_chaste.simulation.VesselImpedanceCalculator3()
haematocrit_calculator = microvessel_chaste.simulation.ConstantHaematocritSolver3()
haematocrit_calculator.SetHaematocrit(Owen11Parameters.mpInflowHaematocrit.GetValue("User"))
wss_calculator = microvessel_chaste.simulation.WallShearStressCalculator3()
mech_stimulus_calculator = microvessel_chaste.simulation.MechanicalStimulusCalculator3()
metabolic_stim_calculator = microvessel_chaste.simulation.MetabolicStimulusCalculator3()
shrinking_stimulus_calculator = microvessel_chaste.simulation.ShrinkingStimulusCalculator3()
viscosity_calculator = microvessel_chaste.simulation.ViscosityCalculator3()
```

Set up and configure the structural adaptation solver.

In [25]:
```
structural_adaptation_solver = microvessel_chaste.simulation.StructuralAdaptationSolver3()
structural_adaptation_solver.SetTolerance(0.0001)
structural_adaptation_solver.SetMaxIterations(100)
structural_adaptation_solver.SetTimeIncrement(Owen11Parameters.mpVesselRadiusUpdateTimestep.GetValue("User"));
structural_adaptation_solver.AddPreFlowSolveCalculator(impedance_calculator)
structural_adaptation_solver.AddPostFlowSolveCalculator(haematocrit_calculator)
structural_adaptation_solver.AddPostFlowSolveCalculator(wss_calculator)
structural_adaptation_solver.AddPostFlowSolveCalculator(metabolic_stim_calculator)
structural_adaptation_solver.AddPostFlowSolveCalculator(mech_stimulus_calculator)
structural_adaptation_solver.AddPostFlowSolveCalculator(viscosity_calculator)
```

Set up a regression solver.

In [26]:
```
regression_solver = microvessel_chaste.simulation.WallShearStressBasedRegressionSolver3()
```

Set up an angiogenesis solver and add sprouting and migration rules.

In [27]:
```
angiogenesis_solver = microvessel_chaste.simulation.AngiogenesisSolver3()
sprouting_rule = microvessel_chaste.simulation.OffLatticeSproutingRule3()
sprouting_rule.SetSproutingProbability(1.e-5*per_second())
migration_rule = microvessel_chaste.simulation.OffLatticeMigrationRule3()
migration_rule.SetChemotacticStrength(0.1)
migration_rule.SetAttractionStrength(0.5)
migration_rule.SetSproutingVelocity((40.0*1.e-6/3600.0)*metre_per_second())
angiogenesis_solver.SetMigrationRule(migration_rule)
angiogenesis_solver.SetSproutingRule(sprouting_rule)
sprouting_rule.SetDiscreteContinuumSolver(vegf_solver)
migration_rule.SetDiscreteContinuumSolver(vegf_solver)
angiogenesis_solver.SetVesselNetwork(network)
```

The microvessel solver will manage all aspects of the vessel solve.

In [28]:
```
microvessel_solver = microvessel_chaste.simulation.MicrovesselSolver3()
microvessel_solver.SetVesselNetwork(network)
microvessel_solver.SetOutputFrequency(1)
microvessel_solver.AddDiscreteContinuumSolver(oxygen_solver)
microvessel_solver.AddDiscreteContinuumSolver(vegf_solver)
microvessel_solver.SetStructuralAdaptationSolver(structural_adaptation_solver)
microvessel_solver.SetRegressionSolver(regression_solver)
microvessel_solver.SetAngiogenesisSolver(angiogenesis_solver)
```

The microvessel solution modifier will link the vessel and cell solvers. We need to explicitly tell is which extracellular fields to update based on PDE solutions.

In [29]:
```
microvessel_modifier = microvessel_chaste.simulation.MicrovesselSimulationModifier3()
microvessel_modifier.SetMicrovesselSolver(microvessel_solver)
update_labels = microvessel_chaste.simulation.VecString()
update_labels.append("oxygen")
update_labels.append("VEGF_Extracellular")
microvessel_modifier.SetCellDataUpdateLabels(update_labels)
```

Set up plotting

In [30]:
```
scene.GetCellPopulationActorGenerator().SetColorByCellData(True)
scene.GetCellPopulationActorGenerator().SetDataLabel("oxygen")
scene_modifier = microvessel_chaste.visualization.JupyterMicrovesselSceneModifier3(nb_manager)
scene_modifier.SetVtkScene(scene)
scene_modifier.SetUpdateFrequency(1)
microvessel_solver.AddMicrovesselModifier(scene_modifier)
```

The full simulation is run as a typical Cell Based Chaste simulation

In [31]:
```
simulator = chaste.cell_based.OnLatticeSimulation3(cell_population)
simulator.AddSimulationModifier(microvessel_modifier)
```

Add a killer to remove apoptotic cells

In [32]:
```
apoptotic_cell_killer = chaste.cell_based.ApoptoticCellKiller3(cell_population)
simulator.AddCellKiller(apoptotic_cell_killer)
```

Add another modifier for updating cell cycle quantities.

In [33]:
```
owen11_tracking_modifier = microvessel_chaste.simulation.Owen2011TrackingModifier3()
simulator.AddSimulationModifier(owen11_tracking_modifier)
```

Set up the remainder of the simulation

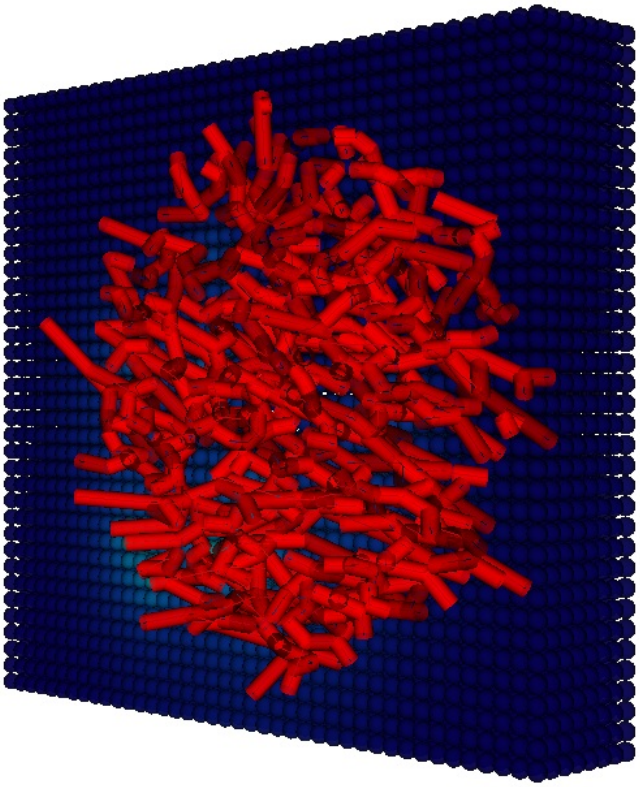
In [34]:
```
simulator.SetOutputDirectory("Python/TestBiologicalNetworkLiteratePaper")
simulator.SetSamplingTimestepMultiple(1)
simulator.SetDt(0.5)
```

This end time corresponds to roughly 10 minutes run-time on a desktop PC. Increase it or decrease as preferred. The end time used in Owen et al. 2011 is 4800 hours.

In [35]:
```
simulator.SetEndTime(2.0)
```

Do the solve. A sample solution is shown at the top of this test.

In [36]:
```
simulator.Solve()
```



Dump the parameters to file for inspection.

In [37]:
```
ParameterCollection.Instance().DumpToFile(file_handler.GetOutputDirectoryFullPath()+"parameter_collection.xml")
nb_manager.add_parameter_table(file_handler)
# Tear down the test
chaste.cell_based.TearDownNotebookTest()
```

name	value	symbol	added_by	descrip
				Gas

Generic_GasConcentrationAtStp	44.6429 m^-3 mol	C_{stp}	Owen2011OxygenBasedCellCycleOdeSystem	concent at STP
Owen11_BasalMetabolicStimulus	1.7 Hz	k_m^0	MetabolicStimulusCalculator	Basal metabol stimulus
Owen11_CellMotilityCancer	8.33333e-15 m^2 s^-1	D_{cancer}	Owen11CaUpdateRule	Maximu cell mot cancer
Owen11_CellOxygenConsumptionRate	0.216667 Hz	k_c^{cell}	User	Cell oxy consum rate
Owen11_CellVegfProductionRate	3.33333e-05 Hz	k_8	Owen2011OxygenBasedCellCycleOdeSystem	Basal V producti rate in c
Owen11_CellVegfSecretionRate	1.66667e-13 m^-3 s^-1 mol	k_v^{cell}	User	Cell veg secretio rate
Owen11_CriticalWallShearStress	0.8 Pa	τ_{wall}	WallShearStressBasedRegressionSolver	Critical shear st for vess pruning
Owen11_InflowHaematocrit	0.45 dimensionless	H_{in}	User	Inflow haemat
Owen11_InletPressure	3333.05 Pa	P_{in}	User	Vessel network pressur
Owen11_MaxCellVegfProductionRate	0.000166667 Hz	k_{8*}	Owen2011OxygenBasedCellCycleOdeSystem	Max VE producti rate in c
Owen11_MaxTimeWithLowWallShearStress	240000 s	T_{prune}	WallShearStressBasedRegressionSolver	Maximu vessel survivia with low shear st
Owen11_MaximumRadius	5e-05 m	R_{MAX}	RadiusCalculator	Maximu possible radius
Owen11_MaximumSproutingRate	4.16667e-06 Hz	P_{sprout}^{max}	Owen2011SproutingRule	Maximu rate of sproutin
Owen11_MinCellCycleCancer	96000 s	T_{min}^{cancer}	Owen2011OxygenBasedCellCycleOdeSystem	Minimur cycle pe cancer
Owen11_MinCellCycleNormal	180000 s	T_{min}^{normal}	Owen2011OxygenBasedCellCycleOdeSystem	Minimur cycle pe normal
Owen11_MinimumRadius	1e-06 m	R_{MIN}	RadiusCalculator	Minimur possible radius
Owen11_OutletPressure	1999.83 Pa	P_{out}	User	Vessel network outlet pressur
				Oxygen

Owen11_OxygenAtHalfMaxCycleRateCancer	186.651 Pa	C^{cancer}	Owen2011OxygenBasedCellCycleOdeSystem	partial pressur half ma cycle ra cancer
Owen11_OxygenAtHalfMaxCycleRateNormal	399.966 Pa	C^{normal}	Owen2011OxygenBasedCellCycleOdeSystem	Oxygen partial pressur half ma cycle ra normal
Owen11_OxygenAtQuiescence	1186.57 Pa	C^{enter}_{quiesc}	Owen2011OxygenBasedCellCycleModel	Oxygen partial pressur quiesce
Owen11_OxygenDiffusivity	2.41667e-09 m^2 s^-1	D_c	User	Oxygen diffusivi
Owen11_OxygenLeaveQuiescence	1306.56 Pa	C^{leave}_{quiesc}	Owen2011OxygenBasedCellCycleModel	Oxygen partial pressur leave quiesce
Owen11_OxygenTensionForHalfMaxP53Degradation	591.95 Pa	C_{p53}	Owen2011OxygenBasedCellCycleOdeSystem	Tissue oxygen tension half-ma degrada
Owen11_OxygenTensionForHalfMaxVegfDegradation	591.95 Pa	C_{VEGF}	Owen2011OxygenBasedCellCycleOdeSystem	Tissue oxygen tension half-ma degrada
Owen11_P53EffectOnVegfProduction	-3.33333e-05 Hz	k_{8**}	Owen2011OxygenBasedCellCycleOdeSystem	Effect o on VEG producti
Owen11_P53MaxDegradationRate	0.000166667 Hz	k_{*7}	Owen2011OxygenBasedCellCycleOdeSystem	Max p5 degrada rate
Owen11_P53ProductionRateConstant	3.33333e-05 Hz	k_7	Owen2011OxygenBasedCellCycleOdeSystem	Intracell p53 producti rate cor
Owen11_PlasmaViscosity	0.0012 m^-1 kg s^-1	μ_{plasma}	ViscosityCalculator	Blood plasma viscosity
Owen11_ReferenceFlowRateForMetabolicStimulus	6.66667e-13 m^3 s^-1	Q_{ref}	MetabolicStimulusCalculator	Referen flow rate metabol stimulus
Owen11_SensitivityToIntravascularPressure	0.5 Hz	k_p	MechanicalStimulusCalculator	Shrinkir intravas pressur
Owen11_ShrinkingTendency	1.7 Hz	k_s	ShrinkingStimulusCalculator	Shrinkir tendenc
Owen11_TimeDeathQuiescence	240000 s	T_{death}	Owen2011OxygenBasedCellCycleModel	Time fo death d sustaine

				quiesce
Owen11_VegfConventrationAtHalfMaxProbSprouting	5e-10 m^-3 mol	V_{sprout}	Owen2011SproutingRule	VEGF concent at half maxima vessel sproutin probabil
Owen11_VegfDecayRate	0.000166667 Hz	δ_v	User	Vegf de rate
Owen11_VegfDiffusivity	1.66667e-11 m^2 s^-1	D_v	User	Vegf diffusivi
Owen11_VegfEffectOnVegfProduction	0.04 dimensionless	j_5	Owen2011OxygenBasedCellCycleOdeSystem	Effect o VEGF o VEGF producti
Owen11_VesselOxygenPermeability	0.001 m s^-1	ψ_c	User	Vessel permeal to oxyge
Owen11_VesselRadiusUpdateTimestep	0.1 s	$_t$	User	Vessel r update timestep
Owen11_VesselVegfPermeability	1.66667e-09 m s^-1	ψ_v	User	Vessel permeal to vegf
Secomb04_OxygenVolumetricSolubility	2.3252e-07 m kg^-1 s^2	α_{eff}	Owen2011OxygenBasedCellCycleOdeSystem	Oxygen solubility

In [ ]:

This tutorial is automatically generated from the file test/python/tutorials//TestPythonOffLatticeAngiogenesisLiteratePaper.py.

In [1]:
```
# Jupyter notebook specific imports
import matplotlib as mpl
from IPython import display
%matplotlib inline
```

An Off Lattice Angiogenesis Tutorial

This tutorial demonstrates functionality for modelling 3D off-lattice angiogenesis in a corneal micro pocket application, similar to that described in Connor et al. 2015 (http://rsif.royalsocietypublishing.org/content/12/110/20150546.abstract).

It is a 3D simulation modelling VEGF diffusion and decay from an implanted pellet using finite element methods and lattice-free angiogenesis from a large limbal vessel towards the pellet.

The Test

In [2]:
```
import numpy as np
import chaste # Core Chaste functionality
import chaste.cell_based # Chaste Cell Populations
chaste.init() # Initialize MPI and PETSc
import microvessel_chaste # Core Microvessel Chaste functionality
import microvessel_chaste.geometry # Geometry tools
import microvessel_chaste.mesh # Meshing
import microvessel_chaste.population.vessel # Vessel tools
import microvessel_chaste.pde # PDE and solvers
import microvessel_chaste.simulation # Flow and angiogenesis solvers
import microvessel_chaste.visualization # Visualization
from microvessel_chaste.utility import * # Dimensional analysis: bring in all units for convenience
# Set up the test
chaste.cell_based.SetupNotebookTest()
```

Set up output file management.

In [3]:
```
file_handler = chaste.core.OutputFileHandler("Python/TestOffLatticeAngiogenesisLiteratePaper")
chaste.core.RandomNumberGenerator.Instance().Reseed(12345)
```

This component uses explicit dimensions for all quantities, but interfaces with solvers which take non-dimensional inputs. The BaseUnits singleton takes time, length and mass reference scales to allow non-dimensionalisation when sending quantities to external solvers and re-dimensionalisation of results. For our purposes microns for length and hours for time are suitable base units.

In [4]:
```
reference_length = 1.e-6 * metre()
reference_time = 3600.0 * second()
reference_concentration = 1.e-9*mole_per_metre_cubed()
BaseUnits.Instance().SetReferenceLengthScale(reference_length)
BaseUnits.Instance().SetReferenceTimeScale(reference_time)
BaseUnits.Instance().SetReferenceConcentrationScale(reference_concentration)
```

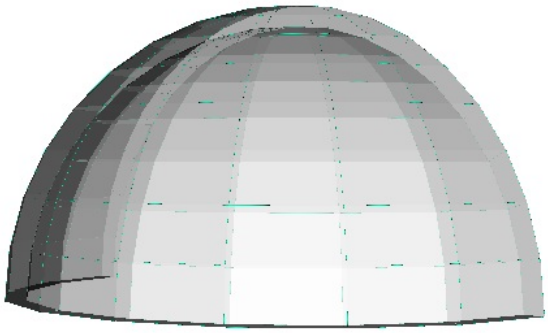
Set up the domain representing the cornea. This is a thin hemispherical shell. We assume some symmetry to reduce computational expense.

In [5]:
```
hemisphere_generator = microvessel_chaste.geometry.MappableGridGenerator()
radius = 1400.0e-6*metre()
thickness = 100.0e-6*metre()
num_divisions_x = 10
num_divisions_y = 10
azimuth_angle = 1.0 * np.pi
polar_angle = 0.5 * np.pi
cornea = hemisphere_generator.GenerateHemisphere(radius/reference_length,
                                                 thickness/reference_length,
                                                 num_divisions_x,
                                                 num_divisions_y,
                                                 azimuth_angle,
                                                 polar_angle)
```

We can visualize the part

In [6]:
```
scene = microvessel_chaste.visualization.MicrovesselVtkScene3()
scene.SetPart(cornea)
scene.GetPartActorGenerator().SetVolumeOpacity(0.7)
scene.GetPartActorGenerator().SetVolumeColor((255.0, 255.0, 255.0))
nb_manager = microvessel_chaste.visualization.JupyterNotebookManager()
nb_manager.vtk_show(scene, height=600, width = 1000)
```

Out[6]:



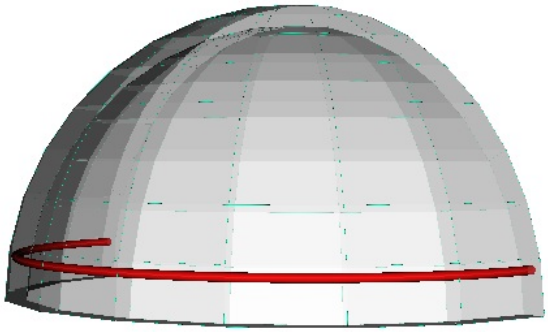
Set up a vessel network, with divisions roughly every 'cell length'. Initially it is straight. We will map it onto the hemisphere.

In [7]:
```
network_generator = microvessel_chaste.population.vessel.VesselNetworkGenerator3()
vessel_length = np.pi * radius
cell_length = 40.0e-6 * metre()
origin = microvessel_chaste.mesh.DimensionalChastePoint3(0.0, 4000.0, 0.0)
network  = network_generator.GenerateSingleVessel(vessel_length, origin, int(float(vessel_length/cell_length)) + 1, 0)
network.GetNode(0).GetFlowProperties().SetIsInputNode(True);
network.GetNode(0).GetFlowProperties().SetPressure(Owen11Parameters.mpInletPressure.GetValue("User"))
network.GetNode(network.GetNumberOfNodes()-1).GetFlowProperties().SetIsOutputNode(True)
network.GetNode(network.GetNumberOfNodes()-1).GetFlowProperties().SetPressure(Owen11Parameters.mpOutletPressure.GetValue(
"User"))
nodes = network.GetNodes();
for eachNode in nodes:
    node_azimuth_angle = float(azimuth_angle * eachNode.rGetLocation().GetLocation(reference_length)[0]*reference_length/vessel_len
gth)
    node_polar_angle = float(polar_angle*eachNode.rGetLocation().GetLocation(reference_length)[1]*reference_length/vessel_length)
    dimless_radius = (float(radius/reference_length)+(-0.5*float(thickness/reference_length)))
    new_position = microvessel_chaste.mesh.DimensionalChastePoint3(dimless_radius * np.cos(node_azimuth_angle) * np.sin(node_pol
ar_angle),
                                                   dimless_radius * np.cos(node_polar_angle),
                                                   dimless_radius * np.sin(node_azimuth_angle) * np.sin(node_polar_angle),
                                                   reference_length)
    eachNode.SetLocation(new_position)
```

Visualize the network

In [8]:
```
scene.SetVesselNetwork(network)
scene.GetVesselNetworkActorGenerator().SetEdgeSize(20.0)
nb_manager.vtk_show(scene, height=600, width = 1000)
```

Out[8]:



In the experimental assay a pellet containing VEGF is implanted near the top of the cornea. We model this as a fixed concentration of VEGF in a cuboidal region. First set up the vegf sub domain.

In [9]:
```
pellet = microvessel_chaste.geometry.Part3()
pellet_side_length = 300.0e-6 * metre()
origin = microvessel_chaste.mesh.DimensionalChastePoint3(-150.0,900.0,0.0)
pellet.AddCuboid(pellet_side_length, pellet_side_length, 5.0*pellet_side_length, origin)
pellet.Write(file_handler.GetOutputDirectoryFullPath()+"initial_vegf_pellet.vtp",
             microvessel_chaste.geometry.GeometryFormat.VTP)
```

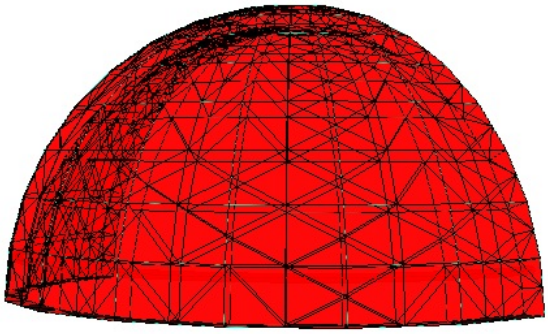
Now make a finite element mesh on the cornea.

In [10]:
```
mesh_generator = microvessel_chaste.mesh.DiscreteContinuumMeshGenerator3_3()
mesh_generator.SetDomain(cornea)
mesh_generator.SetMaxElementArea(1e-6 * metre_cubed())
mesh_generator.Update()
mesh = mesh_generator.GetMesh()
```

We can visualize the mesh

In [11]:
```
scene.GetPartActorGenerator().SetVolumeOpacity(0.0)
scene.SetMesh(mesh)
nb_manager.vtk_show(scene, height=600, width = 1000)
```

Out[11]:



Set up the vegf pde. Note the scaling of the refernece concentration to nM to avoid numerical precision problems.

In [12]:
```
vegf_pde = microvessel_chaste.pde.LinearSteadyStateDiffusionReactionPde3_3()
vegf_pde.SetIsotropicDiffusionConstant(Owen11Parameters.mpVegfDiffusivity.GetValue("User"))
vegf_pde.SetContinuumLinearInUTerm(-1.0*Owen11Parameters.mpVegfDecayRate.GetValue("User"))
vegf_pde.SetMesh(mesh)
vegf_pde.SetUseRegularGrid(False)
vegf_pde.SetReferenceConcentration(1.e-9*mole_per_metre_cubed())
```

Add a boundary condition to fix the VEGF concentration in the vegf subdomain.

In [13]:
```
vegf_boundary = microvessel_chaste.pde.DiscreteContinuumBoundaryCondition3()
vegf_boundary.SetType(microvessel_chaste.pde.BoundaryConditionType.IN_PART)
vegf_boundary.SetSource(microvessel_chaste.pde.BoundaryConditionSource.PRESCRIBED)
vegf_boundary.SetValue(3.e-9*mole_per_metre_cubed())
vegf_boundary.SetDomain(pellet)
```

Set up the PDE solvers for the vegf problem.

In [14]:
```
vegf_solver = microvessel_chaste.pde.FiniteElementSolver3()
vegf_solver.SetPde(vegf_pde)
vegf_solver.SetLabel("vegf")
vegf_solver.SetMesh(mesh)
vegf_solver.AddBoundaryCondition(vegf_boundary)
```

Set up an angiogenesis solver and add sprouting and migration rules.

In [15]:
```
angiogenesis_solver = microvessel_chaste.simulation.AngiogenesisSolver3()
sprouting_rule = microvessel_chaste.simulation.OffLatticeSproutingRule3()
sprouting_rule.SetSproutingProbability(1.e6* per_second())
migration_rule = microvessel_chaste.simulation.OffLatticeMigrationRule3()
migration_rule.SetChemotacticStrength(0.1)
migration_rule.SetAttractionStrength(0.5)
sprout_velocity = (50.0e-6/(24.0*3600.0))*metre_per_second() #Secomb13
migration_rule.SetSproutingVelocity(sprout_velocity)
angiogenesis_solver.SetMigrationRule(migration_rule)
angiogenesis_solver.SetSproutingRule(sprouting_rule)
sprouting_rule.SetDiscreteContinuumSolver(vegf_solver)
migration_rule.SetDiscreteContinuumSolver(vegf_solver)
angiogenesis_solver.SetVesselNetwork(network)
angiogenesis_solver.SetBoundingDomain(cornea)
```

Set up the MicrovesselSolver which coordinates all solves.

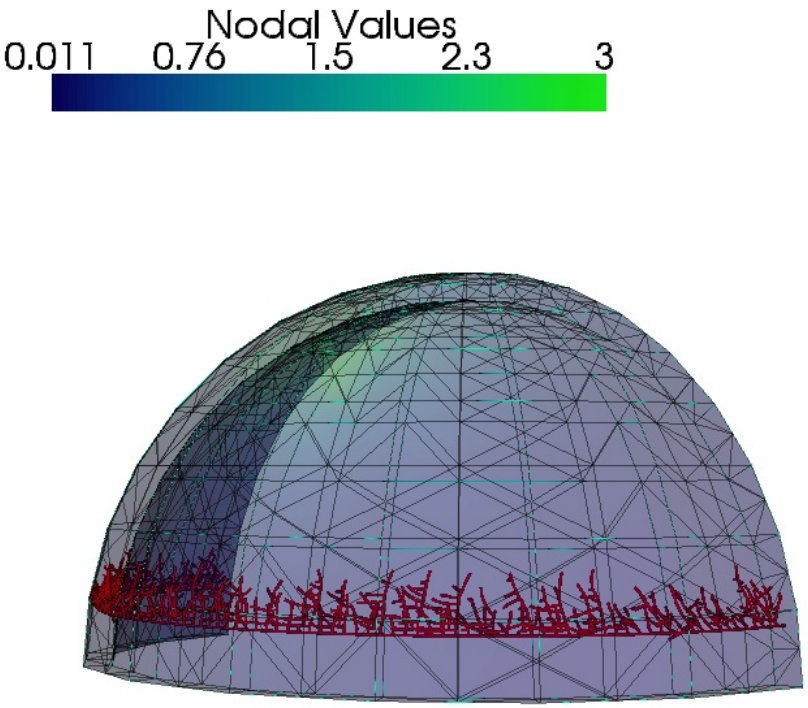
In [16]:
```
microvessel_solver = microvessel_chaste.simulation.MicrovesselSolver3()
microvessel_solver.SetVesselNetwork(network)
microvessel_solver.AddDiscreteContinuumSolver(vegf_solver)
microvessel_solver.SetOutputFileHandler(file_handler)
microvessel_solver.SetOutputFrequency(5)
microvessel_solver.SetAngiogenesisSolver(angiogenesis_solver)
microvessel_solver.SetUpdatePdeEachSolve(False)
```

Set up plotting

In [17]:
```
scene.GetDiscreteContinuumMeshActorGenerator().SetVolumeOpacity(0.3)
scene.GetDiscreteContinuumMeshActorGenerator().SetDataLabel("Nodal Values")
scene.GetVesselNetworkActorGenerator().SetEdgeSize(5.0)
scene_modifier = microvessel_chaste.visualization.JupyterMicrovesselSceneModifier3(nb_manager)
scene_modifier.SetVtkScene(scene)
scene_modifier.SetUpdateFrequency(2)
microvessel_solver.AddMicrovesselModifier(scene_modifier)
```

Set the simulation time and run the solver.

In [18]:
```
chaste.cell_based.SimulationTime.Instance().SetEndTimeAndNumberOfTimeSteps(100.0, 10)
microvessel_solver.Run()
```



Dump the parameters to file for inspection.

In [19]:
```
ParameterCollection.Instance().DumpToFile(file_handler.GetOutputDirectoryFullPath()+"parameter_collection.xml")
nb_manager.add_parameter_table(file_handler)
# Tear down the test
chaste.cell_based.TearDownNotebookTest()
```

name	value	symbol	added_by	description
Owen11_InletPressure	3333.05 Pa	P_{in}	User	Vessel network inlet pressure$
Owen11_MaximumSproutingRate	4.16667e-06 Hz	P^{max}_{sprout}	Owen2011SproutingRule	Maximum rate of sprouting
Owen11_OutletPressure	1999.83 Pa	P_{out}	User	Vessel network outlet pressure
Owen11_VegfConventrationAtHalfMaxProbSprouting	5e-10 m^-3 mol	V_{sprout}	Owen2011SproutingRule	VEGF concentration at half maximal vessel sprouting probability
Owen11_VegfDecayRate	0.000166667 Hz	δ_v	User	Vegf decay rate
Owen11_VegfDiffusivity	1.66667e-11 m^2 s^-1	D_v	User	Vegf diffusivity

In [ ]: