# Tumorcode - A framework to simulate vascularized tumors

Thierry Fredrich[1], Michael Welter[1], Heiko Rieger[1*]

**1** Theoretical physics, Department for natural sciences and technology, Saarland University, Germany
* h.rieger@mx.uni-saarland.de

## Abstract

During the past years our group published several articles exploiting computer simulations to address the complex interaction of tumors and the vasculature as underlying transport network. Considerable advances in imaging and lab techniques pushed in vitro research of tumor spheroids, animal models and clinical studies which already provided more insights to single processes taking part in tumor growth. Computer simulation is a none-invasive option to put the acquired insights together and simulate a quasi in vivo system. To this end, several known models are put together to construct a multi-scale approach which advances towards length scales relevant for clinical applications.

A method to construct synthetic arterio- venous blood vessel networks in 3D plays a key role in our simulations. Justification for this algorithm and the biological models was obtained in previous publications where details about the implementation and the technical aspects were spared. In this manuscript we describe the software used to draw conclusions about elevated interstitial fluid pressure, drug and oxygen distribution in vascularized tumors. By releasing our code to the public domain, the details regarding the implementation, technical descriptions and the examples provided by this manuscript become relevant for the community willing to work with our code.

## Author Summary

## Introduction

Vascular networks play an important role in therapy and progression of cancer. In particular their formation and modifications are a special feature of most malignant tumors [1]. In the case of solid tumors, the growth of cancer cells is facilitated by the diffusion of nutrients within the vicinity of the cells. Growing beyond the average diffusion length of nutrients, new strategies are needed by tumors to acquire nutrients. By transforming the present blood vessels, the tumor exploits them for his own purposes. The full complexity of this process is not yet understood.

Since the vasculature mediates drugs to cancer cells, this transformation is certainly of interest for chemotherapy. Also radio therapy is known to be dependent on the oxygen level inside the tissue which is again dependent on the vessel morphology and vessel wall properties. The intertwined dependence of solute transport or solute distribution and spatial arrangement of the blood vessels is therefore of great importance in every cancer therapy.

Obtaining experimental or clinical data of sufficiently high resolution to reconstruct vascular networks is by itself a hard task [2]. To overcome the lack of data, in silico

modeling of vasculature is done by various authors. In [3] an off-lattice, agent-based model is used to successfully construct vascular structures of around $1mm$ in size. Others also focus on the extra cellular matrix [4], but most studies focusing on structural aspects neglect blood transport inside the vessels which is essential for clinical applications. Well established models consider the blood flow but are restricted to two dimensions only [5]. Fewer are done in 3D [6]. Apart from blood vessel network construction, the growth of tumors is of key interest in research. The implementation strategies for tumor growth fall into two categories. First: discrete, particle based approaches where each cancerous cell is modeled individually [7]. Naturally this takes considerable computational efforts providing greater details on the cell level. Second: models where the tumor is described by continuous equations, reaching dimensions relevant for diagnostics [8]. Nevertheless, it is mostly neglected that tumors in patients or animal models grow within well-vascularized environments which means the original or initial network is not adequately represented. Another problem in the field is that the code is maintained and used by corresponding groups only, rather than sharing implementations which would simply save time. Repositories for open source code are well known in computer science and started to become useful in life science as well [9] [10].

To create blood vessel networks, we use a lattice-based attempt. Discretizing the angular freedom enables us to explore clinical relevant dimensions. To the best of our knowledge there is no other software constructing synthetic arterio-venous blood vessel networks on macroscopic tissue volumes in 3 dimension matching topological, morphological and hydrodynamic properties observed in real samples. The presented multi-scale approach describes tumor growth by partial differential equations which computationally allows us to go to system sizes in the order of cm. Subsequent, the interpretation of clinical studies, animal models and testing of hypotheses on biological processes is feasible. Our attempt considers topological and functional aspects at the same time which is relevant for therapeutics. To foster the understanding of malignant tumors addresses a broad community within life science. Therefore we commented, structured and documented the developed code to render it useful and accessible to other scientist working in this field.

# Design and implementation

Like modern implementations we reassemble two most distinct aspects. On the one hand we use *python* accompanied by the *scipy* packages for an easy and straight forward scripting of scientific problems. On the other hand, computationally expensive tasks are delegated to a core written in *C++* for performance reasons. Following up, a pythonic interface to this core has been created with the help of *Boost.Python*. We decided to use the *'hdf5'*- file format [11] to store data since it is highly scale-able and supported by both programming languages used.

## Computational core

The source code is commented in a standard way such that the provided *doxygen* file creates information about the dependencies within the object-oriented kernel. Further information can be found in the subfolder *'doc'*.

Demanding main applications for the core are calculating the hydro- and hemodynamics and solving partial differential equations where either of this tasks is ascribed to solving systems of sparse linear equations. The package *Trilinos* described in [12] is very general and sophisticated enough to fulfill our requirements.

### Hydro- and hemodynamics

Before determining properties of the network, the topology itself is created by the vesselgenerator (see folder *'src/common/vesselgen'*). The idea follows a 3 dimensional generalization of the algorithm presented in [13]. An example output can be found in Fig 2 and a complete description of the algorithm is provided in [2, section 2.1]. Once the radii are fixed, we assume mass conservation at each vertex to get a matrix equation analogues to a Kirchhoff circuit in electrodynamics. As boundary condition, one can either use a fixed pressure or fixed flow at any of the networks inlets and outlets. Fåhræus-Lindqvist and phase separation effect are considered accordingly.

### Tumor growth

The bulk tissue tumor model is based on a framework developed in [8], where the tissue is described as a mixture of various components. Our implementation includes normal, tumor and necrotic phases. The levelset-method is used to distinguish the different phases and define the interface between them. Assuming incompressibility, one can describe the composition in terms of volume fractions which leads to a system of partial differential equations of diffusion- convection- reaction type for each phase. Introducing a regular cubic lattice for bulk tissue allows to solve these equations by Finite Difference techniques. Consequently, the numerical resolution of *tumorcode* is governed by the lattice constant chosen. The number of grid points and vessels is limited by the provided hardware memory. Note that the bulk tissue and vascular network coexist in the same space, but different lattices were used for discretization. To fully utilize modern computing equipment we parallelized our code with the shared memory API of OpenMP. More details about this methodology could be found in the well written article [14].

### Finite Elements Method

In [15] we showed that the equation for the vascular oxygen distribution together with the diffusion equation for tissue form a complicated nonlinear system of equations with the partial pressure values as unknowns. For the longitudinal transport a bisection search is used to gather the partial oxygen pressure values within each vessel. Constructing consistent expressions for the transvascular exchange of oxygen across the vessel lumen was a challenging task, but enabled us to use the exchange as source or sink for the diffusion equation of the tissue oxygen field. Standard Galerkin methods were used to translate the task of solving the diffusion equation into the solving sparse linear equations. To this end the algebraic multigrid methods implemented in *Trilinos* were appropriate. Alternating the solver for vascular oxygen transport and the solver for the tissue oxygen results in a stable self consistent equilibrium oxygen distribution.

## Python API

The Input Interface is completely written in *python*. Note that necessary parameters are stored as python dictionaries.

How the single calculations are invoked and executed is presented in appendix B.1 to B.5. A typical work flow is depicted in Figure 1. In general, one starts by creating synthetic blood vessels which function as host and transport network (see ). This step is followed by the simulation of tumor growth including vascular remodeling. Fig 3 shows a snapshot of the simulation carried out in appendix B.3. Every simulation outputs its state at time intervals specified by the parameters into a file where the groups are named out0000, out0001, etc. Upon creating, one of the detailed biology models can be applied. Finally, we offer a large number of scripts that we use to analyze raw simulation data.

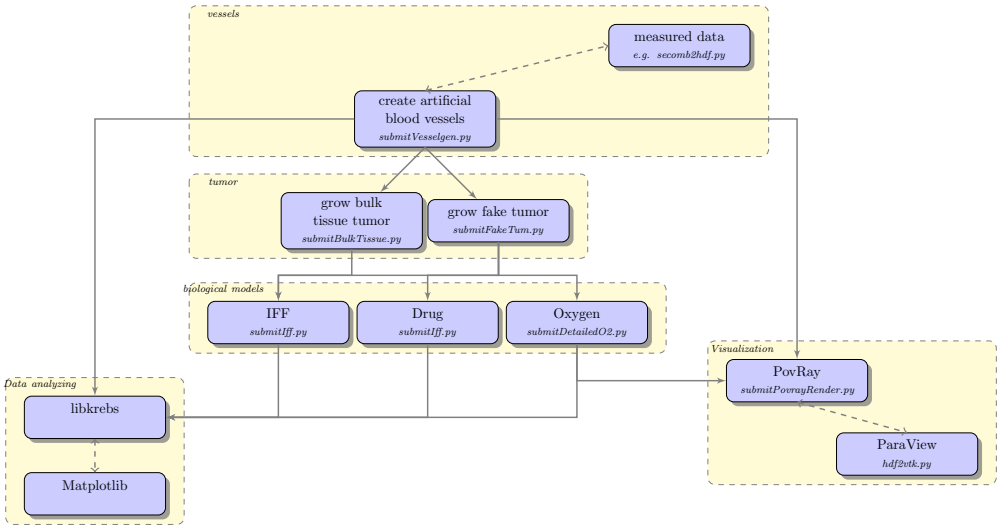


**Fig 1. Workflow of *tumorcode***
Dotted lines indicate that connected modules can be interchanged. The straight arrows must be followed in the direction indicated by the arrow head.

## Launching simulations

After successful installation, the main tools described in this section can be found in the *'bin'* folder. *tumorcode* is ready to use on computing clusters equipped with a queuing system. To remember the fact that the computation will be submitted to a queuing system, the name of the simulation programs start with *submit*. Currently we support *'PBS (Portable Batch System)'* and *'Slurm Workload Manager'*. Job details such as expected runtime and allocated memory could be provided as optional commandline arguments. If no supported queuing system is present or the flag (*–q-local*) is invoked, the computation runs locally. Each submit scripts expect the user to supply at least the name of a parameter set as command line argument. The help flag (*-h*) reminds the user that the parameter sets can be found in the folder *'py/krebsjobs/parameters'* and illustrates the structure of commandline calls.

The file storing an artificial vasculature has a distinctive structure. To present the data structure of a vessel file, a minimal working example is created. It contains 3 arteries, 3 veins and 2 capillaries forming a loop. In principal each vessel is defined as an edge linking two nodes. Details can be discerned from Fig 4 as well as from the complete source stored in *'py/tests/simpleVesselConfig.py'*. The hemodynamic and morphological data calculated throughout the creation process as described in Hydro- and hemodynamics is stored for each vessel within the edges directory, e.g flow, radius and hematocrit. The flag (*'–twoD'*) is of particular interest here. If selected, the creation process is restricted to a plane. All other programs are in full 3D.

After creation of a vessel system, one has to pass this file together with a parameter set to the next step in the workflow, the tumor stage. The program *'submitBulkTissueTumor'* runs simulations using the algorithm from the core discussed in Tumor growth. For explorative studies, a fake tumor calculation which neglects the full dynamics of the cells and represents a tumor as spherically expanding mass of cells at a constant speed is implemented. Still the remodeling of the vasculature via vessel cooption, regression, and growth takes place. This allows to proceed faster from the tumor step to the biological models. This variant is invoked by *'submitFakeTum'*.

After this stage, a *'hdf5'*-file containing snapshots of either tumor simulation should

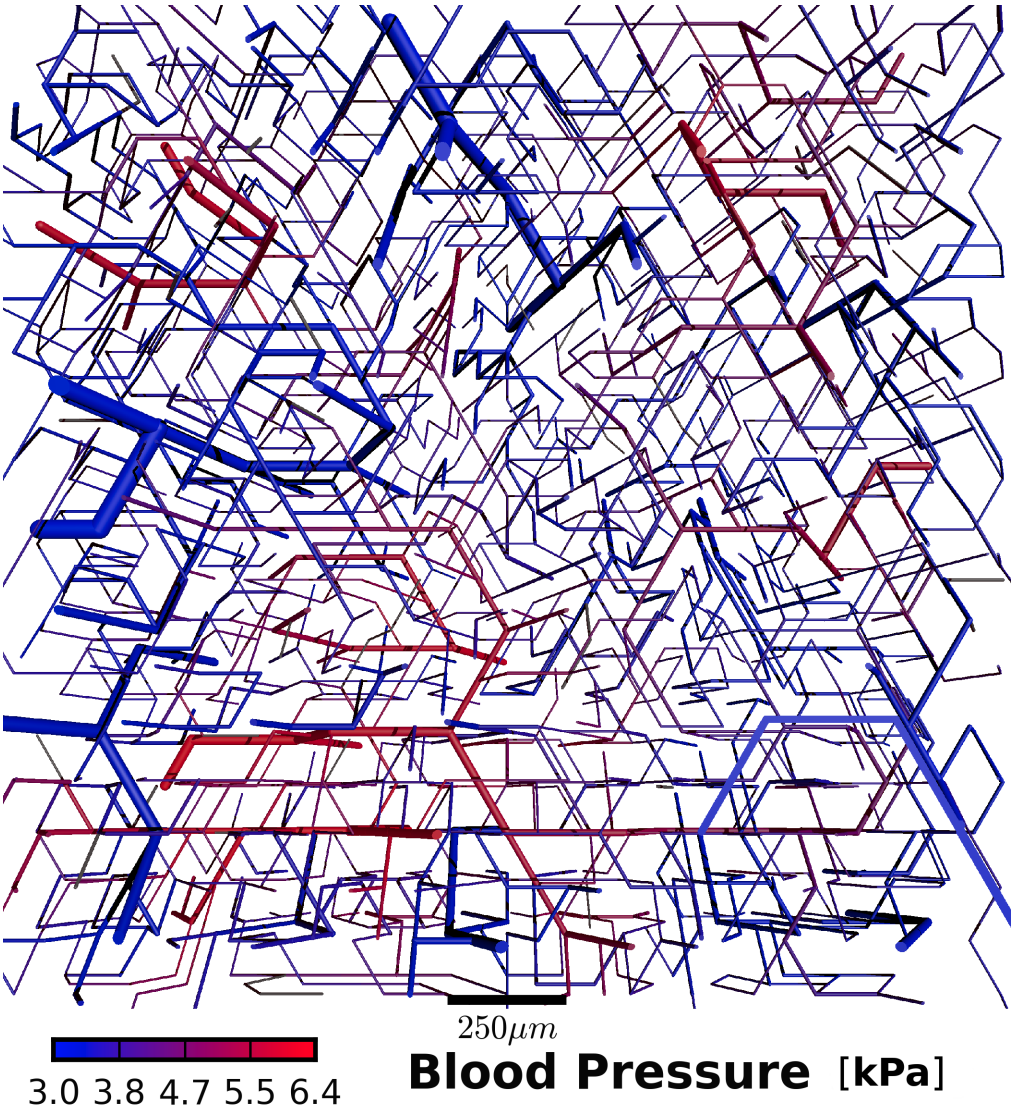


**Fig 2. Rendering of blood vessel network**
Here the rendering of the blood vessel network created in appendix B.1 is shown. Instructions on how to obtain such images from the software are explained in B.5. Blood pressure is color-coded in $kPa$ and a scale bar of 250 $\mu m$ is present. A slice throughout the whole 3D cube is displayed in the figure.

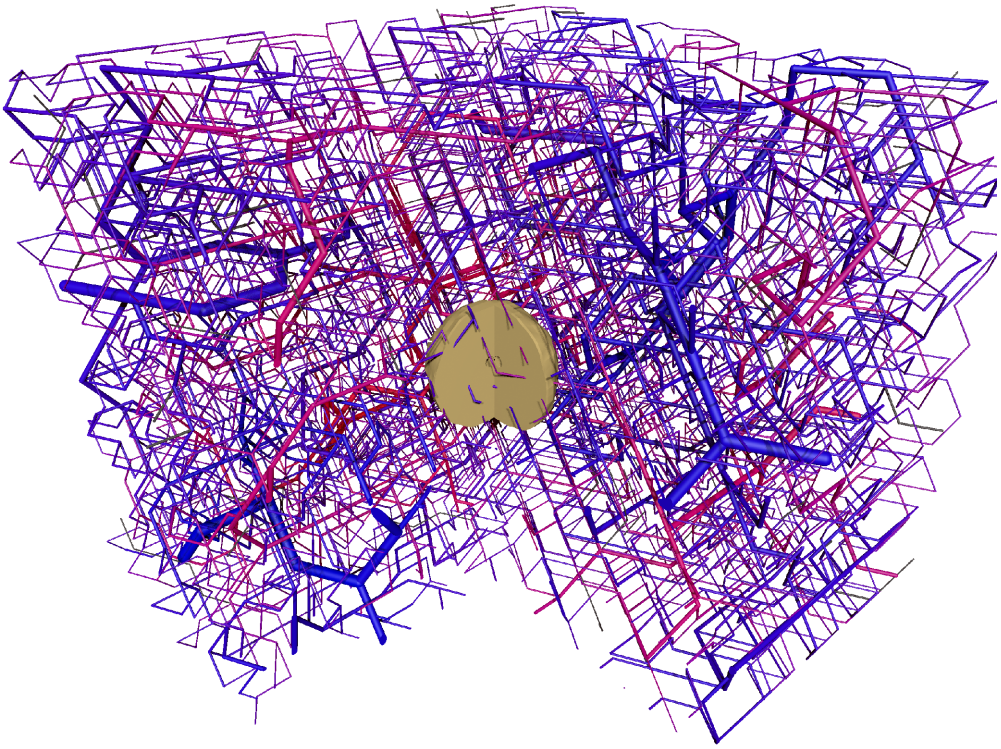

**Fig 3. Rendering of a bulk tissue tumor simulation**
Appendix B.2 explains how to obtain this image. The camera position 'pie' is chosen in order to see the tumor inside the cube. The vessel network is the same as shown in Fig 2

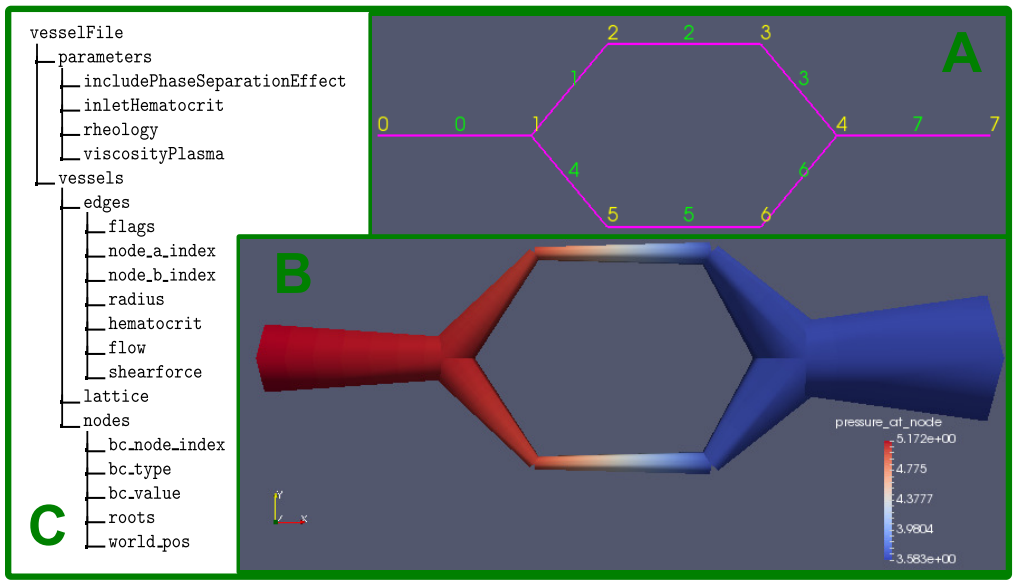


**Fig 4. Illustration of vessel data structure** This minimal example contains 3 arteries, 3 veins and 2 capillaries. Inset A: the topological structure given by labels of the node points in yellow and the label of the edges in green. Inset B: morphology calculated by tumorcode, note the size radii of the vessels and the colorcode due to the pressure given in $kPa$. Inset C: structure of the *'hdf5'* file containing the vessel structure.

be present. Using this file as argument of *'submitIff'* allows you to calculate distributions of the interstitial fluid flow and corresponding advection maps of drugs. Obviously, the dominant parameter in this simulation is the molecular weight of the drug (see IFF and drug distribution and the mentioned references). Also the oxygen content in the vessels and the surrounding tissue can be identified executing the detailed oxygen simulation by *'submitDetailedO2'*.

## Analyzing and visualization run

Like the simulations, our analysis programs are hybrids between *C++* and *python*. Since the datasets can become quite large in filesize and take a long time to be processed, most evaluation programs are equipped with datacache i.e. the results of certain costly computations are saved to disk and loaded from there the next time the execution is called with the same arguments. This facilitates iterative development, at least to some degree, without tedious recomputation.

In order to use state of the art visualization techniques, we export datasets created with *tumorcode* in 2 different data formats as demonstrated in Visualization tools. The amenities of *ParaView* allow an elaborated inspection of the vessel network structure. It offers interactive viewing and basic statistics on medium size systems, depending on your machines RAM. To render even the largest systems, we follow the ray trace approach of *POV-Ray*. Similar to *ParaView*, we offer a wrapper which converts vessel, tumor, oxygen and drug datasets to *POV-Ray* scene files, and run *POV-Ray* automatically in succession. Images of these kind are depicted in our latest publications.

# Applications

As an illustration of the capabilities of *tumorcode*, we designed the examples in the appendixB.1 to B.5 within the realm of the two studies already published. However doing the same, we scaled the simulation in the examples to be smaller and quickly reproducible. On a standard desktop PC (Intel i7,2x8GB RAM) they are executed within minutes.

For details on the exact setup we refer to [15, 16].

## Blood vessels and tumor morphology

In silico modeling features the opportunity to measure structural and dynamical quantities at the same time. Simple averages of e.g. flow, radii and vesselvolume are implemented straightforward in our software. Additionally, we can average data over spherical surfaces to mimic various experimental measurements. A profound inspection of a bulk tissue tumor simulation on a large scale could be found in [16, Fig 3]. There the graphs display MVD, vessel radius, cell velocity, oxygen, wall thickness and wall shear stress where the time course of tumor growth is visualized by different colors. The level set method allows to localize the distance of the tumor- tissue interface which is used as ordinate for the cited plots. Within that framework it is easy to define "inside" and "outside" of a tumor and focus on the differences.

Distinct experimental techniques such as MRI, micro-CT or xenografts are available to access morphological data of tumor tissue as well. But comparable data obtained from animal models usually requires the immolation of an animal per time point. Therefore such studies are heavily biases by inter- subject variation. Recently, we used this framework to create networks matching the morphology of a swine animal model. In appendix B.1 we create a dataset of this kind and present the analysis.

## IFF and drug distribution

To measure IFF and the related interstitial fluid pressure (IFP) in vivo is an especially hard task even with modern measurement methods. Regarding the medical aspects, special interest is drawn to the vasculature as source for therapeutics and nutrients. A solid understanding of IFF and IFP is also crucial for fighting cancer since an elevated IFP is believed to pose a barrier to drug delivery and oxygenation. The resulting hypoxia leads to inefficacy of drugs and radiation. For these type of simulation we consider the IFP as driving force of the flow, i.e. via Darcy's law. Differences between IFP, blood pressure and the fluid pressure within lymphatics drive transvascular fluxes according to Starling's equation. We solve the stationary case leading to a Poisson equation for the pressure.

Ensuing the determination of IFF, solute transport in tissue can be simulated. If specified by parameters (see appendix B.3), the mass balance equation for solutes is integrated numerically based on the calculated flow, diffusion coefficients, rate constants, etc. For an exemplary overview of such quantities, see Fig 5. For results of larger systems contact [16, Fig 4,7] where also a time course of Doxorubicin administration is shown. These type of simulations lay the basis to further analyze drug concentration distributions. Notably, our results obtained by these simulations suggest that in spite of an IFP plateau within the tumor, IFF does not cease and still allows for substantial convective transport which was contradictory to the view at that time. Furthermore we varied numerous parameters and the administration protocol emulating clinical studies which would be extensive and lengthy in practice.

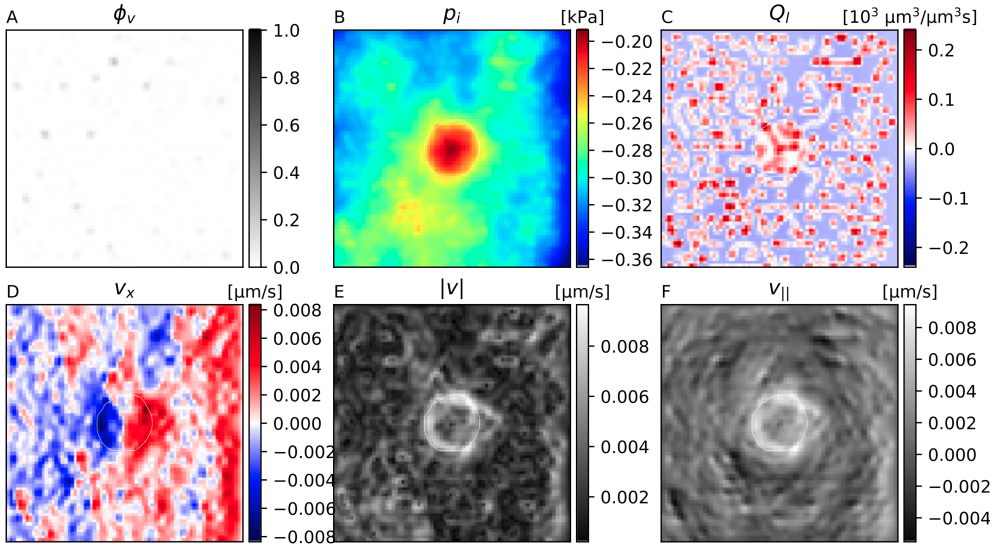


**Fig 5. Snapshots of interstitial fluid flow quantities** (A) Vessel volume fraction, (B) IFP, (C) Fluid source term, (D) x-component of the IF velocity, (E) Magnitude of the IF velocity, (F) Projection of IF velocity in outward direction. The plots were generated from 2d slices through the center of the simulation result obtained in appendix B.3. The contour line indicates the boundary of the viable tumor mass. The internal regions consist of necrotic tissue, while the outer area is normal host tissue.

## Detailed oxygen distribution

We augmented the model of oxygen transport in the simulation such that intravascular variation of the oxygen content along vessel center lines is taken into account. Moreover a varying hematocrit based on the (Red Blood Cell) phase separation effect (see [17] and section Hydro- and hemodynamics,Finite Elements Method ) has been implemented. Since most oxygen is transported through the body while bound to the RBCs, the hematocrit distribution is the physical key to the oxygen transport. Together with the unbound oxygen within the blood, we implemented a coarse-grained model of oxygen concentration along the direction of blood flow which causes tissue oxygenation. An example of these type of simulations is presented in appendix B.4. Here we show the results as illustration in Fig 6.

The focus of the conducted and published study was to simulate tissue total hemoglobin concentrations cHb and tissue blood oxygen saturation Y of tumors and host tissue and compare the results with clinical data on breast tumors and surrounding normal breast tissue obtained from a cohort of 87 breast cancer patients using optical mammograph. Following the workflow of *tumorcode* we began by constructing 90 distinct host vasculatures followed by growing fake tumors. Using the detailed oxygen simulation, we showed that the redirection of the hematocrit during the tumor growth processes accompanied by changes in flow rates is taking place and identified this as key element of the modulation in blood oxygen saturation. In contrast the fact that vascular dilatation leads to an elevated tumor oxygen saturation is not intuitive but confirmed by our simulations. Strikingly we found evidence that different types of initial vascular configurations are correlated to the variance of tissue hemoglobin concentration, tissue blood oxygen saturation and perfusion which raises biological insight.

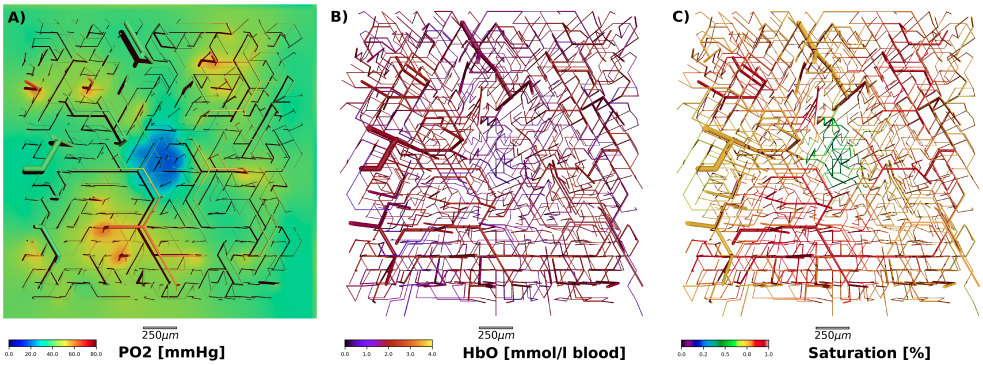


**Fig 6. Overview of detailed oxygen simulation.** *Inset A)* partial oxygen pressure within vessels and surrounding tissue. Note the hypoxic core in the center where the tumor grows. *Inset B)* amount of oxygenated hemoglobin within the vessels. For illustration purposes the tissue is not depicted. *Inset C)* blood oxygen saturation within the vessels. Note that all three figures show the same blood vessel network. See appendix B.4 for construction details.

# Availability and future directions

Good scientific practice demands to provide all relevant details in order to reproduce experiments. By sharing this code under open source license at `https://github.com/thierry3000/tumorcode` we allow everybody to reproduce our simulations. The installation procedure and the handling is elaborated in the corresponding wiki. Despite the code is platform independent, successful compilation and usage as described in the article is restricted to Unix-based systems at the moment. If required, support for different operating systems is feasible. Likewise, implementing a graphical user interface as extension of our module-based code is straight forward and could serve other researchers in life science with less background in computer science. Following the direction towards a common code base in the field, we hope this article and mainly the examples in the supplemental help to overcome technical struggles. In future, merging tumorcodewith other codes may be an option.

Currently the drug as well as the detailed oxygen distribution is calculated at time snapshots of the tumor growth process only. To engage pharmacodynamics and improve predictions, intertwining the time points could be considered in future. Additional information on the tumor could be added. For example the pH, ATP consumption or formation of metastasis is also relevant for clinics and could be included in our framework. The questions in the broad field of cancer research are immense. To overcome technical issues will spare time for scientific questions.

# A Digital resources

## A.1 Data created by examples see appendix B

**Data created in the examples** We add all data produced in Blood vessel network creation to Visualization tools.

## A.2 S1 Video.

**Blood vessel network creation process.** To illustrate the process of network

creation, we show an example in fast motion. The resulting dataset is quite big and therefore all intermediate stages are omitted. The complete source used to construct the data for the movie can be found in the file *povrayRenderVesselgeneratorDbg.py* for references.

## A.3 S2 Video.

**3D model of tumor- induced angiogenesis.** To illustrate the tumor model we show a video of the dynamical growth process.

# B Examples

## B.1 Blood vessel network creation

In this supplementary example we create an artificial blood vessel network using *submitVesselgeneration* and show some measurement tools. The algorithmically description is documented in [2].

# Usage

Entering

```
submitVesselgeneration -h
```

in your console should print possible options as seen in Listing 1.

**Listing 1.** Console options for *submitVesselgeneration*

```
usage: submitVesselgeneration [-h] [-t TYPE [TYPE ...]] [-p VESSELPARAMSET]
                              [-w WIDTH_CUBE] [--twoD] [-i ITER_H]
                              [-e ENSEMBLE_SIZE]

Compute an artificial blood vessel network.

optional arguments:
  -h, --help            show this help message and exit
  -t TYPE [TYPE ...], --type TYPE [TYPE ...]
                        Type of root node configurations, range from 0 to 8
  -p VESSELPARAMSET, --VesselParamSet VESSELPARAMSET
                         specify Parameterset for creation, possible configs
                         are found in /krebsjobs/parameters/vesselGenParams.py
  -w WIDTH_CUBE, --width_cube WIDTH_CUBE
                         Size of vesselcube you want to create
  --twoD                 Creates only 2D network, default is 3D
  -i ITER_H, --iter_h ITER_H
                         Number of hieracial iterations
  -e ENSEMBLE_SIZE, --ensemble_size ENSEMBLE_SIZE
                         Number of realizations
```

We will create systems of type 8 which means that the inlets (arterious roots) and the outlets (venous drains) are randomly distributed around a cube with length $2000\mu m$. Different types are shown here. One hierarchal iteration step is applied and in total 5 different realizations are created. The default parameters used in this example are illustrated in Fig 7. The command is:

```
submitVesselgeneration -t 8 -p default -w 2000 -i 1 -e 5
```

In about 10 minutes this will create the files *vessels-default-typeI-15x19L130-sample00.h5* to *vessels-default-typeI-15x19L130-sample04.h5* which we could analyze now.

# Analyzing

One of the main tools to extract information about a vessel network is

```
python2 /your/install/path/py/krebs/plotVessels.py -h
```

This call lists the following options

```
usage: plotVessels.py [-h] [-O] [-T] [-a] [-s]
                      [vesselFileNames [vesselFileNames ...]] grp_pattern

Plot/ Analyze infos about vessel network.

positional arguments:
  vesselFileNames       Vessel file to calculate
  grp_pattern           Where to find the vessel group in the file

optional arguments:
  -h, --help            show this help message and exit
  -O, --with-o2         look at detailed o2 data
  -T, --only_two_root   flag to change the considered types
  -a, --with_all_types  take all types
  -s, --singel_type
```

```
default (parameterset name)
├─ tip_radius_arterial = 2.5
├─ tip_radius_capi = 2.5
├─ tip_radius_vein = 3.8
├─ murray_alpha_vein = 3
├─ murray_alpha_artery = 3
├─ scale = 130
├─ max_sprout_radius_artery = 8
├─ max_sprout_radius_vein = 8
└─ calcflow
   ├─ viscosityPlasma = 1.2e-6
   ├─ rheology = ‘RheologySecomb2005‘
   ├─ inletHematocrit = 0.37
   └─ includePhaseSeparationEffect = True
```

**Fig 7.** Structure of a vessel creation parameters as stored in *vesselGenParams.py*.

By default the group where a vessel network is stored is labeled ‘vessels‘. In order to analyze the bunch of created vessel networks we hand the files together with group to the plot program.

```
python2 /your/install/path/py/krebs/plotVessels.py \
vessels-default-typeI-15x19L130-sample00.h5 vessels
```

This will create a file called *vessels-default-typeI-15x19L130-sample00.pdf* where the results are stored. The instructions to visualize such a data set are could be found in an other example.

## B.2 Tumor Growth

In this supplementary example we grow tumors on the previous created artificial blood vessel networks and show how to invoke some quantifications. In principal there are two programs starting tumor simulations, *submitBulkTissue* and *submitFakeTum* where the API works the same for both. The difference is that *submitBulkTissue* exploits our full tumor model also mimicking cells in contrast to *submitFakeTum* where the dynamic of the cells is just and expanding sphere with constant speed set by the parameters.

# Usage

Entering

```
submitBulkTissue -h
```

or

```
submitFakeTum -h
```

in your console should print possible options as seen in Listing 2.

**Listing 2.** Console options for *submitBulkTissue*

```
usage: submitBulkTissue [-h] [-n]
                        tumParamSet [vesselFileNames [vesselFileNames ...]]

Compute BulkTissue tumor. Either with or without vessels

positional arguments:
  tumParamSet       Valid configuration are found in
                    /py/krebsjobs/parameters/fakeTumorParams.py
  vesselFileNames   Vessel file to calculate

optional arguments:
  -h, --help        show this help message and exit
  -n, --no_vessel   compute the continuum model of tumor cells, no vessels
                    needed for that
```

We chose the parameter setting *defaultconfig_bulktissue* and provide a vessel network file created in previous examples by typing:

```
submitBulkTissue defaultconfig_bulktissue \
vessels-default-typeI-15x19L130-sample00.h5
```

It takes a desktop PC about 20 minutes to execute the simulation where the file *tumBulkdefaulttypeIsample00defaultconfig_bulktissue.h5* with the structure as depicted in 8 is created which we are going to analyze in the following.

# Analyzing

## Oxygen

To extract so informations about the oxygen content your can use

```
python2 \
/your/install/path/py/krebs/analyzeTissueOxygen.py -h
```

we run

```
python2 \
/your/install/path/py/krebs/analyzeTissueOxygen.py \
tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5
```

```
tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5
├── field_ld
├── parameters
├── out0000 ..................................timeframe of simulation
├── out0001
│   ├── fieldGf ......................growth factor conc. at timepoint
│   ├── fieldOxy ........................... oxygen conc. at timepoint
│   ├── vessel_volume_fraction
│   ├── tumor ..........................................tumor data
│   └── vessels ........................................vessel data
│       ├── edges
│       ├── lattice
│       └── nodes
├── out0002
├── out0003
├── out0004
└── out0005
```

**Fig 8.** Structure of bulk tissue tumor file as stored in *tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5*. Exemplary a single time frame is zoomed in.

## Fractal dimension

The fractal dimension are calculated by

```
python2 \
/your/install/path/py/krebs/fractaldim.py -h
```

This program comes with 2 modes. First, without any arguments we calculate the fractal dimensions by box counting. This could also be done on cluster computers. We run

```
python2 \
/your/install/path/py/krebs/fractaldim.py \
tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5 \
out0005
```

After calculating you created a file called *fractaldim_tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5*. Using the plot flag, you can now plot your data by

```
python2 \
/your/install/path/py/krebs/fractaldim.py --plot \
fractaldim_tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5 \
out0005
```

## Morphological data

Some morphological data is sampled by

```
python2 \
/your/install/path/py/krebs/analyzeVesselsBulkTumor.py -h
```

```
iff_default (parameterset name)
├── field_ld
├── iff ..........results of the IFF simulation
├── measurements
├── out0000
├── out0001 ......timeframe of drug-simulation
├── out0002
├── out0003
├── out0004
└── parameters
    ├── iff ......................related to IFF
    └── ift ......................related to drug
```

**Fig 9.** Structure of IFF simulation file as result of *submitIff*.

### B.3 Interstiatial fluid flow and drug transport

Providing a tumor simulation, we describe here how interstitial fluid pressure and drug distributions could be calculated and visualized. These simulation where used for the publication [16].

## Usage

Entering

```
submitiff -h
```

in your console should print possible options as seen in Listing 3.

**Listing 3.** Console options for *submitIff*

```
usage: submitIff [-h]
                 Iffparams [tumorFileNames [tumorFileNames ...]] grp_pattern

Compute IFF distributions and drugs analyze them.

positional arguments:
  Iffparams        choose the parameter for the simulation, possible configs
                   are found in /krebsjobs/parameters/parameterSetsIff.py
  tumorFileNames   tumor files to calculate
  grp_pattern      Where to find the tumor. Usually this is somthing with out*

optional arguments:
  -h, --help       show this help message and exit
```

We will simulate IFF for bulk tumors created in previous examples (*tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5*). From the parameters we choose the 'iff_default' where the evolution of the drugs is only calculated for 0.5 hours which is doable in about 2 minutes. See the parameter 'out_times' for this.

```
submitIff iff_default \
tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5 \
out0005
```

Upon a successful run the file *iff_default_ulk-default-typeI-sample00-defaultconfig_bulktissue.h5* with structure as depicted in Fig 9 will be created.

# Analyzing

## IFF

One of the main tools to extract information about an IFF simulation file is

```
python2 /your/install/path/py/krebs/plotIff.py -h
```

We run

```
python2 /your/install/path/py/krebs/plotIff.py \
iff_default_ulk-default-typeI-sample00-defaultconfig_bulktissue.h5
```

This gives a detailed analysis on the iterstitial fluid pressure and flow which was calculated. Integral plots as well as radial averages are calculated and shown in diagrams. Further a 2D histograms on the dataset is shown as image.

## Drug

Quantitative informations on the evolution of drugs within the blood and the interstial volume could be drawn with the help of

```
python2 /your/install/path/py/krebs/plotDrug.py -h
```

We run

```
python2 /your/install/path/py/krebs/plotDrug.py \
iff_default_ulk-default-typeI-sample00-defaultconfig_bulktissue.h5
```

```
o2_tumBulk-default (filename)
├── po2
│   ├── out0005
│   ├── field_ld
│   ├── iterations
│   ├── parameters
│   ├── samples_and_fluxes
│   ├── p2field ............................ O2 tissue field
│   └── po2vessels ......................... O2 in vessels
└── recomputed_flow ............ flow computed before O2
```

**Fig 10.** Structure of DetailedO2 simulation file as result of *submitDetailedO2*.

### B.4 Detailed oxygen distribution

Providing a tumor simulation, we calculate the detailed oxygen distribution as described in [15] and run the scripts in order to analyze those oxygen distribution.

## Usage

Entering

```
submitDetailedO2 -h
```

in your console should print possible options as seen in Listing 4.

**Listing 4.** Console options for *submitDetailed*

```
usage: submitDetailedO2 [-h] [-a] [-r] [--systemsize SYSTEMSIZE]
                        o2params [vesselFileNames [vesselFileNames ...]]
                        grp_pattern

Compute po2 distributions and analyze them. In normal mode it takes arguments:
parameter set name, filenames, h5 group pattern. In analysis mode -a, it takes
arguments: filenames, h5 group pattern.

positional arguments:
  o2params              choose the parameter for the simulation. found at
                        /py/krebsjobs/parameters/parameterSetsO2.py
  vesselFileNames       Vessel file to calculate
  grp_pattern           Where to find the vessel group in the file

optional arguments:
  -h, --help            show this help message and exit
  -a, --analyze         loop through all files analyze data and make plot
  -r, --render          launch rendering of overview images
  --systemsize SYSTEMSIZE
                        num threads = 1 * systemsize, memory = 2 Gb *
                        systemsize (for o2 computation; analysis uses
                        different scaling (see source)
```

We will simulate the detailed oxygen distribution for bulk tumors created in previous examples (here *tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5*). From the parameters we choose the 'default_o2'.

```
submitDetailedO2 default_o2 \
tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5 \
out0005
```

Upon a successful run the file *o2_tumBulk-default-typeI-sample00-defaultconfig_bulktissue_default_o2.h5* with structure as depicted in Fig 10 will be created.

## Analyzing

In principal most of our evaluation scripts focus on averaging multiple runs of a single parameter set to get some statistics. However, in the scope of this example we restrict the application to a single file only due to the computational efforts.

The script which draws conclusions about perufsion, tissue oxygen saturation etc. is *plotVessels.py* together with the flag *-O* indication the calculation of the detailed oxygen quantities.

```
python2 /your/install/path/py/krebs/plotVessels.py -h
```

Running

```
python2 /your/install/path/py/krebs/plotVessels.py -O \
o2_tumBulk-default-typeI-sample00-defaultconfig_bulktissue_default_o2.h5 \
po2/out0005
```

This creates a pdf containing necessary informations.

## B.5 Visualization tools

Using the data created in previous examples, we show how the simulation results can be visualized using your tools facilitated by Povray and ParaView.

# Povray

Entering

```
submitPovrayRender -h
```

in your console should print possible options as seen in Listing 5. There are quite some options here for adjusting the graphical representation. Note that you can also store and load these settings from the file *povrayRenderSettings.py* as it is done in your other scripts. In order to not track every graphic setting ever made, we decided not to include the file *povrayRenderSettings.py* in the repository. However we provide a template called *povrayRenderSettingsTemplate.py*.

**Listing 5.** Console options for *submitPovrayRender*

```
usage: submitPovrayRender [-h] [-p POVPARAMSSET] [-d DATALIST] [-f]
                          [--filterradiushighpass FILTERRADIUSHIGHPASS]
                          [--filterradiuslowpass FILTERRADIUSLOWPASS]
                          [--overlay] [--dpi DPI] [--format FORMAT] [-c CAM]
                          [-u] [-a] [--fontcolor FONTCOLOR]
                          [--temp_file_dir TEMP_FILE_DIR] [--keep_files]
                          [--assumed_gamma ASSUMED_GAMMA]
                          [--background BACKGROUND]
                          [--ambient_color AMBIENT_COLOR] [--res RES]
                          [--num_threads NUM_THREADS] [--out_alpha]
                          [--cam_distance_multiplier CAM_DISTANCE_MULTIPLIER]
                          [--colored_slice COLORED_SLICE]
                          [--not_render_volume] [--not_render_vessels]
                          vesselFileNames [vesselFileNames ...] grp_pattern

Povray wrapper

positional arguments:
  vesselFileNames
  grp_pattern

optional arguments:
  -h, --help            show this help message and exit
  -p POVPARAMSSET, --povParamsSet POVPARAMSSET
                        use parameters found in povrayRenderSettings.py
                        (default: None)
  -d DATALIST, --data DATALIST
                        which data (pressure, flow, shearforce, hematocrit) as
                        comma separated list (default: ['pressure'])
  -f, --filter-uncirculated
                        filter uncirculated vessels (default: False)
  --filterradiushighpass FILTERRADIUSHIGHPASS
                        filter vessels from tree above this value (default:
                        -1.0)
  --filterradiuslowpass FILTERRADIUSLOWPASS
                        filter vessels from tree below this value (default:
                        -1.0)
  --overlay             decide if mpl overlay is created (default: True)
  --dpi DPI             dpi for the rendering (default: 300.0)
  --format FORMAT       output format of image (default: png)
  -c CAM, --cam CAM     camera mode: topdown, pie, topdown_slice (default:
                        topdown_slice)
  -u, --plot_auc        for area under curve, we have only a single timepoint
                        (default: False)
  -a, --auto_colorscale
                        (default: False)
  --fontcolor FONTCOLOR
                        fontcolor in overlay, use mpl style colors (default:
                        black)
  --temp_file_dir TEMP_FILE_DIR
                        dir for temp povray scene (default: None)
  --keep_files          keep tmp file? (default: False)
  --assumed_gamma ASSUMED_GAMMA
                        (default: 1.0)
  --background BACKGROUND
                        (default: 1.0)
  --ambient_color AMBIENT_COLOR
                        (default: (0.1, 0, 0))
  --res RES             (default: (1024, 1024))
  --num_threads NUM_THREADS
                        (default: 7)
  --out_alpha           (default: False)
  --cam_distance_multiplier CAM_DISTANCE_MULTIPLIER
                        (default: 1.0)
  --colored_slice COLORED_SLICE
                        (default: True)
  --not_render_volume   For combined images (default: False)
  --not_render_vessels  For combined images (default: False)
```

This program automatically recognizes the type of file your entered. Depending your input, different subroutines to render vessels, tumors, IFF or detailed oxygen datasets are called. We provide examples for all.

## Vessels

We begin by visualizing a vessel network.

```
submitPovrayRender \
vessels-default-typeI-15x19L130-sample00.h5 vessels
```

creates the file *vessels-default-typeI-15x19L130-sample00_vessels_pressure.png* which is also show in main document. The *-d*, *–data* flag allows you to switch the data which is colorcoded on the vessels e.g.

```
submitPovrayRender \
vessels-default-typeI-15x19L130-sample00.h5 vessels \
-d flow,shearforce
```

will create renderings of the flow or the shearforce. The figure shown in the main document was created by

```
submitPovrayRender \
vessels-default-typeI-15x19L130-sample00.h5 vessels \
-p 'figure_plos_vessels'
```

## Tumors

We create a rendering of the tumor simulation at time point *out0005* from the file *tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5*.

```
submitPovrayRender \
tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5 \
out0005
```

Upon a successful run the image *tumBulk-default-typeI-sample00-defaultconfig_bulktissue-out0005.png_slice.png* should be present. To illustrate the camera option (*-c*, *–cam*), we run an other example with

```
submitPovrayRender \
tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5 \
out0005 \
-c pie
```

The figure shown in the main document was created by

```
submitPovrayRender \
tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5 \
out0005 \
-p 'figure_plos_tumor'
```

## Detailed Oxygen

Of course these datasets can also be visualized. We run

```
submitPovrayRender \
o2_tumBulk-default-typeI-sample00-defaultconfig_bulktissue_default_o2.h5 \
po2/out0005
```

Note here the *–render_volume* and *–render_vessels* option which allows you to visualize datasets with and without the vessels in the same plane.

```
submitPovrayRender \
o2_tumBulk-default-typeI-sample00-defaultconfig_bulktissue_default_o2.h5 \
po2/out0005 \
--not_render_vessels
```

### IFF

Analogously we run here

```
submitPovrayRender \
iff_default_ulk-default-typeI-sample00-defaultconfig_bulktissue.h5 \
out0004
```

## ParaView

To export our file format to paraView, we exploit the script

```
python2 /your/install/path/py/krebs/hdf2vtk.py -h
```

### Vessels

For example, here with the lately produced vessels

```
python2 /your/install/path/py/krebs/hdf2vtk.py \
vessels-default-typeI-15x19L130-sample00.h5 vessels
```

creating the file
*vessels-default-typeI-15x19L130-sample00-vessels.vtk* which is readable by paraView.

### Tumor

Here, with tumor file.

```
python2 /your/install/path/py/krebs/hdf2vtk.py \
tumBulk-default-typeI-sample00-defaultconfig_bulktissue.h5 \
out0005 \
```

This creates input files for paraView. One for the vessels and one for the field. When you load these sets into your paraView, make sure to change representation style to something meaningful e.g. “Points”.

## References


1. Carmeliet P, Jain RK. Angiogenesis in cancer and other diseases. Nature. 2000;407(6801):249–257. doi:10.1038/35025220.

2. Rieger H, Fredrich T, Welter M. Physics of the tumor vasculature: Theory and experiment. Eur Phys J Plus. 2016;131(2):31. doi:10.1140/epjp/i2016-16031-9.

3. Perfahl H, Hughes BD, Alarcón T, Maini PK, Lloyd MC, Reuss M, et al. 3D hybrid modelling of vascular network formation. Journal of Theoretical Biology. 2017;414:254–268. doi:10.1016/j.jtbi.2016.11.013.

4. Merks RMH, Perryn ED, Shirinifard A, Glazier JA. Contact-Inhibited Chemotaxis in De Novo and Sprouting Blood-Vessel Growth. PLOS Computational Biology. 2008;4(9):e1000163. doi:10.1371/journal.pcbi.1000163.

5. Anderson ARA, Chaplain MaJ. Continuous and discrete mathematical models of tumor-induced angiogenesis. Bull Math Biol. 1998;60(5):857–899. doi:10.1006/bulm.1998.0042.

6. Macklin P, McDougall S, Anderson ARA, Chaplain MAJ, Cristini V, Lowengrub J. Multiscale modelling and nonlinear simulation of vascular tumour growth. J Math Biol. 2009;58(4-5):765–798. doi:10.1007/s00285-008-0216-9.

7. Milotti E, Chignola R. Emergent Properties of Tumor Microenvironment in a Real-Life Model of Multicell Tumor Spheroids. PLOS ONE. 2010;5(11):e13942. doi:10.1371/journal.pone.0013942.

8. Preziosi L, Tosin A. Multiphase modelling of tumour growth and extracellular matrix interaction: mathematical tools and applications. J Math Biol. 2008;58(4-5):625–656. doi:10.1007/s00285-008-0218-7.

9. Grogan J, Connor AJ, Markelc B, Muschel RJ, Maini PK, Byrne HM, et al. Microvessel Chaste: An Open Library for Spatial Modelling of Vascularized Tissues. bioRxiv. 2017;doi:10.1101/105692.

10. Shirinifard A, Gens JS, Zaitlen BL, Popławski NJ, Swat M, Glazier JA. 3D Multi-Cell Simulation of Tumor Growth and Angiogenesis. PLOS ONE. 2009;4(10):e7190. doi:10.1371/journal.pone.0007190.

11. The HDF Group. Hierarchical Data Format, version 5; 1997.

12. Heroux M, Bartlett R, Hoekstra VHR, Hu J, Kolda T, Lehoucq R, et al. An Overview of Trilinos. Sandia National Laboratories; 2003. SAND2003-2927.

13. Gödde R, Kurz H. Structural and biophysical simulation of angiogenesis and vascular remodeling. Dev Dyn. 2001;220(4):387–401. doi:10.1002/dvdy.1118.

14. Hogea CS, Murray BT, Sethian JA. Simulating complex tumor dynamics from avascular to vascular growth using a general level-set method. J Math Biol. 2006;53(1):86–134. doi:10.1007/s00285-006-0378-2.

15. Welter M, Fredrich T, Rinneberg H, Rieger H. Computational Model for Tumor Oxygenation Applied to Clinical Data on Breast Tumor Hemoglobin Concentrations Suggests Vascular Dilatation and Compression. PLOS ONE. 2016;11(8):e0161267. doi:10.1371/journal.pone.0161267.

16. Welter M, Rieger H. Interstitial Fluid Flow and Drug Delivery in Vascularized Tumors: A Computational Model. PLoS ONE. 2013;8(8):e70395. doi:10.1371/journal.pone.0070395.

17. Pries AR, Secomb TW, Gaehtgens P, Gross JF. Blood flow in microvascular networks. Experiments and simulation. Circulation Research. 1990;67(4):826–834. doi:10.1161/01.RES.67.4.826.