1 Introduction

The Vascular Porous (VaPor) model is an open source software designed to simulate biological mass flows and heat transfer. Blood vessels are modelled as 1D line segments embedded within a 3D domain representing tissue. All mass and energy balances between these domains are conserved. The current version is set up to focus on cerebral temperatures. It includes domain reading, vessel generation, as well as flow and temperature solvers. Additionally, some display features are included to help read the results.

The purpose of this document is to detail the various functions within the code and how to use them.

2 Setting-Up

This section runs through the various options for setting up a simulation with the VaPor model. A full hierarchy of the various functions within the VaPor model are listed in Fig. 1.

To run a simulation, execute the script: `mainScript`. This script is split into two main parts, the initialisation (where all options and physical values are set) and the execution. All user input should be located within the initialisation scripts (located in the Initialisation folder) and alterations to the execution scripts should only be performed with understanding of the effects. The initialisation scripts contain all the options used in generating the data used in the the publication. A brief summary of the initialisation scripts and the options within is found below:

- `physicalParameterInitialisation`: This script establishes information on the physical properties for various aspects used within the model.
    - `Option_ConvertPerfusion[true false]`: This option allows the input of perfusion in either units of $ml/100g/min$ (`[true]`) or $kg/m^3/s$ (`[false]`).
    - `Option_TargetPerfusion[true false]`: This option allows the scaling of perfusion within the brain domain to a target value (e.g. if the average perfusion of the brain was desired to be exactly $50ml/100g/min$).
    - `Option_TargetPerfusionConvert[true false]`: This option allows the input of the target perfusion used in `Option_TargetPerfusion` either units of $ml/100g/min$ (true) or $kg/m^3/s$ (false).
- `domainReadingInitialisation`: This script establishes the file locations and sizes of the image sequences used in creating the 3D domains. It is currently set up to use the provided image sequences and tissue types. These tissue types correspond with the physical parameters specified in `physicalParameterInitialisation`.
    - `Option_DomainAdjustment[default]`: This sets the options for adjusting the data from the image sequences provided. Currently, no other option than `[default]` is defined.
    - `Option_VoxelSizeAdjustment[true false]`: This option allows the adjustment of the voxel size in the model. This does not affect the number of voxels but instead scales the model. The default value is $1.5mm$.
    - `Option_DomainCoarsening[none two three six]`: This option adjusts the number of voxels in the domain. The options shown designate the factor by which the domain is coarsened. It is highly recommended that a least a factor of two is selected otherwise the linear solvers require considerable amounts of memory.

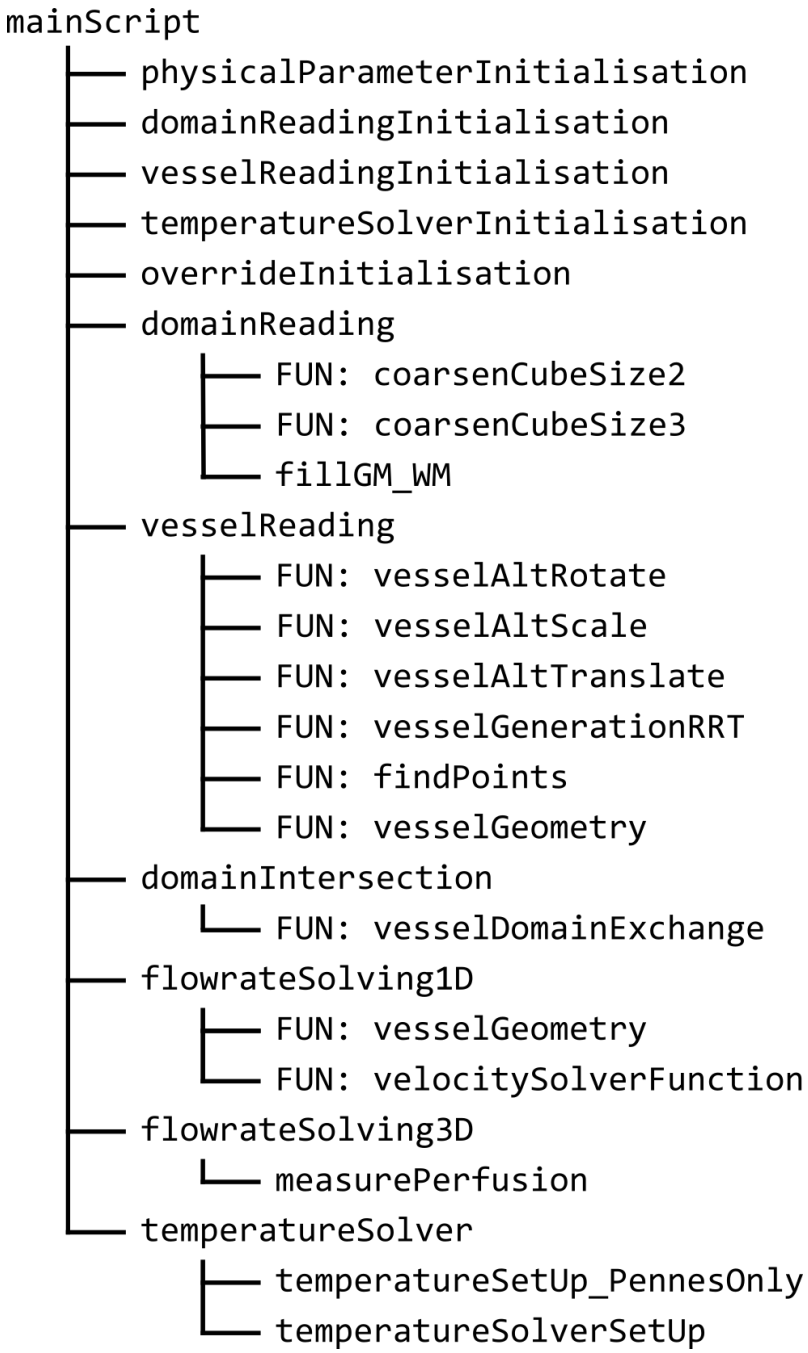

Figure 1: Hierarchy of script files and function files used within the VaPor software. The filenames pre-empted by `FUN` denote functions to which only certain variables are passed. All other files are script files.

- `vesselReadingInitialisation`: This script establishes the file locations for extracting the 1D vessel domains and defining inlet and outlet locations and fractions of overall flowrate. It is currently set up to either load the base arterial or venous trees or the possibility to load vessels from a previous trial. The parameters for vessel generation are also included in here.

    - `Option_VesselAdjustment[default]`: This sets the options for adjusting the data from the arterial and venous trees provided. Currently, no other option than `[default]` is defined.

    - `Option_LoadPrevious[true false]`: This option loads the vessel trees from previous results to be used within the model. It overrides any of the default vessel information used but can still be expanded upon using `Option_GenerateVessels`.

    - `Option_LoadPreviousCoarsening[none two three six]`: If the previous trial was performed using a finer domain than the current trial, then the vessels can be coarsened to fit.

    - `Option_LoadPreviousRefining[none two three six]`: If the previous trial was performed using a coarser domain than the current trial, then the vessels can be refined to fit.

    - `Option_GenerateVessels[true false]`: This option allows for the expansion of the established vessel tree. Currently a space filling algorithm is used to ensure the distribution of blood but it is expected a more physiological version will be added in future.

- `temperatureSolverInitialisation`: This script establishes the boundary conditions used in the temperature solver.

    - `Option_PennesOnly[true false]`: This options solves the system using Pennes Bioheat Equation only. By selecting this option, many of the scripts are bypassed as all flowrates and vessel data become unnecessary.

    - `Option_TemperatureDifference[true false]`: This option sets up a second temperature solve with a different boundary temperature and/or heat transfer coefficient. This allows for quick comparison under different conditions.

    - `Option_LimitedCooling[true false]`: This options specifies a height limit in the domain where cooling occurs. Below the set limit, no heat transfer to the boundary occurs.

- `overrideInitialisation`: This is an optional initialisation script that overrides any previously established parameter or option. This is useful if multiple trials would be run in a loop and the options to be varied can be specified here.

3 Display

Three display functions are included to facilitate the post-processing of data:

- `FUN: planecut(Dir,Val,Data)`: this displays a cross section of any 3D data set generated by the program using the `slice` function in Matlab. It requires three inputs. `Dir` is the plane of the slice which is a string input `['x' 'y' 'z']`. `Val` is the location of the slice given with respect to voxel location. If a specific distance (e.g. in m) is required, the dividing the distance by `VoxelSize` will give the voxel location. `Data` is the 3D data set to be sliced.

- FUN: `drawVessels(Vessel)`: this displays the vessel geometry as line segments. Red lines indicate generated segments or segments altered by the generation procedure. This function is not recommended for vessel trees larger than 20,000 nodes as it can take a substantial amount of time to visualise.

- FUN: `draw3DVessels(Vessel,Data,VoxelSize,Limit)`: this displays the vessel geometry as 3D cylinders, coloured by input data (e.g. temperature or flowrate). There is a limit imposed on the minimum diameter that is shown (default 20% of `VoxelSize`) beyond which the vessel segment will not be shown.

Multiple slices and vessel trees can be displayed with vessel trees by employing the `hold on` option for the figure within Matlab.

4 Outputs

This section describes the main outputs of the program that should be used for data processing.

- Domains: These are identifications for voxels present in different domains.
  - `DomTot`: A 3D matrix containing identification of voxels present in the full domain (brain and surrounding tissue).
  - `GM_WM`: A 3D matrix containing identification of voxels present in the brain domain only (grey matter and white matter).
  - `GM`: A 3D matrix containing identification of voxels present in the grey matter (grey matter $\geq 50\%$).
  - `WM`: A 3D matrix containing identification of voxels present in the white matter (grey matter $< 50\%$).
  - `Borders`: A 3D matrix containing identification of voxels that are located on the boundary.
  - `BordersScalp`: A 3D matrix containing identification of voxels that are located on the boundary, ignoring those at with a boundary only at $K = 1$ as base of the model is assumed to be adiabatic.

- Vessels: these contain the vessel information. The contents of the vessel trees is explained further within the theory section.
  - `Vessel1`: A $N \times 7$ matrix containing the information for arteries (where N is the number of nodes in the arterial vessel tree).
  - `Vessel2`: A $N \times 7$ matrix containing the information for veins (where N is the number of nodes in the venous vessel tree).
  - `L1`: A 1D vector containing the lengths (in m) of each segment in the arterial vessel tree.
  - `L2`: A 1D vector containing the lengths (in m) of each segment in the venous vessel tree.
  - `Davg1`: A 1D vector containing the average diameter (in m) of each segment in the arterial vessel tree.
  - `Davg2`: A 1D vector containing the average diameter (in m) of each segment in the venous vessel tree.

- `AL1`: A 1D vector containing the surface area (in m^2) of each segment in the arterial vessel tree.
- `AL2`: A 1D vector containing the surface area (in m^2) of each segment in the venous vessel tree.
- `Aavg1`: A 1D vector containing the average cross-sectional area (in m^2) of each segment in the arterial vessel tree.
- `Aavg2`: A 1D vector containing the average cross-sectional area (in m^2) of each segment in the venous vessel tree.
- `Vol1`: A 1D vector containing the volume (in m^3) of each segment in the arterial vessel tree.
- `Vol2`: A 1D vector containing the volume (in m^2) of each segment in the venous vessel tree.

- Flowrates: these contain the flowrate and velocity results from the two flowrate solvers, `flowrateSolving1D` and `flowrateSolving3D`.
  - `FdotArt`: A $N \times 2$ matrix containing the flowrate information (in kg/s) for arteries (where N is the number of nodes in the arterial vessel tree).
  - `FdotVein`: A $N \times 2$ matrix containing the flowrate information (in kg/s) for veins (where N is the number of nodes in the venous vessel tree).
  - `Perfusion`: A 3D matrix containing the calculated predicted values for perfusion (in $kg/m^3/s$) based on input tissue values for each voxel in the domain.
  - `PerfusionAlt`: A 3D matrix containing the calculated predicted values for perfusion (in $ml/100g/min$) based on input tissue values for each voxel in the domain.
  - `MassFlow`: A 3D matrix containing the calculated mass flowrates (in kg/s) for each voxel in the brain domain.
  - `MeasuredPerfusion`: A 3D matrix containing the calculated perfusion values (in $ml/100g/min$) for each voxel in the brain domain.
  - `U`: A 3D matrix containing the superficial velocities (in m/s) of blood at the interface between every voxel in the x direction.
  - `V`: A 3D matrix containing the superficial velocities (in m/s) of blood at the interface between every voxel in the y direction.
  - `W`: A 3D matrix containing the superficial velocities (in m/s) of blood at the interface between every voxel in the z direction.

- Temperatures: these contain the temperature results from `temperatureSolver`.
  - `Tt`: A 3D matrix containing the tissue temperatures of the system.
  - `Tb`: A 3D matrix containing the blood temperatures of the system.
  - `T_Art`: A 1D vector containing the temperatures of the arteries at each corresponding node.
  - `T_Vein`: A 1D vector containing the temperatures of the arteries at each corresponding node.
  - `Tt_Save`: Same as `Tt`. If `Option_TemperatureDifference` is enabled this stores the results from the first iteration of `temperatureSolver`.

- `Tb_Save`: Same as `Tb`. If `Option_TemperatureDifference` is enabled this stores the results from the first iteration of `temperatureSolver`.
- `T_Art_Save`: Same as `T_Art`. If `Option_TemperatureDifference` is enabled this stores the results from the first iteration of `temperatureSolver`.
- `T_Vein_Save`: Same as `T_Vein`. If `Option_TemperatureDifference` is enabled this stores the results from the first iteration of `temperatureSolver`.
- `Tt_Diff`: A 3D matrix containing the difference of tissue temperatures between the two iterations of `temperatureSolver` if `Option_TemperatureDifference` is enabled (the subtraction of `Tt_Save` from `Tt` after both iterations are complete).

The Domain outputs are useful in processing data. For example, to find the average tissue temperature within the brain only, the command `mean(Tt(GM_WM))` can be used. To find the average tissue temperature in the whole domain, the command `mean(Tt(DomTot))` can be used. Values outside of domains are generally designated `NaN` (not a number).