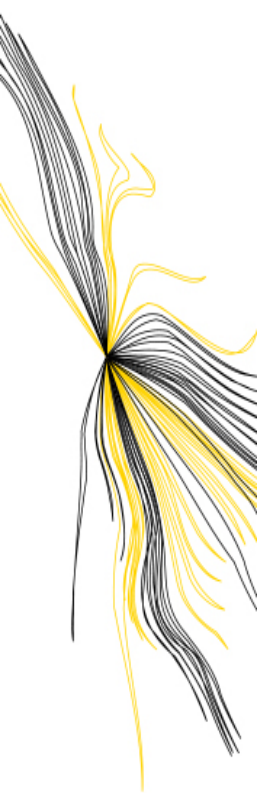
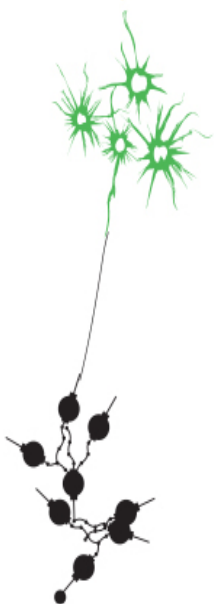


3D ANISOTROPIC CONDUCTION MODEL

MANUAL BY DANIEL WILMES

- This manual is intended to explain the most important parts of the anisotropic Matlab 3D conduction model and how to use it
- It explains the main usage pipeline and which parameters are intended to be changed by the user
- For more detailed information into the calculation process the user is advised to look into the code as the comments in there are quite elaborate and give more information than this manual can



MAIN USAGE

- Running the model with current parameters only requires executing the first block of code that contains RunModel with an additional option to start a Matlab parallel pool.

```
25
26      %whether to parallelize computation or not
27 -    use_parallel = false;
28
29 -    [U,I,P,Z] = RunModel(use_parallel);
30
```

- All user-controlled parameters are defined in InitParameters and should be adjusted inside this function

```
32
33      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
34      %Initialize calculations to represent Transducers Paper structure + 3D
35      %additions
36      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37    + function [Ntraxels,Width,Height,Length,Rho,Sigma_y,Sigma_z,epsilon,Ctrans_y,Ctrans_z,Uin,Uou,Fmin,Fmax,Fnum]=InitParameters...
110
```

CHANGING PARAMETERS

- In addition to the more obvious constants regarding the geometry and material properties the user can also control the meandering and terminal-ground-locations inside InitParameters
- There are two helper-functions to establish the meanders.

1. AllOpenConnections which is necessary to first create the cuboid in terms of the boundary conditions
2. AllMeanderInput to connect traxels through meanders according to standard printing behaviour

```
103
104 -      [Uin,Uou]=AllOpenConnections(Ntraxels);
105 -      [Uin,Uou]=AllMeanderInput(Ntraxels,Uin,Uou);
106 -      Uin( 1: 1,1:2)=[1 Uhig];
107        %   Uin( end:end,1:2)=[1 Ulow];
108 -      Uou( end:end,1:2)=[1 Ulow];
109 -  end
```

- More exotic connections are also possible but need to be hardcoded

2D VISUALIZATION

- The 2D-layer-wise visualization has 2 boolean arguments that change the scaling of values across the plot
- The function is called inside RunModel (ctrl+f → „slice_3d_plot“)
- The second last argument updates the axes of the graph to only adapt to the current layer‘s range of values rather than scaling it to the min and max of the whole cuboid. This allows better insight per layer.
- The last argument scales the colormap in the same way

```
126     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
127   function slice_3D_plot(Length,Width,N,U,I,P,Pdens,Freq,AdjustAxes,AdjustColor)...
270
```

3D VISUALIZATION

```
function f = Visualize_3D(U,I,P,N, freq, U_alpha_map, I_alpha_map, P_alpha_map, dims, scale2cube)
```

- „Visualize_3D“ is the main function for the 3D plot and contains all changeable parameters as arguments
- The alphamaps are the only ones intended to be changed by a user

- There is a helper function called „single_plot“ which does most of the more complicated visuals behind „Visualize_3D“, so if you intend to modify or use the code outside of this project this is most likely your point of interest

```
function single_plot(data, N, ax, alphatype, dims, scale2cube)
```

COMMENTS

- All functions have comments above them to explain the arguments and what they do. Example for „Visualize_3D“ bottom right.

- Additionally, there are comments inside each function to explain the process in which they perform actions. This should make it quite easy to find certain points of interest in case the user wants to change something. Example inside „single_plot“ bottom left.

```
%make meshgrids for slices at bottom and end of x- and y-axes
[xz,yz] = meshgrid(0:d(2), 0:d(1));
zz = xz .* 0;

[xy,zy] = meshgrid(0:d(2), 0:d(3));
yy = xy .* 0 + d(1) - 1;

[yx,zx] = meshgrid(0:d(1), 0:d(3));
xx = yx .* 0 + d(2) - 1;
```

```
%Function to create a 3D-model of voltage, current and power in one figure.
%The opacity of each part is determined by an alphamap.
%Only plots the real part of complex data
% U : (complex) double(N(1)*N(2), inf)
% I : (complex) double(N(1)*N(2), inf)
% P : (complex) double(N(1)*N(2), inf)
% N : int Number of traxels in y- and z-direction [y,z]
% U_alpha_map : Alphamap for voltage. Can be a string to use a
% matlab-native map or a custom array. A good one here is 'vdown' which is
% (counterintuitively) a 'V'-shape making very large and small values
% visible.
% I_alpha_map : Alphamap for current. Can be a string to use a
% matlab-native map or a custom array. A good one here is 'rampup' which is
% just a linearly increasing map.
% P_alpha_map : Alphamap for power. Can be a string to use a
% matlab-native map or a custom array. A good one here is 'rampup'
% dims : double(3) = [Length, Width, Height] of individual traxels
% scale2cube : bool, causes scaling of visualization to a cube rather
% than displaying it realistically using the dims-argument
function f = Visualize_3D(U,I,P,N, freq, U_alpha_map, I_alpha_map, P_alpha_map, dims, scale2cube)
```