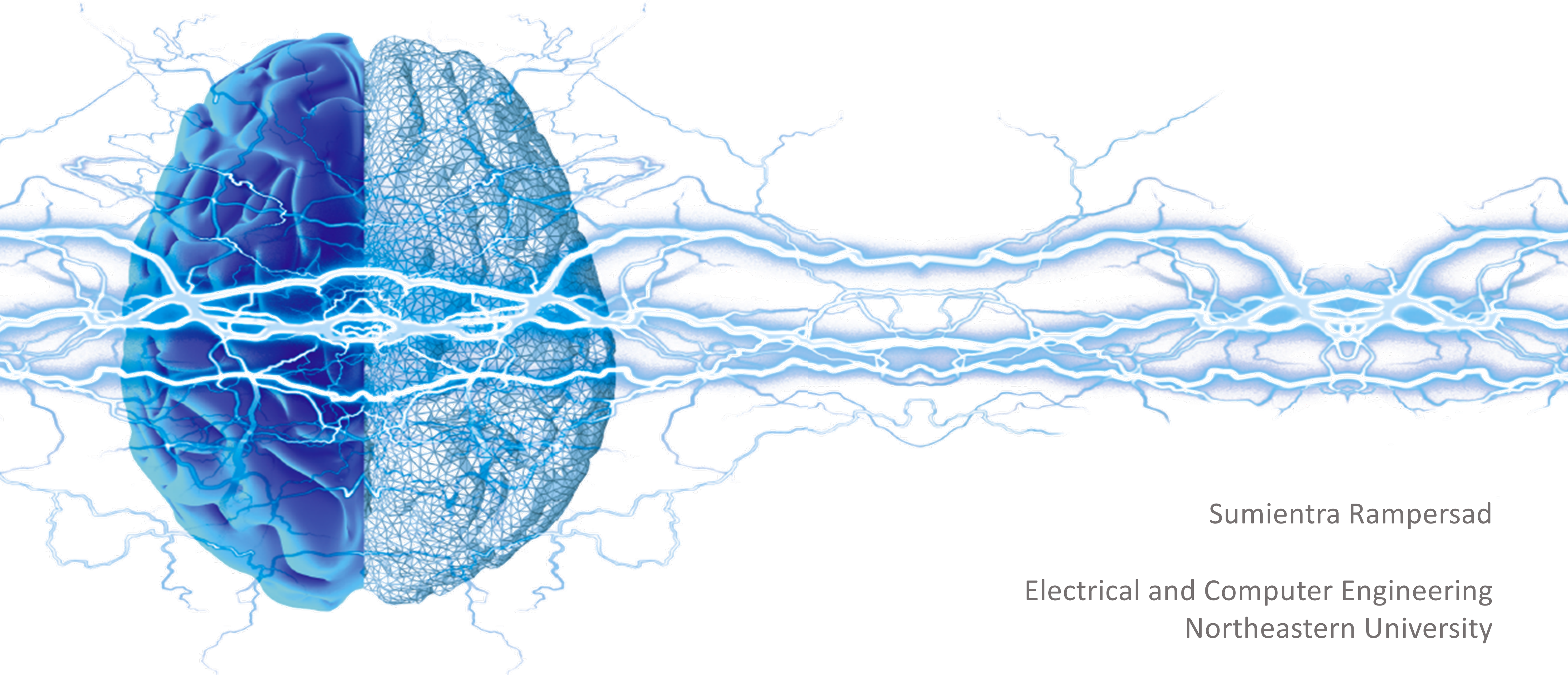
Uncertainty Quantification for Simulations of Neuromodulation



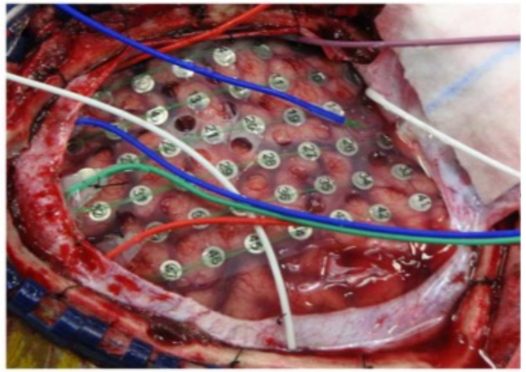
Sumientra Rampersad

Electrical and Computer Engineering
Northeastern University

Transcranial direct current stimulation (tDCS)

- Electrodes on scalp
- ≤ 2 mA for 10-30 min
- Applications:
    - Rehabilitation
    - Depression
    - Working memory

Electrocorticography (ECoG) stimulation

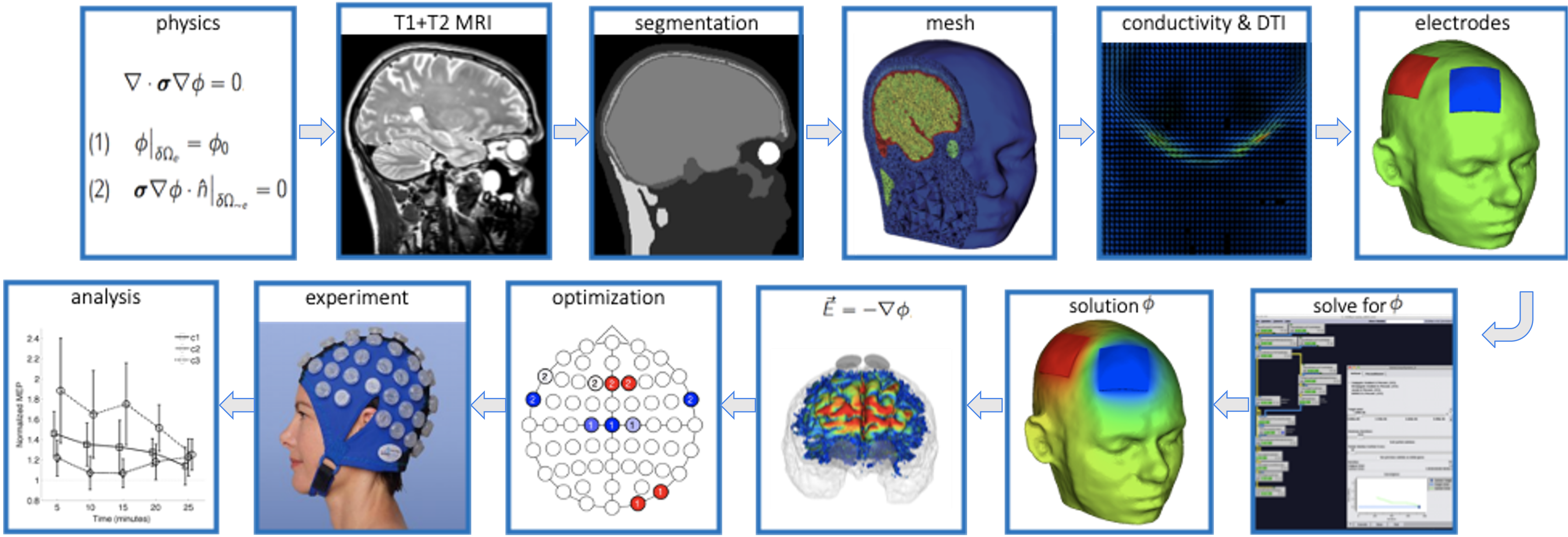


- Electrodes on cortex
- 1-10 mA pulses
- Applications:
    - Clinical mapping of cortical regions
    - Approved therapy for epilepsy
    - Brain-computer interfaces

tDCS vs ECoG stimulation

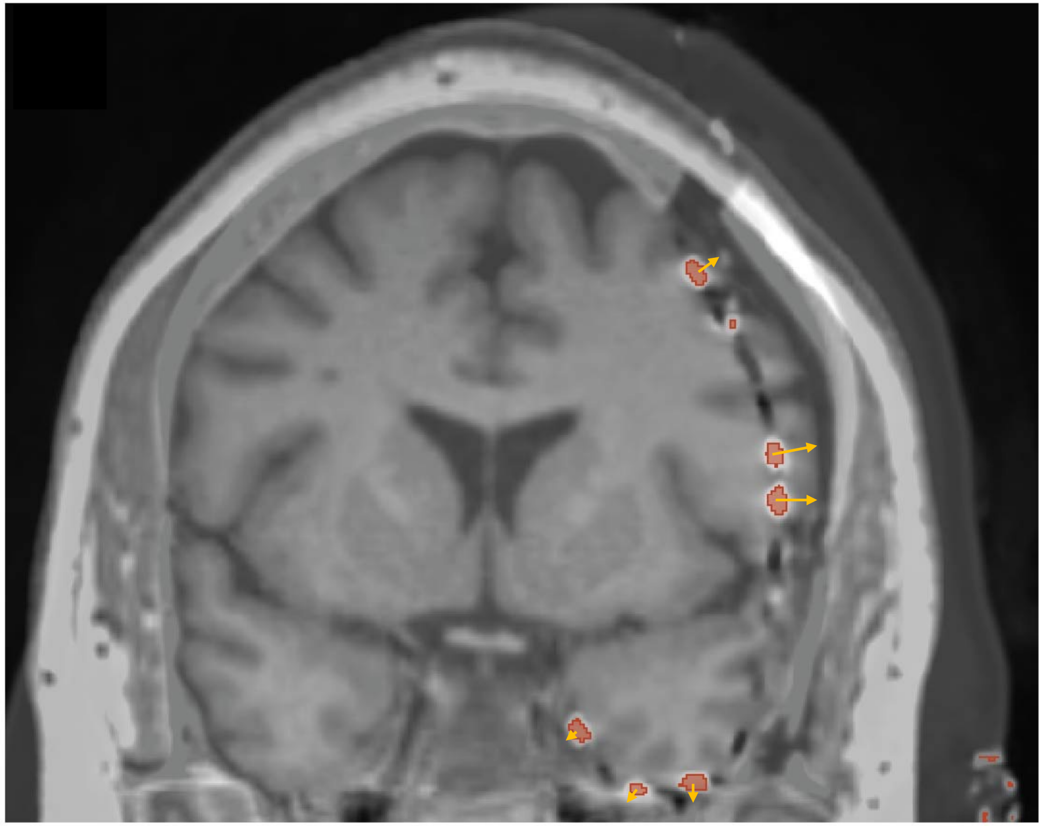
tDCS	ECoG
Noninvasive	Invasive
Easy, cheap, minor side effects	Risks due to surgery
Precise targeting difficult	Precise targeting difficult for deep regions
Various positive effects on brain activity	
Variable effects	
Mechanisms not fully known	
Simulation and optimization can improve understanding and targeting	

Improving stimulation through simulation



Uncertainties in simulations of brain stimulation

- Geometry
  - Cortical shape
  - CSF depth
- Tissue conductivities
  - Temperature
  - Frequency
  - Individual
  - Local
- Electrode locations
  - Imaging resolution
  - Brain shift



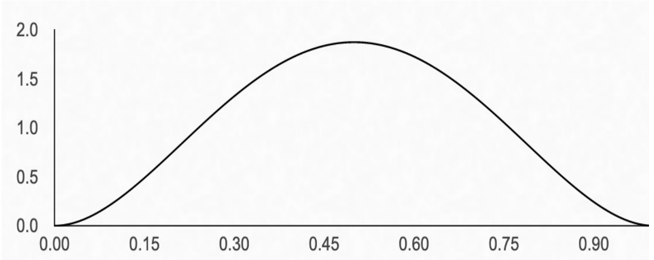
Hastreiter *et al*. 2004, Dalal *et al*. 2008, Hill *et al*. 2000.

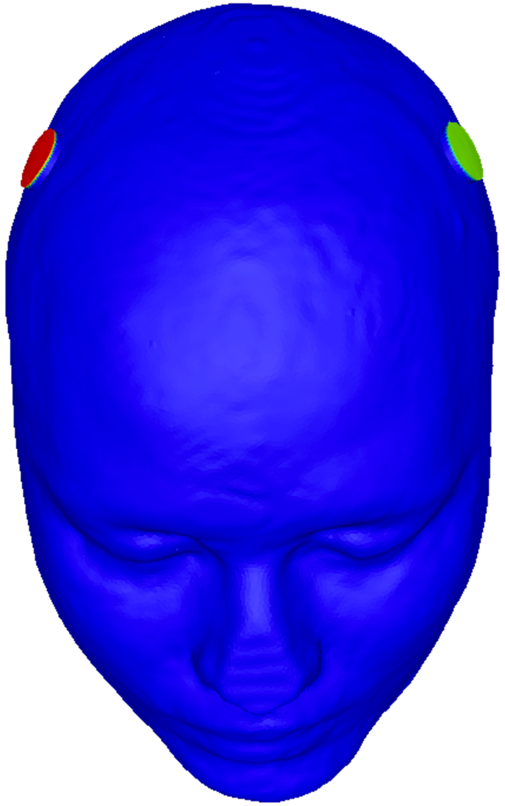
Quantifying effects of uncertainties in tDCS simulations

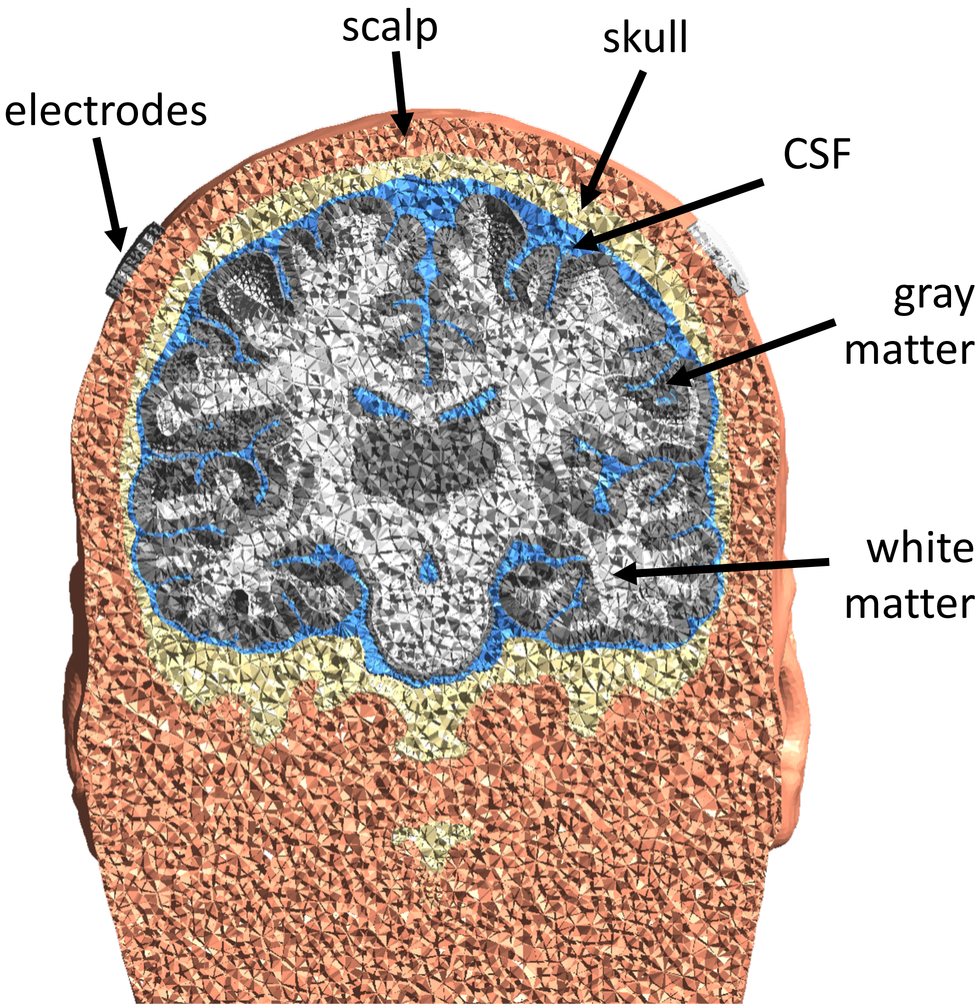
- Simulate |E| for tDCS with 2 electrodes at 1 mA in SCIRun
- Model tissue conductivities in UncertainSCI:

Tissue	Conductivity (S/m)
White matter	0.09 – 0.29
Gray matter	0.22 – 0.67
CSF	1.7696 – 1.8104
Skull	0.0016 – 0.33
Skin	0.28 – 0.87

- Beta distribution with α = β = 3







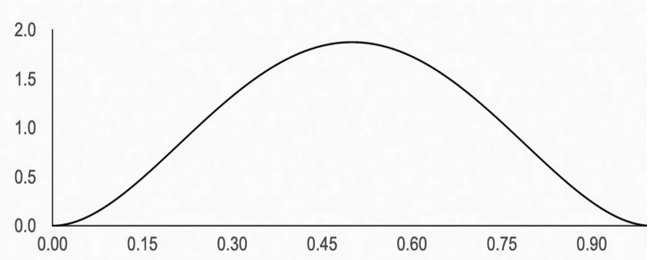
Vorwerk et al., Frontiers in Neuroscience, 2019

Quantifying effects of uncertainties in tDCS simulations

- Simulate |E| for tDCS with 2 electrodes at 1 mA in SCIRun
- Model tissue conductivities in UncertainSCI:

Tissue	Conductivity (S/m)
White matter	0.09 – 0.29
Gray matter	0.22 – 0.67
CSF	1.7696 – 1.8104
Skull	0.0016 – 0.33
Skin	0.28 – 0.87

- Beta distribution with α = β = 3



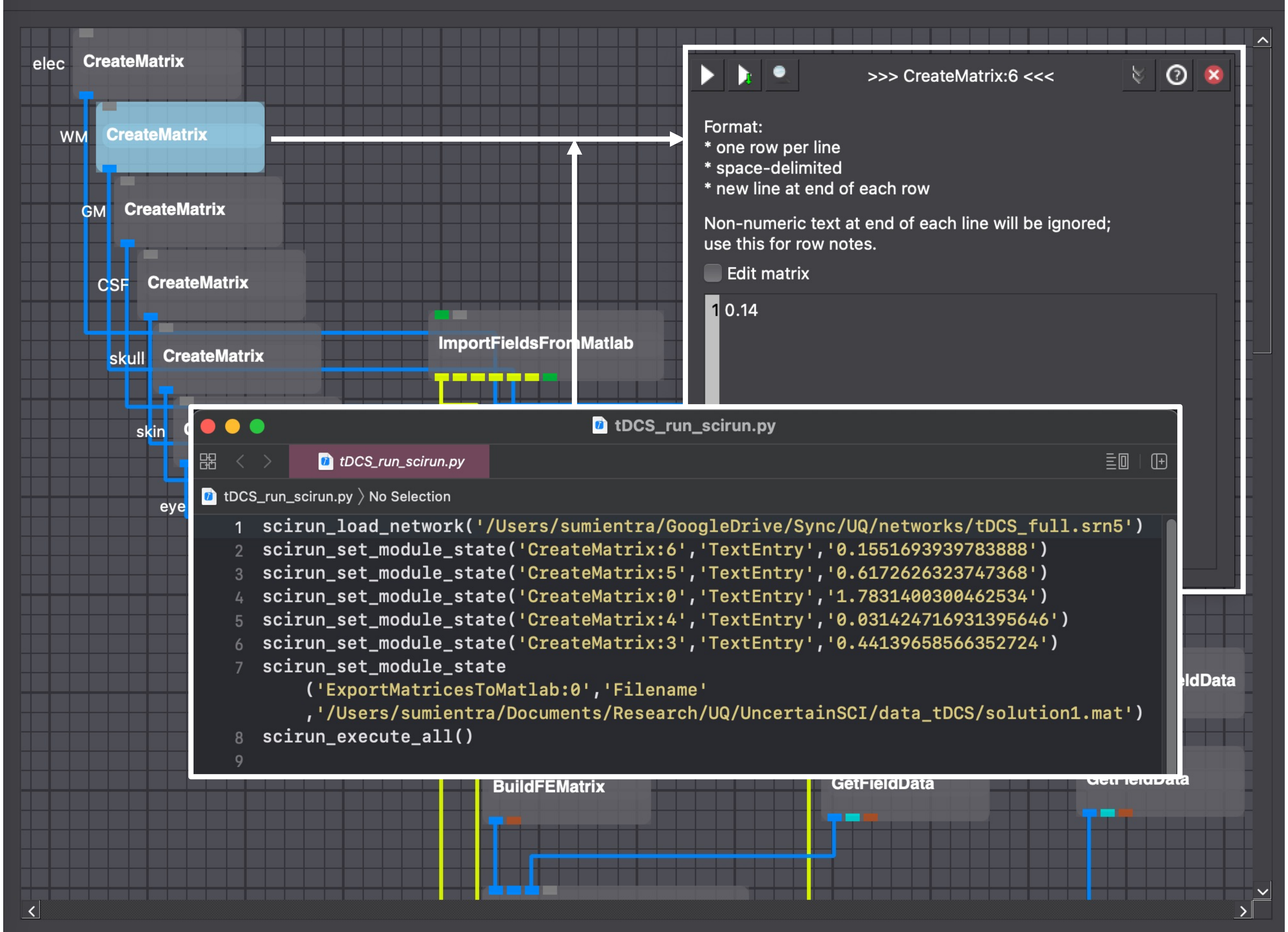

```
## Setup distributions
cond_range_WM    = np.resize(np.array([0.09, 0.290]), [2, 1])  # min and max of
cond_range_GM    = np.resize(np.array([0.22, 0.67]), [2, 1])   # min and max of
cond_range_CSF   = np.resize(np.array([1.7696, 1.79]), [2, 1]) # min and max of
cond_range_skull = np.resize(np.array([0.016, 0.033]), [2, 1]) # min and max of
cond_range_skin  = np.resize(np.array([0.28, 0.87]), [2, 1])   # min and max of
alpha = 3.  # input to beta distribution
beta = 3.   # input to beta distribution
cond_WM     = BetaDistribution(alpha=alpha, beta=beta, domain=cond_range_WM)
cond_GM     = BetaDistribution(alpha=alpha, beta=beta, domain=cond_range_GM)
cond_CSF    = BetaDistribution(alpha=alpha, beta=beta, domain=cond_range_CSF)
cond_skull  = BetaDistribution(alpha=alpha, beta=beta, domain=cond_range_skull)
cond_skin   = BetaDistribution(alpha=alpha, beta=beta, domain=cond_range_skin)
dist = TensorialDistribution(distributions = [cond_WM, cond_GM, cond_CSF, cond_s

## Initialize PCE object
dimension = 5   # number of parameters
order = 8       # polynomial order
indices = TotalDegreeSet(dim=dimension, order=order)
pce = PolynomialChaosExpansion(indices, dist)
```

```python
## Produce samples
if os.path.exists(output_dir+'parameters.txt'):      # if parameter_file exists, load it and continue
    print('Loading samples from file')
    pce.samples = np.loadtxt(output_dir+'parameters.txt')
else:                                                # if parameter_file does not exist, create parameters and save
    print('Generating samples')
    pce.generate_samples()
    np.savetxt(output_dir+'parameters.txt',pce.samples,delimiter = ' ')
print(pce.samples)

## Compute PCE (runs SCIRun)
print('Evaluating model at samples')
N_output = 4176987  # number of data points in SCIRun output = n... sh
model_output = np.zeros([pce.samples.shape[0], N_output]) # ... utputs
for ind in range(pce.samples.shape[0]): # loop over all samples, run SCIRun and store the outputs
    model_output[ind,:] = run_SCIRun(pce.samples[ind,:], ind, output_dir)
pce.build(model_output = model_output)
```

pce.samples = #iterations x dimension

#iterations x #elements

```python
## Function to run SCIRun network
def run_SCIRun(params,ind,output_dir):
  # Set paths
  scirun_call = '/Applications/SCIRun5.app/Contents/MacOS/SCIRun -0 -S '
  scirun_net  = '/Users/sumientra/GoogleDrive/Sync/UQ/networks/tDCS_full.srn5'
  output_file = output_dir+'solution'+str(ind+1)+'.mat'

  if os.path.exists(output_file):     # if output_file exists, load it and skip this iteration
      data     = sio.loadmat(output_file)
      solution = data.get('matrixInput1')
      solution = np.resize(solution, solution.shape[0])

  else:                               # if output_file does not exist, run SCIRun
      # Get conductivity parameters for this iteration
      WM    = str(params[0])    # get WM conductivity for this iteration
      GM    = str(params[1])    # get GM conductivity for this iteration
      CSF   = str(params[2])    # get CSF conductivity for this iteration
      skull = str(params[3])    # get skull conductivity for this iteration
      skin  = str(params[4])    # get skin conductivity for this iteration

      # Write python file that will prep and run SCIRun
      scirun_file = output_dir+'tDCS_run_scirun.py'     # set file name
      scirun_file_write = open(scirun_file,'w+')        # open file for writing
      scirun_file_write.write("scirun_load_network('"+scirun_net+"')\n") # load SCIRun network
      scirun_file_write.write("scirun_set_module_state('CreateMatrix:6','TextEntry','"+WM+"')\n")        # write WM conductivity into network
      scirun_file_write.write("scirun_set_module_state('CreateMatrix:5','TextEntry','"+GM+"')\n")        # write GM conductivity into network
      scirun_file_write.write("scirun_set_module_state('CreateMatrix:0','TextEntry','"+CSF+"')\n")       # write CSF conductivity into network
      scirun_file_write.write("scirun_set_module_state('CreateMatrix:4','TextEntry','"+skull+"')\n")     # write skull conductivity into network
      scirun_file_write.write("scirun_set_module_state('CreateMatrix:3','TextEntry','"+skin+"')\n")      # write skin conductivity into network
      scirun_file_write.write("scirun_set_module_state('ExportMatricesToMatlab:0','Filename','"+output_file+"')\n") # write file name into network
      scirun_file_write.write("scirun_execute_all()\n") # execute SCIRun network
      scirun_file_write.close()                         # close python file

      # Run SCIRun
      os.system(scirun_call+scirun_file)    # execute SCIRun python file
      data     = sio.loadmat(output_file)   # load SCIRun output
      solution = data.get('matrixInput1')
      solution = np.resize(solution, solution.shape[0])

  return solution                         # return this iteration's solution to UncertainSCI
```


elec
CreateMatrix
WM
CreateMatrix
GM
CreateMatrix
CSF
CreateMatrix
skull
CreateMatrix
skin
eye
ImportFieldsFromMatlab
>>> CreateMatrix:6 <<<
Format:
* one row per line
* space-delimited
* new line at end of each row
Non-numeric text at end of each line will be ignored;
use this for row notes.
Edit matrix
1 0.14
tDCS_run_scirun.py
tDCS_run_scirun.py
tDCS_run_scirun.py 〉 No Selection
scirun_load_network('/Users/sumientra/GoogleDrive/Sync/UQ/networks/tDCS_full.srn5')
scirun_set_module_state('CreateMatrix:6','TextEntry','0.1551693939783888')
scirun_set_module_state('CreateMatrix:5','TextEntry','0.6172626323747368')
scirun_set_module_state('CreateMatrix:0','TextEntry','1.7831400300462534')
scirun_set_module_state('CreateMatrix:4','TextEntry','0.031424716931395646')
scirun_set_module_state('CreateMatrix:3','TextEntry','0.44139658566352724')
scirun_set_module_state
('ExportMatricesToMatlab:0','Filename'
,'/Users/sumientra/Documents/Research/UQ/UncertainSCI/data_tDCS/solution1.mat')
scirun_execute_all()
BuildFEMatrix
GetFieldData
GetFieldData
ldData

```python
## Function to run SCIRun network
def run_SCIRun(params,ind,output_dir):
  # Set paths
  scirun_call = '/Applications/SCIRun5.app/Contents/MacOS/SCIRun -0 -S '
  scirun_net  = '/Users/sumientra/Google
  output_file = output_dir+'solution'+st

  if os.path.exists(output_file):      #
      data     = sio.loadmat(output_file
      solution = data.get('matrixInput1'
      solution = np.resize(solution, sol

  else:                                #
      # Get conductivity parameters for
      WM    = str(params[0])    # get WM
      GM    = str(params[1])    # get GM
      CSF   = str(params[2])    # get CS
      skull = str(params[3])    # get sk
      skin  = str(params[4])    # get sk

      # Write python file that will prep and run SCIRun
      scirun_file = output_dir+'tDCS_run_scirun.py'      # set file name
      scirun_file_write = open(scirun_file,'w+')         # open file for writing
      scirun_file_write.write("scirun_load_network('"+scirun_net+"')\n") # load SCIRun network
      scirun_file_write.write("scirun_set_module_state('CreateMatrix:6','TextEntry','"+WM+"')\n")       # write WM conductivity into network
      scirun_file_write.write("scirun_set_module_state('CreateMatrix:5','TextEntry','"+GM+"')\n")       # write GM conductivity into network
      scirun_file_write.write("scirun_set_module_state('CreateMatrix:0','TextEntry','"+CSF+"')\n")      # write CSF conductivity into network
      scirun_file_write.write("scirun_set_module_state('CreateMatrix:4','TextEntry','"+skull+"')\n")    # write skull conductivity into network
      scirun_file_write.write("scirun_set_module_state('CreateMatrix:3','TextEntry','"+skin+"')\n")     # write skin conductivity into network
      scirun_file_write.write("scirun_set_module_state('ExportMatricesToMatlab:0','Filename','"+output_file+"')\n") # write file name into network
      scirun_file_write.write("scirun_execute_all()\n") # execute SCIRun network
      scirun_file_write.close()                          # close python file

      # Run SCIRun
      os.system(scirun_call+scirun_file)     # execute SCIRun python file
      data     = sio.loadmat(output_file)    # load SCIRun output
      solution = data.get('matrixInput1')
      solution = np.resize(solution, solution.shape[0])

  return solution                       # return this iteration's solution to UncertainSCI
```

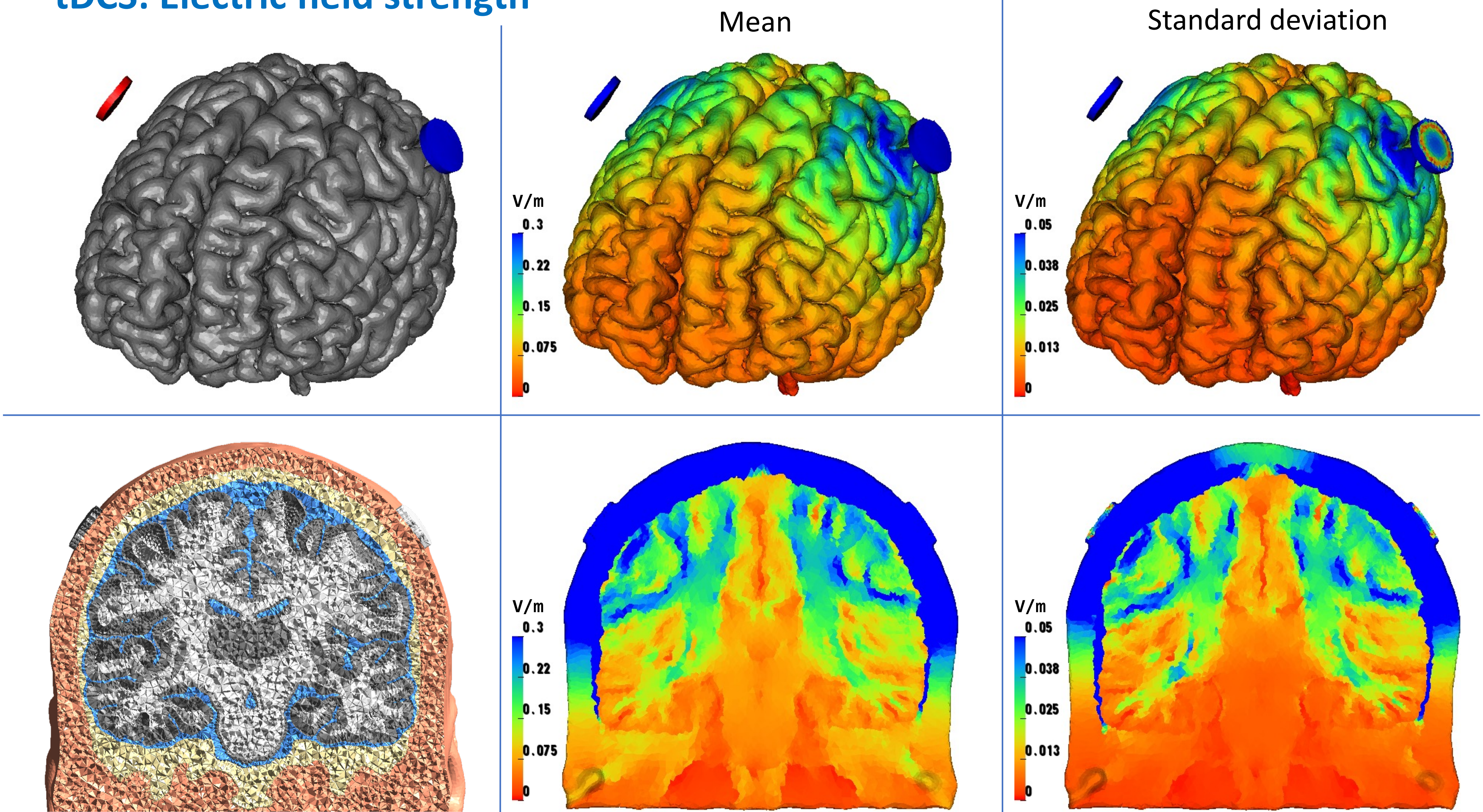
tDCS_run_scirun.py

tDCS_run_scirun.py 〉 No Selection

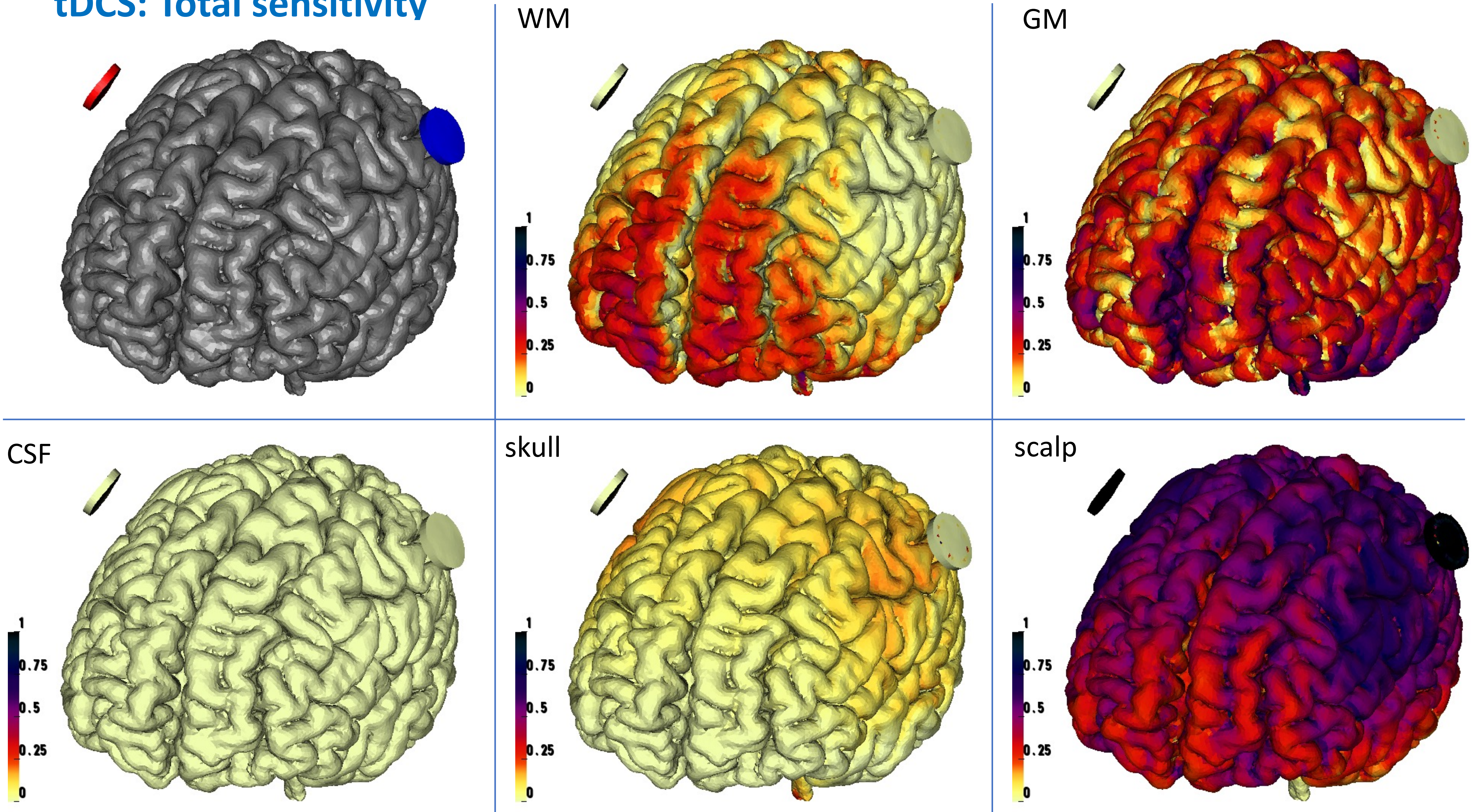
```python
scirun_load_network('/Users/sumientra/GoogleDrive/Sync/UQ/networks/tDCS_full.srn5')
scirun_set_module_state('CreateMatrix:6','TextEntry','0.1551693939783888')
scirun_set_module_state('CreateMatrix:5','TextEntry','0.6172626323747368')
scirun_set_module_state('CreateMatrix:0','TextEntry','1.7831400300462534')
scirun_set_module_state('CreateMatrix:4','TextEntry','0.031424716931395646')
scirun_set_module_state('CreateMatrix:3','TextEntry','0.44139658566352724')
scirun_set_module_state
    ('ExportMatricesToMatlab:0','Filename'
    ,'/Users/sumientra/Documents/Research/UQ/UncertainSCI/data_tDCS/solution1.mat')
scirun_execute_all()
```

```python
## Postprocess PCE
print('Starting postprocessing')
mean    = pce.mean() # calculate mean of all SCIRun outputs
stdev   = pce.stdev() # calculate standard deviation of all SCIRun outputs

# Sensitivities
total_sensitivity   = pce.total_sensitivity() # calculate total sensitivity for each parameter
global_interactions = list(chain.from_iterable(combinations(range(dimension), r) for r in range(1, dimension+1))) # get
global_sensitivity  = pce.global_sensitivity(global_interactions) # calculate global sensitivity for each interaction
global_labels       = ['[' + ' '.join(str(elem) for elem in [i+1 for i in item]) + ']' for item in global_interactions]

# Quantiles
quantile_levels = np.array([0.05, 0.5, 0.95]) # select levels at which to calculate quantiles
quantiles       = pce.quantile(quantile_levels, M=int(2e3)) # calculate quantiles

# Save data to Matlab file
matlab_file = output_dir+'data.mat'
sio.savemat(matlab_file,{'data_mean': mean,
                         'data_std': stdev,
                         'quantiles': quantiles,
                         'quantile_levels': quantile_levels,
                         'total_sensitivity': total_sensitivity,
                         'global_sensitivity': global_sensitivity,
                         'global_interactions': global_labels,
                         'samples': pce.samples,
                         'solutions': model_output})
```

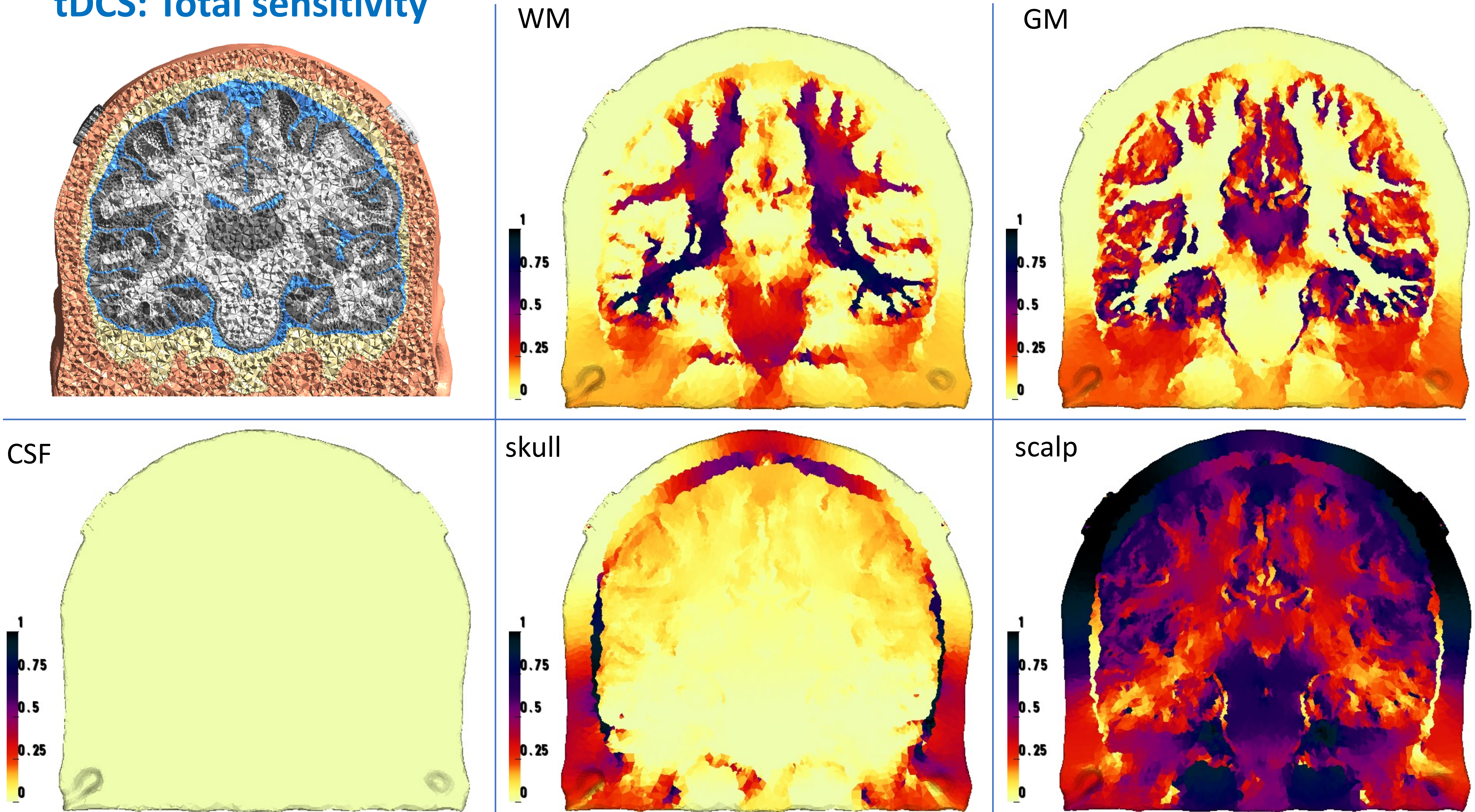
tDCS: Electric field strength



tDCS: Total sensitivity



tDCS: Total sensitivity

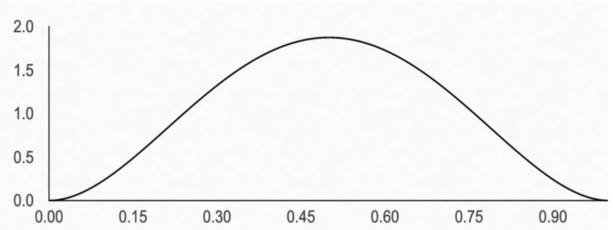


Quantifying effects of uncertainties in ECoG stimulation simulations

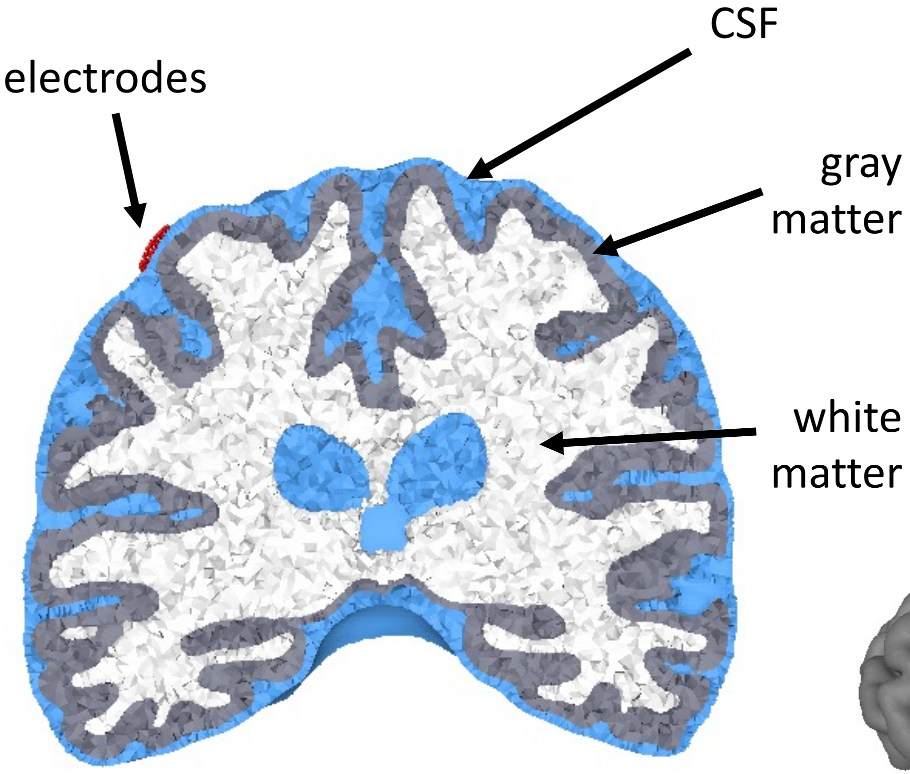
- Simulate V for ECoG with 2 electrodes at 0.75 mA in SCIRun
- Tissue conductivities

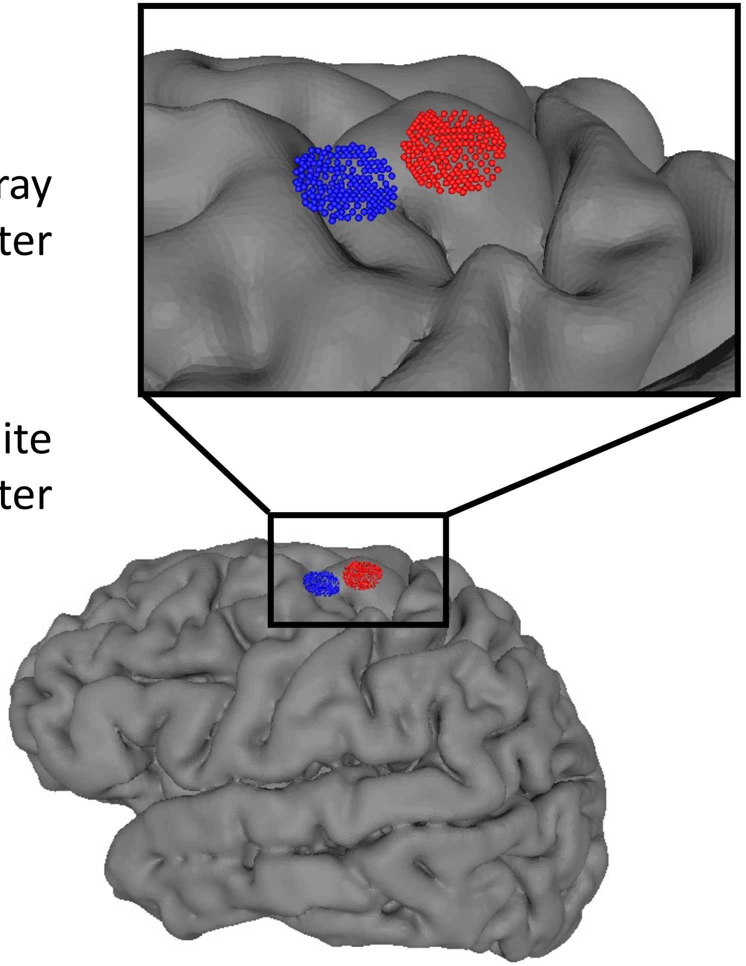
Tissue	Conductivity (S/m)
CSF	1.7696 – 1.8104
Gray matter	0.22 – 0.67
White matter	0.09 – 0.29

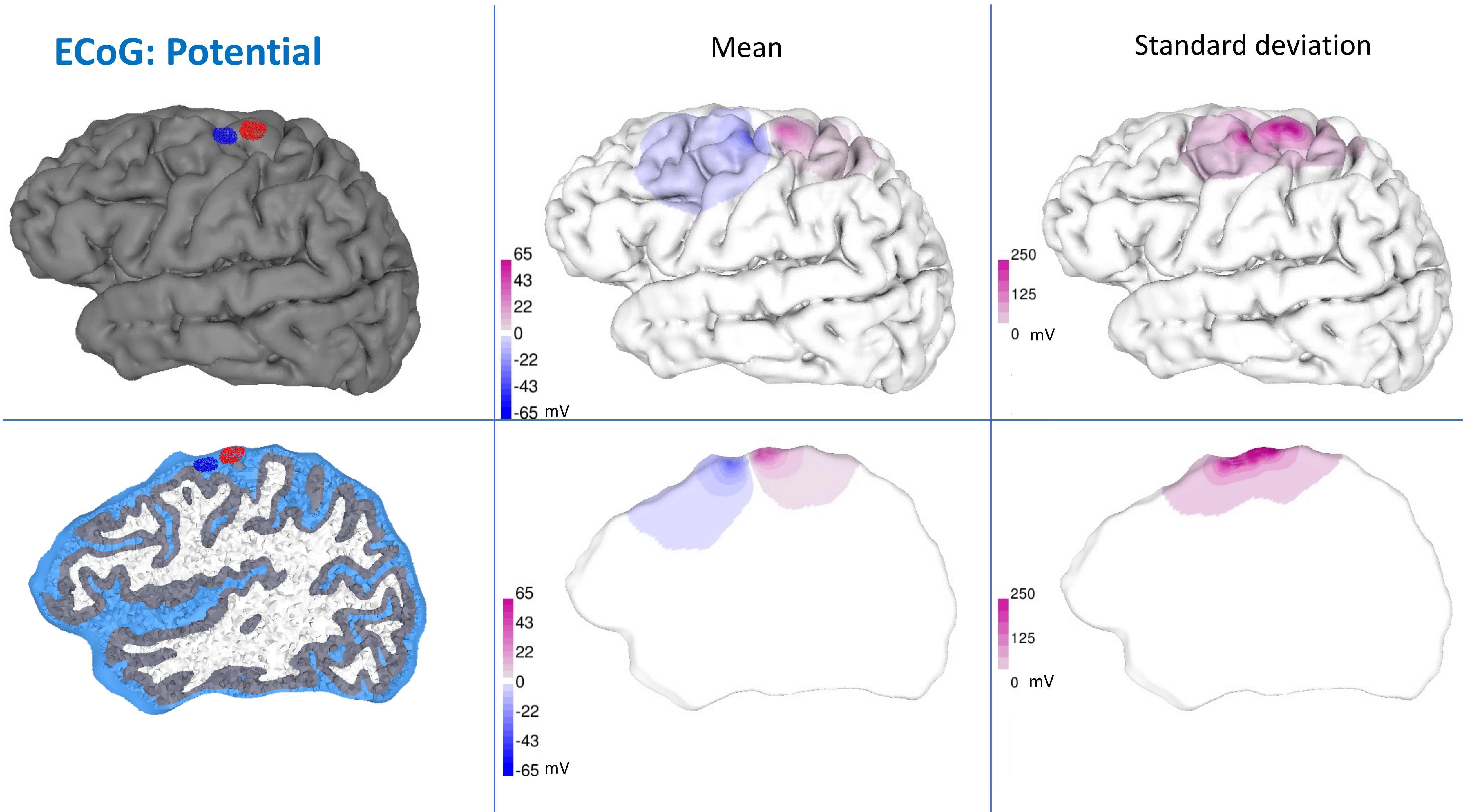
  - Beta distribution with $\alpha = \beta = 3$



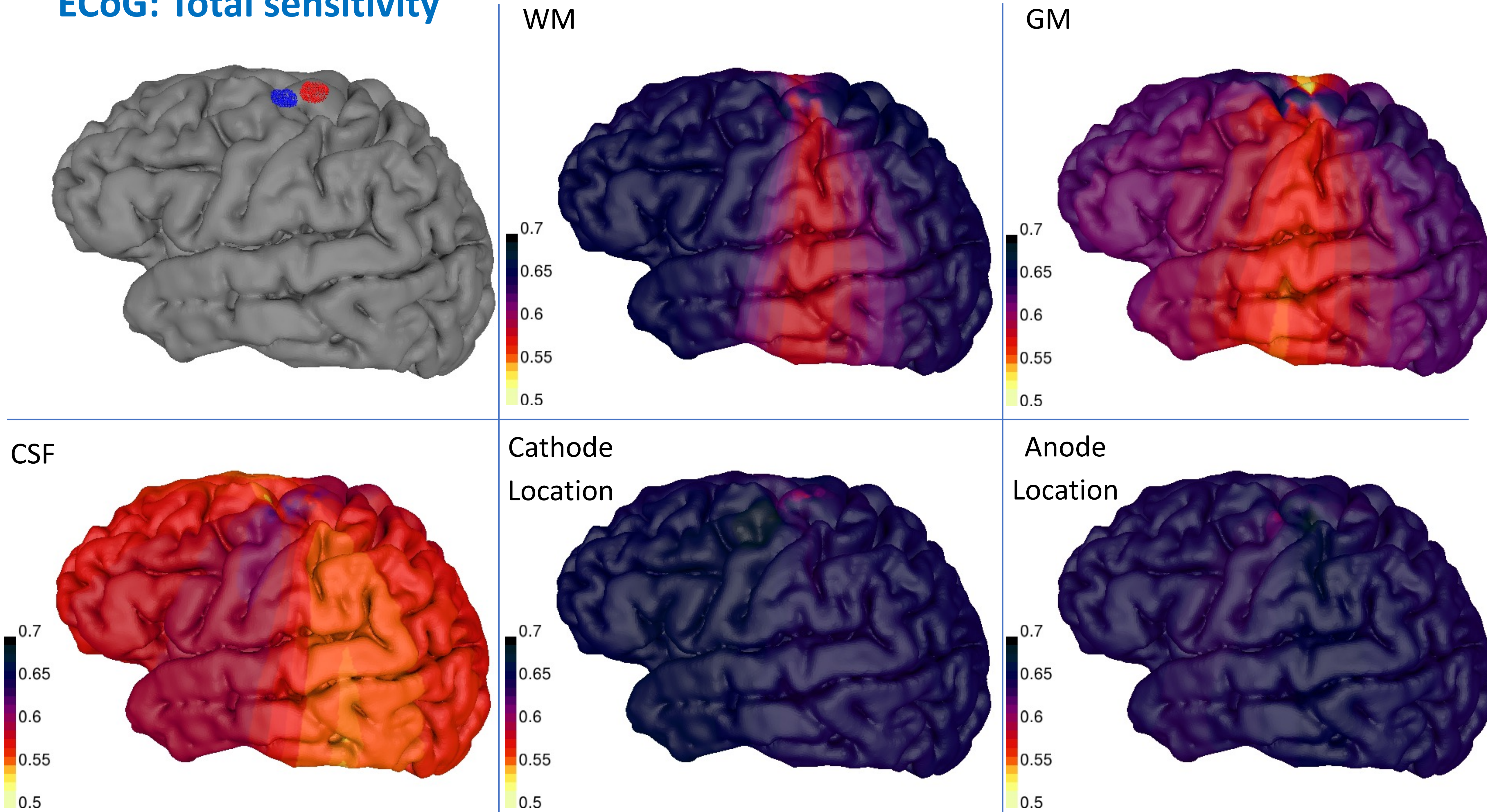
- Electrode locations
  - Cathode location
  - Anode location
  - Uniform distribution of point source nodes



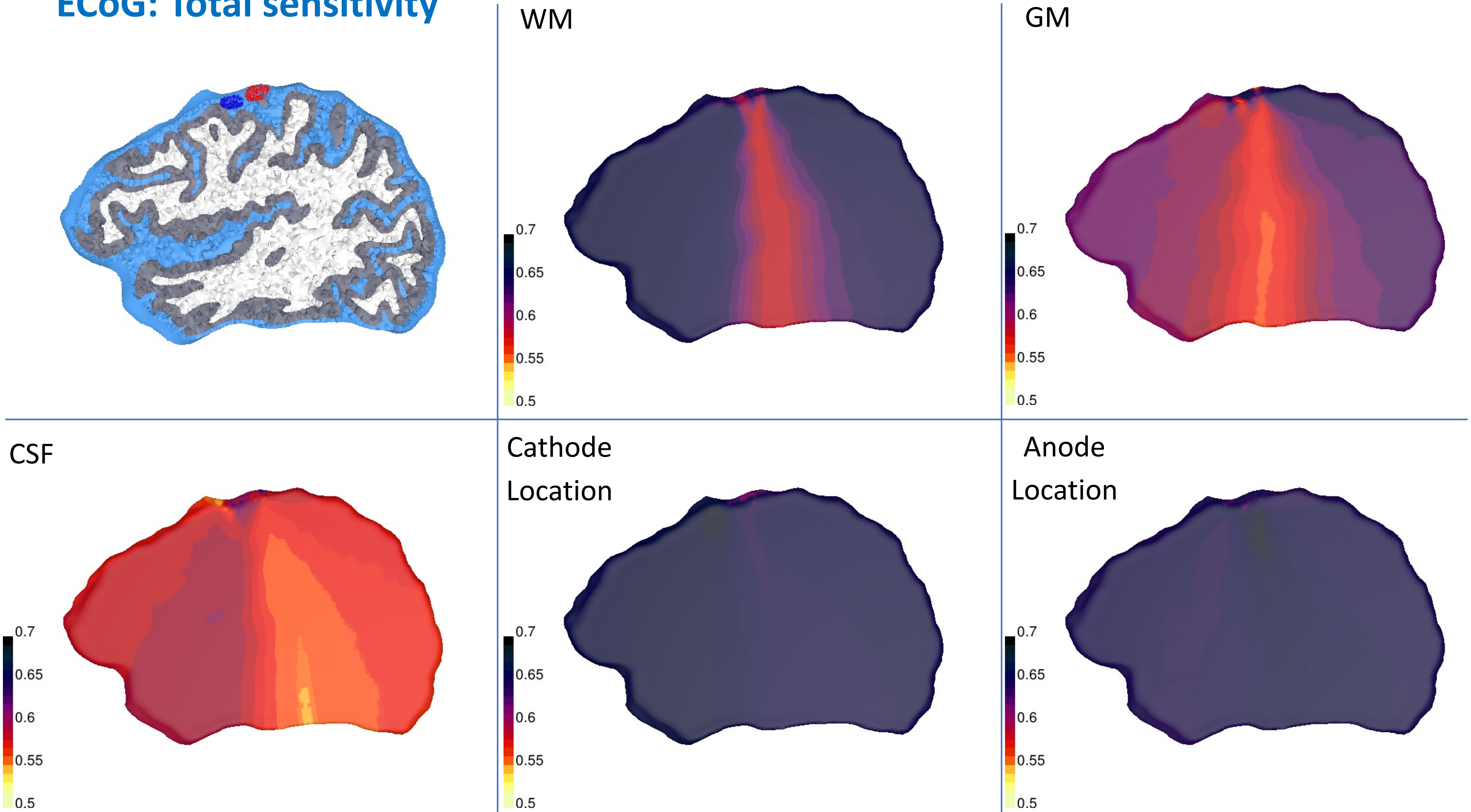



ECoG: Potential
Mean
Standard deviation
65
43
22
0
-22
-43
-65 mV
250
125
0 mV
65
43
22
0
-22
-43
-65 mV
250
125
0 mV

ECoG: Total sensitivity



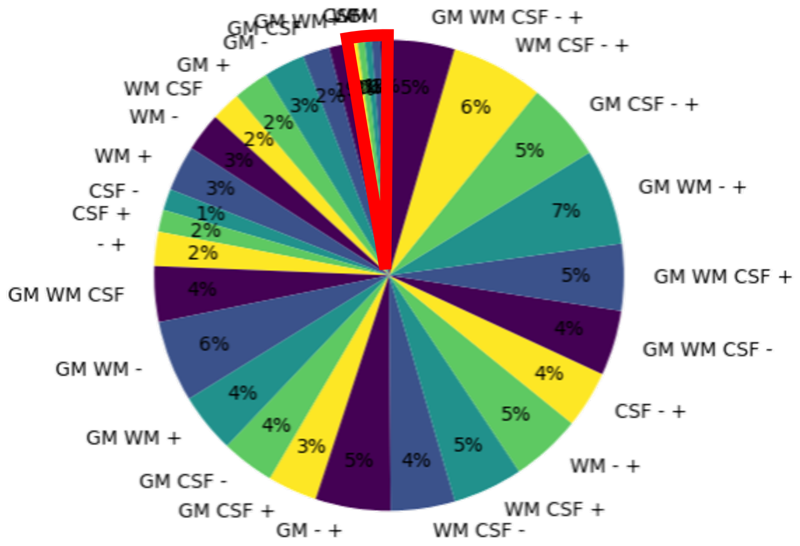
ECoG: Total sensitivity

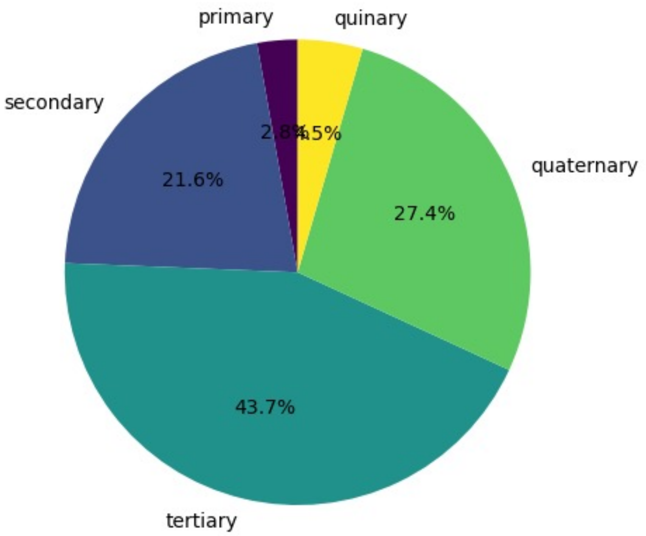


Global sensitivity

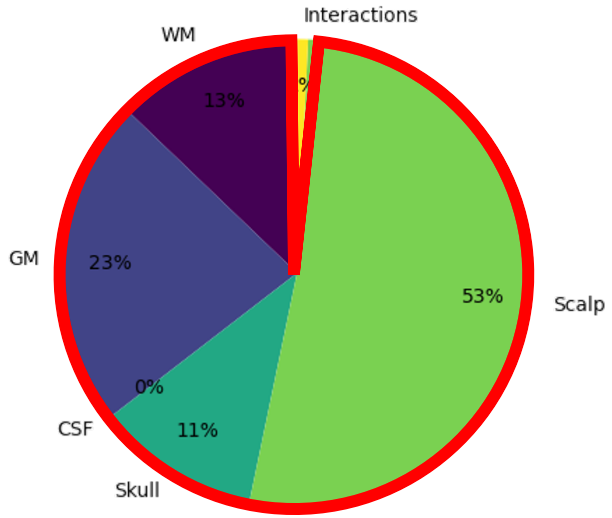
Total mesh

ECoG





tDCS

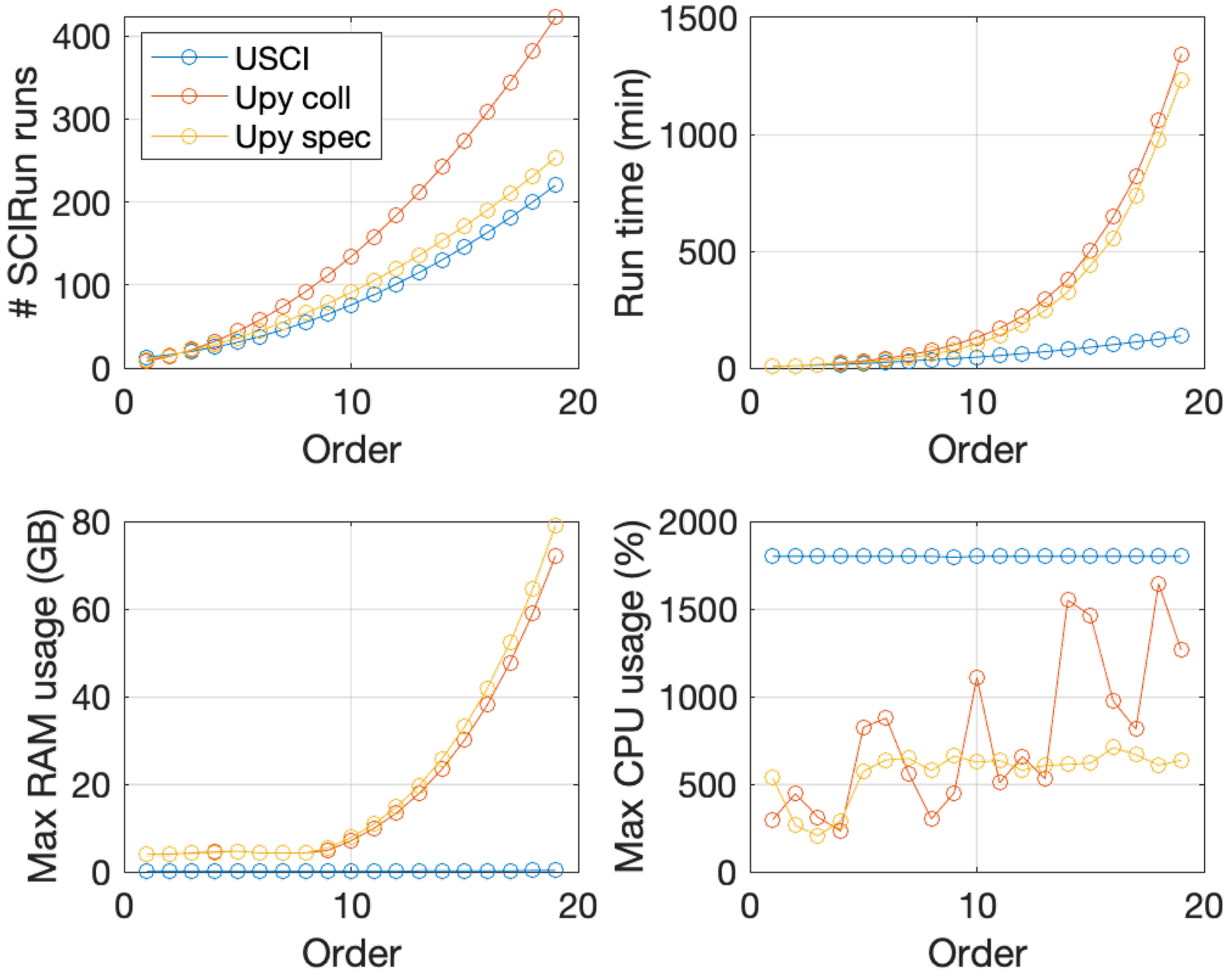


Comparison to other software

Uncertainpy

- Free python-based software
- Polynomial chaos
  - Collocation / spectral
- Connects to other software
- Limited model size
  - Full model: 4.2M elements
  - Reduced model: 48k elements

https://uncertainpy.readthedocs.io



Conclusions

- UncertainSCI accurately and efficiently quantifies uncertainties in simulations of brain stimulation.
- Simulations of tDCS are mainly affected by scalp and GM conductivity.
- Simulations of ECoG stimulation are strongly affected by anode and cathode location, with many interactions with conductivities.
- Future work will investigate the effects of cortical and CSF geometry, white matter anisotropy, and electrode location for these and other stimulation modalities.