Hindawi Publishing Corporation
Journal of Computational Medicine
Volume 2016, Article ID 9826596, 15 pages
http://dx.doi.org/10.1155/2016/9826596

Research Article

An Object-Oriented Framework for Versatile Finite Element Based Simulations of Neurostimulation

Edward T. Dougherty[1] and James C. Turner[2]

[1]*Mathematics Department, Rowan University, Glassboro, NJ 08028, USA*
[2]*Mathematics Department, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA*

Correspondence should be addressed to Edward T. Dougherty; doughertye@rowan.edu

Received 28 May 2015; Revised 27 August 2015; Accepted 21 October 2015

Academic Editor: Camillo Porcaro

Computational simulations of transcranial electrical stimulation (TES) are commonly utilized by the neurostimulation community, and while vastly different TES application areas can be investigated, the mathematical equations and physiological characteristics that govern this research are identical. The goal of this work was to develop a robust software framework for TES that efficiently supports the spectrum of computational simulations routinely utilized by the TES community and in addition easily extends to support alternative neurostimulation research objectives. Using well-established object-oriented software engineering techniques, we have designed a software framework based upon the physical and computational aspects of TES. The framework's versatility is demonstrated with a set of diverse neurostimulation simulations that (i) reinforce the importance of using anisotropic tissue conductivities, (ii) demonstrate the enhanced precision of high-definition stimulation electrodes, and (iii) highlight the benefits of utilizing multigrid solution algorithms. Our approaches result in a framework that facilitates rapid prototyping of real-world, customized TES administrations and supports virtually any clinical, biomedical, or computational aspect of this treatment. Software reuse and maintainability are optimized, and in addition, the same code can be effortlessly augmented to provide support for alternative neurostimulation research endeavors.

1. Introduction

Transcranial electrical stimulation (TES) is a collection of noninvasive neurostimulation techniques that strategically modulate activity in regions of the brain with low magnitude electric current delivered through electrodes positioned on the scalp surface. Forms of TES include the commonly used transcranial direct current stimulation (tDCS), as well as transcranial alternating current stimulation (tACS) [1]. Recently, the use of numerous smaller sized electrodes, termed high-definition tDCS (HD-tDCS), has emerged as a form of TES that enhances electrical current focality [2, 3]. Clinical and biomedical research continue to demonstrate the capabilities of TES as a medical treatment. For example, Alzheimer and Parkinson's disease patients have demonstrated increased memory abilities [4–6]. In addition, TES has shown to alleviate symptoms of psychiatric disorders including depression [7, 8] and schizophrenia [9–11].

The efficacy and comprehension of TES have been enhanced with mathematical modeling and computational simulation. In particular, simulations can compute the current density distribution for a given patient and TES apparatus configuration [1, 12–15] and have demonstrated the importance of modulating treatment stimulation dosage [16]. Other simulations have illustrated the importance of incorporating anisotropic tissue conductivity data [17, 18], and numerical studies aid in identifying the computational solution methods most efficient in simulating TES mathematical models [19].

Object-oriented design is a software design approach that defines objects, which are simply software entities that encapsulate data and functionality and the relationships among objects. Features of object-oriented design can be utilized to maximize code reusability, simplify software maintenance, and promote application versatility [20]. The prevalence of object-oriented design in scientific and biomedical

applications began in the 1990s, and its use has dramatically increased due to its advantages over procedural software implementations [21–26]. In particular, mathematical and physical attributes of a model can be naturally represented with objects [27, 28].

A common approach for simulating partial differential equation (PDE) based mathematical models is to use prebuilt simulation software programs. These "black-boxes" can simplify model implementation; however, there are limitations with this approach. A researcher is often confined to the numerical algorithms and programming controls offered by the simulation program; it can be very difficult, or perhaps not possible, to incorporate numerics and solution techniques not supported by the software. In addition, integrating a prebuilt software application with external data sources and applications can be very challenging, which obstructs the use of its simulations within larger software solutions.

An alternative strategy is to create custom software that utilizes numerical application programming interfaces (APIs). While this approach generally takes longer to implement, it allows the use of problem-specific algorithms and programming logic that facilitate accurate and expedient simulation results [28]. Coupling this philosophy with object-oriented software engineering techniques can produce a final product that is versatile, expandable, and computationally efficient. Interactions with external systems can be seamlessly integrated, and the ability to compile the software to executable machine code simplifies its deployment to alternative hardware platforms, for example, medical imaging machines.

In this paper, we present an object-oriented software framework for finite element based TES simulations. Multiple features of object-oriented design and programming are utilized to create a modular software architecture that encapsulates medical, mathematical, and computational attributes of TES. The result is a program that maximizes code reuse and TES simulation versatility. We demonstrate this versatility with several simulations, each with a distinct TES research focus. These simulations utilize MRI-derived three-dimensional head models, with physiologically based tissue conductivities and real-world electrode configurations. Finally, we show how the same software can be easily extended to address alternative neurostimulation research areas, such as deep brain stimulation (DBS). In addition, all components of the framework are available to the community as a supplement to this paper.

2. Materials and Methods

2.1. Governing Equations. The electric current density within the head and brain from neurostimulation can be modeled by the Poisson equation; namely, $-\nabla \cdot \mathbf{M}\nabla\Phi = f(\vec{x})$, where Φ is the electric potential, $\mathbf{M}$ is the tissue conductivity tensor, and $f(\vec{x})$ is a given electrical source term. For isotropic mediums, $\mathbf{M}$ can be represented as a scalar, which varies for different tissue types.

Electric current delivered by TES anode electrode(s) is given by the nonhomogeneous Neumann boundary condition $\vec{n} \cdot \mathbf{M}\nabla\Phi = I(\vec{x})$, where $I(\vec{x})$ represents the stimulation current density and $\vec{n}$ is the outward boundary normal vector. Cathode electrode(s) are represented by the homogeneous Dirichlet condition $\Phi(\vec{x}) = 0$. All other points on the skin surface are insulated by the surrounding air; thus $\vec{n} \cdot \mathbf{M}\nabla\Phi = 0$.

Our governing equations are as follows:

$$-\nabla \cdot \mathbf{M}\nabla\Phi = f(\vec{x}), \quad \vec{x} \in \Omega, \tag{1a}$$

$$\Phi = 0, \quad \vec{x} \in \partial\Omega_C, \tag{1b}$$

$$\vec{n} \cdot \mathbf{M}\nabla\Phi = I(\vec{x}), \quad \vec{x} \in \partial\Omega_A, \tag{1c}$$

$$\vec{n} \cdot \mathbf{M}\nabla\Phi = 0, \quad \vec{x} \in \partial\Omega_S, \tag{1d}$$

where Ω is the head volume, $\partial\Omega_A$ and $\partial\Omega_C$ represent the areas on the scalp covered by anode and cathode electrodes, respectively, and $\partial\Omega_S$ is the remaining portion of the head surface. Note that, for TES simulations, there is no source term within the volume, and so $f(\vec{x}) = 0$ (1a). Alternatively, DBS is realized with an appropriate definition of $f(\vec{x})$ and the homogeneous Neumann boundary condition (1d) applied to the entire boundary; namely, $\partial\Omega_s = \partial\Omega$ [29].

The associated weak formulation (see Appendix) is to find $\Phi(\vec{x}) \in H_0^1(\Omega)$ given $f(\vec{x}) \in L_2(\Omega)$ such that

$$\int_\Omega \nabla v \cdot \mathbf{M}\nabla\Phi \, dx = \int_{\partial\Omega_A} vI \, ds + \int_\Omega f v \, dx, \quad \forall v(\vec{x}) \in H_0^1(\Omega), \tag{2}$$

where

$$H_0^1(\Omega) = \left\{ u \mid u \in H^1(\Omega), \, u = 0 \ \forall \vec{x} \in \partial\Omega_C \right\},$$
$$H^1(\Omega) = \left\{ u \mid u \in L_2(\Omega), \, \frac{\partial u}{\partial x_i} \in L_2(\Omega) \right\}, \tag{3}$$
$$L_2(\Omega) = \left\{ p \mid \int_\Omega |p|^2 \, dx < \infty \right\}.$$

2.2. Framework Design. The fundamental task in the design of the framework was to create objects based on attributes of TES and TES simulations. We refer to [30] for a detailed explanation of object-oriented design and provide just a brief overview of the key aspects utilized within our framework.

Objects encapsulate data and functions that operate on the data. A *class* provides the description of an object type by defining variable and function names, and an object is more formally viewed as a specific instance of a class. *Inheritance* is a class relationship and provides the ability to extend the functionality of a class with a so-called *subclass*. A subclass can contain data and functionality from a parent class and can define its own as well. Inheritance is a major object-oriented design concept that exploits code reuse, since all subclasses can reuse data constructs and functions from a parent class. *Polymorphism* enables a subclass to give specific functionality to an abstract function defined by a parent class. This powerful technique allows parent classes to encapsulate generic and broad ideas by delegating specific function implementations to subclasses [30].

We required that electrode configuration, stimulation parameters, tissue conductivity information, computational

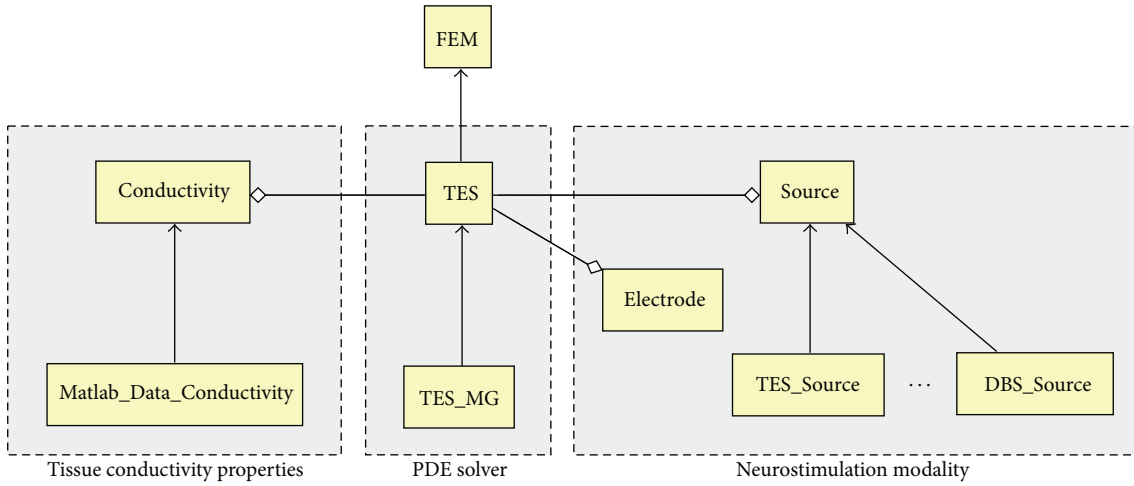


Figure 1: Software architecture and main classes in the object-oriented TES simulation framework. Classes are represented with boxes, and arrow and diamond tipped lines represent inheritance and aggregation, respectively.

domain, and numerical solution methods be specified via an input file. This permits customized TES simulations without the need to rewrite and recompile code. In addition, by mathematically retaining the source term, $f(\vec{x})$, in the weak formulation (2) and allowing this function to be defined by a user, different neurostimulation modalities, for example, DBS, can be simulated.

We choose to base our framework design on Diffpack [31], which is a numerical API library for solving PDEs. It is based on the C++ programming language [32] and possesses a vast collection of PDE solution algorithms [27, 28]. The C++ language incorporates well-established software engineering practices [33] and is compiled to machine code, resulting in fast execution speeds and portability to alternative hardware platforms. Despite our use of a specific numerical library and programming language, the object-oriented design and implementation strategies presented in this paper apply to any programming language and API package that supports an object-oriented approach.

Figure 1 displays the main software components in the TES simulation framework. Each box represents a class, and each arrow and diamond tipped line represents inheritance and aggregation, respectively. `FEM` is a class in the Diffpack library that offers fundamental finite element method data structures and algorithms. Class `TES`, which is the cornerstone of the framework, inherits `FEM` functionality to solve simulations given by system (1a)–(1d). Class `TES_MG` extends `TES` functionality so that multigrid (MG) algorithms can also be utilized in solving this system.

For TES simulations to be fully realized, tissue conductivity and electrode information are needed. These ideas have been encapsulated with respective classes, and `TES` possesses instances of each. Since conductivity data can come from a variety of sources with differing storage formats, the `Conductivity` class merely defines abstract function names that are viewed as common to all conductivity data sources. Then, specific functionality for specific conductivity data sources is implemented in subclasses, for example, `Matlab_Data_Conductivity`, via polymorphism. This approach allows different conductivity data sources to be incorporated into the framework without needing to modify code in the `TES` and `Conductivity` classes. The boundary conditions are also managed by `TES`, using information from class `Electrode`, which maintains TES electrode location and size information. In addition, by retaining the source term $f(\vec{x})$ in the weak formulation (2) and encapsulating it as a class, namely, `Source`, assignments to $f(\vec{x})$ enable the simulation of alternative types of neurostimulation, such as DBS.

2.3. Framework Implementation. In this section, key software implementation aspects of the framework and related Diffpack concepts are described. For a complete guide to Diffpack, see [27].

2.3.1. Tissue Conductivity. Class `Conductivity` defines general function names but delegates the implementation of these functions to its subclasses (Code 1). `MatlabConductivity` inherits `Conductivity` and provides specific implementations of the `loadConductivities` and `getConductivity` functions based on a Matlab [34] binary data file format produced by the SimNIBS software package [35].

Two additional data members are defined by `MatlabConductivity`, namely, a `MatlabHelper` object, `mh`, which manages a runtime interface with the MatlabEngine [34], and a Matlab matrix, `ct`. These two members are needed by the `MatlabConductivity` subclass, not the parent `Conductivity` class, and are therefore included in only the subclass. The `MatlabConductivity` implementation of the `loadConductivities` function simply creates a Matlab command to load the anisotropic conductivity data source, executes this command within the MatlabEngine via the `mh` object, and then stores the anisotropic conductivity tensor data into the `ct` matrix which are then accessible throughout a TES simulation via the `getConductivity` function (Code 1).

```
class Conductivity
{
  Conductivity();
  virtual ~Conductivity();

  // Load tensor conductivities from data files
  virtual void loadConductivities (String& fileName);

  // Get ith component of conductivity tensor at pt (x,y,z)
  virtual double getConductivity (int i, int x, int y, int z);

class MatlabConductivity: public Conductivity
{
  MatlabConductivity();
  virtual ~MatlabConductivity();

  // Inherited from Conductivity
  virtual void loadConductivities(String& fileName);
  virtual double getConductivity(int i, int x, int y, int z);

  MatlabHelper* mh; // Mangage interface with the MatlabEngine
  mxArray* ct;      // Matrix with conductivity tensor data

void MatlabConductivity::loadConductivities(String& fileName){
  String name = "load('" + fileName + "');";
  engEvalString(mh->ep, name.c_str());
  ct = engGetVariable(mh->ep, "ct");
}
```

Code 1: Class definitions for tissue conductivity data. Class `Conductivity` provides general function names, that is, `loadConductivities` and `getConductivity`. The `MatlabConductivity` subclass implements these functions for a particular data source. For example, its `loadConductivities` function loads a Matlab anisotropic conductivity data source into the `ct` matrix for use in a simulation.

The `Conductivity`-`MatlabConductivity` relationship demonstrates the advantages of object-oriented inheritance and polymorphism. `Conductivity` is used to define function names common to all conductivity data sources and serves as the bridge between these repositories and the framework. Polymorphism permits the `MatlabConductivity` subclass to implement specific `loadConductivities` and `getConductivity` functionality for its particular data format. Code within the `Conductivity` class and all other framework classes does not dependent on these subclass implementations. Thus, alternative conductivity data sources can be easily incorporated into the framework as a subclass of `Conductivity`, in an identical fashion, requiring no modification to any other framework component.

2.3.2. Source Term. Class `Source` is a software abstraction of the $f(\vec{x})$ source term (1a) and like `Conductivity` defines basic functionality to be implemented in subclasses (Code 2). `Source` inherits the Diffpack class `FieldFunc` which allows a scalar function to be defined over the domain. Different subclass implementations of the `valuePt` function can be used to model different neurostimulation modalities. For TES, `valuePt` in class `TES_Source` simply returns zero since $f(\vec{x}) = 0$ in this case (Code 2).

Main TES Class. Class `TES` is the main class in the framework. Code 3 presents the key elements of this class, which contains numerous objects, functions, and Diffpack concepts. Class `TES` inherits the Diffpack `FEM` class, giving it access to finite element data structures and functionality. `Handle` objects, which are pointers in Diffpack that include memory management features, are defined for the computational grid, `grid`, and electric potential and current density solution results, `u` and `currDensity`, respectively.

Several other class members are needed to implement TES simulations. Variables to store boundary condition values are included, as well as the `anodes` and `cathodes` vectors to store electrode information. The vector `isoSigma` and matrix `sigma` store isotropic and anisotropic conductivity data, and the choice of conductivity representation is determined by a user via the `isotropic` boolean variable. The `mc` data member is a reference to the `Conductivity` class, providing an interface to conductivity data. Note that `mc` is type `Conductivity` and not `MatlabConductivity`. The `mc` object can therefore use the `loadConductivities` and `getConductivity` functions of any `Conductivity` subclass, including `MatlabConductivity`. This design approach allows new `Conductivity` subclasses to be defined and utilized without the need to modify code in class `TES`.

```
class Source: public FieldFunc
{
  TES* data;                // Provides access to TES class members

  Source (){}
  virtual dpreal valuePt   // Source term value at pt. x; Abstract function
                (const Ptv(dpreal)& x, dpreal t = DUMMY) = 0;
};

dpreal TES_Source:: valuePt(const Ptv(dpreal)& x, dpreal t) {
  dpreal val = 0.0;
  return val;
}
```

Code 2: Source term class definitions. The `TES_Source` subclass of `Source` enables TES simulations by defining its `valuePt` function to return a value of zero at all points.

```
class TES: public FEM
{
  // DATA TYPES:
  Handle(GridFE)      grid;             // Underlying finite element grid
  Handle(FieldFE)     u;                // Electric potential over grid
  Handle(FieldsFE)    currDensity;      // Electric Current density over grid

  dpreal              dirichlet_val1;   // Constant phi value at a boundary
  dpreal              dirichlet_val2;   // Constant phi value at boundary
  dpreal              robin_U0;         // Constants for Neumann boundary condition

  vector<Electrode>   anodes;           // Anode electrodes
  vector<Electrode>   cathodes;         // Cathode electrodes

  bool                isotropic;        // Isotrpoic or anisotropic bool control
  Vec(dpreal)         isoSigma;         // Isotropic brain tissue conductance values
  MatSimple(dpreal)   sigma;            // Anisotropic conductivity tensor matrix
  Conductivity*       mc;               // Interface class for anisotropic conductivities

  Handle(FieldFunc)   source;           // Source term for initializing f(x)
  Handle(FieldFunc)   alternatingStim;  // Object to model non-direct TES, i.e. tACS

  // FUNCTIONS:
  // Standard finite element functions inherited from class FEM
  virtual void fillEssBC ();            // Set dirichlet boundary conditions
  virtual void calcElmMatVec            // Compute FE coefficient matrix and load vector
                        (int e, ElmMatVec& elmat, FiniteElement& fe);
  virtual void integrands               // Implement weak formulation integrand
                        (ElmMatVec& elmat, const FiniteElement& fe);
  virtual void integrands4side          // Neumann boundary integral
                        (int side, int boind, ElmMatVec& elmat, const FiniteElement& fe);
```

Code 3: Main elements within the `TES` class. Handles are Diffpack pointers that include memory management features. The class definition contains a `Handle` for the computational grid (`grid`) and electric potential (`u`) and current density (`currDensity`) solution results. Next, variables store boundary condition values and anode (`anodes`) and cathode (`cathodes`) electrode information. Support for both isotropic and anisotropic conductivity data is provided, as well as functions for source and nonconstant anode stimulation. Functions inherited from `FEM` are required to perform finite element calculations.

```
void TES:: integrands (ElmMatVec& elmat, const FiniteElement& fe)
{
  int i,j,q;                              // Loop control variables
  const int nbf = fe.getNoBasisFunc(); // no of nodes/basis functions
  dpreal detJxW = fe.detJxW();          // Numerical integration weight
  const int nsd = fe.getNoSpaceDim();  // Number of spatial dimensions

  // Get conductivity tensor for the current element.
  updateConductivityTensors(fe);

  // Get a point in the global domain represented by elemnt fe.
  Ptv(NUMT) x(nsd);
  fe.getGlobalEvalPt(x);

  // Get the source term, f(x), for this element.
  dpreal f_val;
  if (source.ok())
    f_val = source->valuePt(x);
  else f_val = 0;

  // Compute weak formulation
  dpreal gradNi_gradNj;
  for (i = 1; i <= nbf; i++) {
    for (j = 1; j <= nbf; j++) {
      gradNi_gradNj = 0;
      for (q = 1; q <= nsd; q++){
        // Compute inner product of grad(N_i) and grad(N_j).
        gradNi_gradNj += sigma(q,q) * fe.dN(i,q) * fe.dN(j,q);
      }

      // Update linear system coefficient matrix.
      elmat.A(i,j) += gradNi_gradNj*detJxW;
    }
    // Update linear system RHS (load vector).
    elmat.b(i) += fe.N(i)*f_val*detJxW;
  }
}
```

Code 4: TES class integrands function for computing weak formulation volume integrals. Conductivity and source term values for finite element `fe` are retrieved with calls to the `updateConductivityTensors` and source `valuePt` functions. Finite element basis functions are then iterated to compute the volume integrals.

There is also a reference to a `Source` object, for example, `TES_Source`, as well as a field function for nonconstant stimulation currents, for example, tACS. Finally, `TES` inherits functions from `FEM` to perform finite element computations, including `fillEssBC` to set Dirichlet boundary conditions, `calcElmMatVec` to assemble the finite element linear system coefficient matrix and load vector, and the `integrands` and `integrands4side` functions to evaluate the weak formulation volume and boundary integrals, respectively.

2.3.3. Weak Formulation Integration. The weak formulation volume integrals (2) are computed in the `TESintegrands` function (Code 4). The Diffpack linear system assembler automatically calls this function as it iterates over the finite elements in the computational grid. Conductivity values for the current element are attained with a call to the `updateConductivityTensors` function, which determines if isotropic or anisotropic conductivities are desired and then populates the `sigma` matrix accordingly.

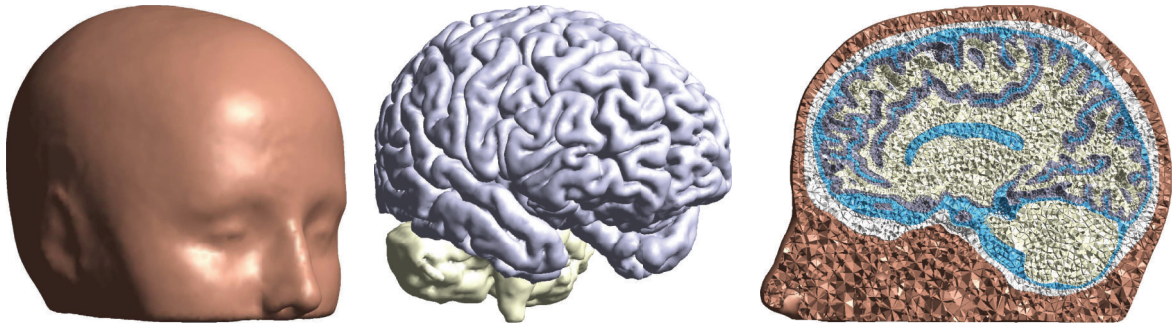
The source term $f(\vec{x})$ over the current element is retrieved via a call to the `valuePt` function in a `Source` class. Then, the finite element basis functions for this element are iterated over, and the weak formulation is computed. Note that, in this implementation, `sigma` is assumed to be diagonal; however, this can be generalized with the incorporation of an additional loop. Finally, the coefficient matrix, `A`, and load vector, `b`, are updated. The boundary integral in the weak formulation is computed similarly in the `integrands4side` function, and its contribution is incorporated into `b`.

2.3.4. Multigrid. Multigrid is a linear system solution algorithm that utilizes multiple computational grid resolutions to

```
class TES_MG: public TES
{
  int                  no_of_grids;
  Handle(MGtools)      mgtools;

  // Set boundary condtion on grid level specified by space
  virtual void mgFillEssBC (SpaceId space);
```

Code 5: New members of `TES_MG` class definition. The `mgtools` object references the Diffpack multigrid toolbox. The `no_of_grids` integer stores the number of MG grid levels, and the `mgFillEssBC` function sets the essential boundary condition on all grid levels.



(a) Head surface (b) Brain region (c) View of sagittal cross section

Figure 2: Computational domain used in numerical simulations.

achieve fast solution convergence [36]. By performing just a few iterations on each grid and then changing between finer and coarser grids, large portions of the error are efficiently removed [37]. In addition, as a preconditioner to the commonly used conjugate gradient (CG) method, MG is highly effective at solving linear systems that result from finite element based TES simulations [38]. Two commonly used MG cycles are the V-cycle and the W-cycle, and identifying the optimal cycle type for a given PDE system, in addition to other MG parameters including the number of grid levels, can be challenging [38].

As a subclass of `TES`, very few new data members and functions are needed in `TES_MG` (Code 5), since the entirety of `TES` is efficiently reused. The new members in `TES_MG` are a reference to the Diffpack multigrid toolbox, `mgtools`, and an integer representing the number of MG grid levels, `no_of_grids`. Finally, just one new function, `mgFillEssBC`, is needed to set the essential boundary condition (1b) on all grid levels.

2.4. Computational Tools. Numerical simulations were performed on a three-dimensional mesh (Figure 2) generated from human MRI images by the SimNIBS software package [35]. The mesh contains the skin, skull, cerebral spinal fluid (CSF), gray matter (GM), and white matter (WM) tissues of the head. Gmsh [39] enabled mesh visualization, identification of electrode coordinates, and grid conversion to a form supported by Diffpack. Electric potential and current density results were exported from Diffpack and visualized with ParaView [40] and gnuplot [41]. Anisotropic conductivity data for the GM and WM regions are provided by SimNIBS in a Matlab binary data file; these data are accessed by the MatlabEngine [34] via the `MatlabHelper` object of the `MatlabConductivity` class (Code 1).

2.5. Computational Simulations. Multiple simulations were performed with the TES framework. Simulations were selected to demonstrate the framework's versatility and ability to target medical, biophysical, and computational research objectives. Finally, to illustrate how alternative forms of neurostimulation can be simulated with the same software, we show a trivial extension to the framework that enables support for DBS applications.

Simulation 1 (comparison of isotropic and anisotropic conductivity data). Previous research suggests that incorporating anisotropic, rather than isotropic, brain tissue conductivity data is important for most accurately modeling TES electrical current distribution [17, 42]. To evaluate the impact that these two conductivity representations have on simulation results, TES simulations were performed with isotropic and anisotropic data. The three-dimensional domain shown in Figure 2 was used with approximately 2.8 million linear tetrahedra finite elements.

Anode and cathode electrodes were positioned at C3 and C4 [43], respectively, each with a surface area of approximately 16 cm^2, and the anode electric current magnitude was set to 1.0 mA. This montage has been used to stimulate the motor cortex ipsilateral to the C3 anode electrode [44, 45]. First, isotropic electrical conductivities were assigned to different tissues: skin = 0.465, skull = 0.010, CSF = 1.654, GM = 0.276, and WM = 0.126, each with units (S/m) [46].

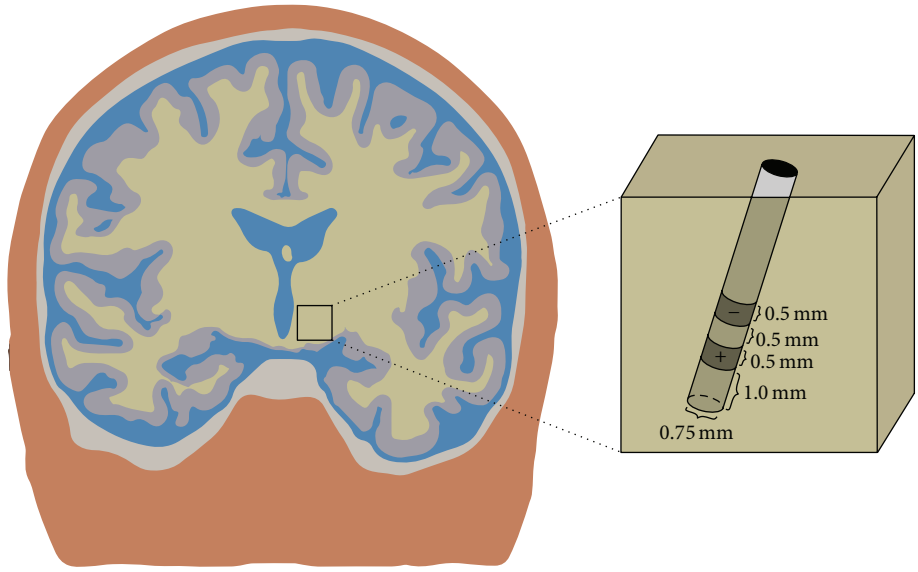


Figure 3: Simulation 4, DBS electrode positioning (subthalamic nucleus) and dimensions. Anode and cathode contacts are denoted with "+" and "−" symbols, respectively.

Then, anisotropic conductivity data were used via the `MatlabConductivity` class as previously described (see Code 1).

Simulation 2 (comparison of tDCS and HD-tDCS electrode montages). High-definition TES electrodes have demonstrated a greater ability to focus its electrical current on a targeted brain region than traditional tDCS, which uses two larger electrodes [2, 3]. Using the same computational grid as Simulation 1, the current densities produced by tDCS and HD-tDCS were compared.

For tDCS, the anode was positioned over C3 and the cathode over the contralateral supraorbital, each with a surface area of approximately 16 cm^2. In comparison, high-definition electrodes, each circular with a 12 mm diameter, were positioned according to a 4×1 configuration; a single anode was positioned over C3, and four cathodes were placed approximately 5 cm radially from the anode, forming a square. Both of these montages are known to target the motor cortex region ipsilateral to the anode electrode [15, 47–50]. Anode stimulation strength for each simulation was once again 1.0 mA, and both utilized anisotropic conductivities.

Simulation 3 (comparison of finite element linear system solvers). Finite element based simulations of TES require the solution of large systems of linear equations, which can become a computational bottleneck for simulations performed on very fine meshes. Therefore, the effectiveness of a TES simulation is directly related to the efficiency of the chosen linear solver. The CG method is ideal for solving these linear systems and appropriate preconditioning can rapidly accelerate numerical results [36, 38].

The numerical efficiency of the preconditioned CG method was evaluated with TES simulations on the three-dimensional volume mesh (Figure 2), with approximately 29 million linear tetrahedra finite elements and roughly 5.1 million unknowns. The anode was positioned at CZ, with a stimulation strength of 1.0 mA, and the cathode at OZ [12, 51]. Simulations were performed with the CG method preconditioned with symmetric successive overrelaxation (SSOR), relaxed incomplete LU decomposition (RILU), and multigrid. The RILU relaxation parameter was set to 0.5 [27]. The MG preconditioner was simulated with a V-cycle with both two and three grid levels, and a W-cycle with three grid levels. The relative residual convergence tolerance was set to 10^{-8} for all numerical experiments.

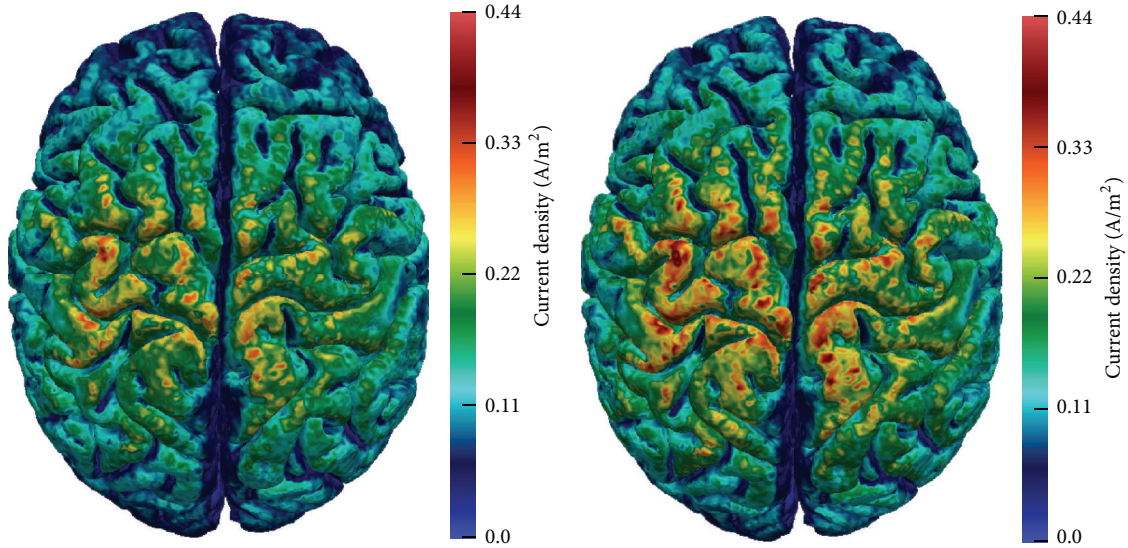
Simulation 4 (impact of conductivity representation on DBS simulation results). The importance of incorporating anisotropic conductivities in TES simulations suggests that accurately simulating other neurostimulation modalities may depend on this conductivity representation as well. In this numerical experiment, we compared DBS simulation results using both isotropic and anisotropic conductivities. A single electrode was positioned in the subthalamic nucleus (STN), the most commonly targeted location for DBS [52]. The electrode is a simplified version of the Medtronic (Model 3387) electrode used in humans [53]. The lower 5.0 mm of the electrode was modeled. The anode is positioned 1.0 mm from the electrode tip and the cathode is separated by 0.5 mm from the anode. Both the anode and cathode are 0.5 mm in height, and the overall electrode diameter was set to 0.75 mm (Figure 3). The anode and cathode metal contacts were modeled as conductors by setting the electrical conductivities of these regions to 10^6 (S/m), and the remaining electrode shaft was modeled as an insulator with conductivity equal to 10^{-6} (S/m) [54]. Because DBS electric current is proximal to the electrode [29, 55], a 10.0 mm × 4.0 mm × 4.0 mm subset of tissue around the electrode was considered as the computational domain.

```
class DBS_Source: public Source
{
  DBS_Source(TES* data);
  virtual dpreal valuePt(const Ptv(dpreal)& x, dpreal t = DUMMY);
};

dpreal DBS_Source:: valuePt(const Ptv(dpreal)& x, dpreal t) {
  dpreal val = 0.0;

  if (4.625 <= x(1) && x(1) <= 5.375 &&
      4.625 <= x(2) && x(2) <= 5.375 &&
      3.500 <= x(3) && x(3) <= 4.000 ) {
    val = 0.001;
  }
```

Code 6: `DBS_Source` class definition and sample code from its `valuePt` function.



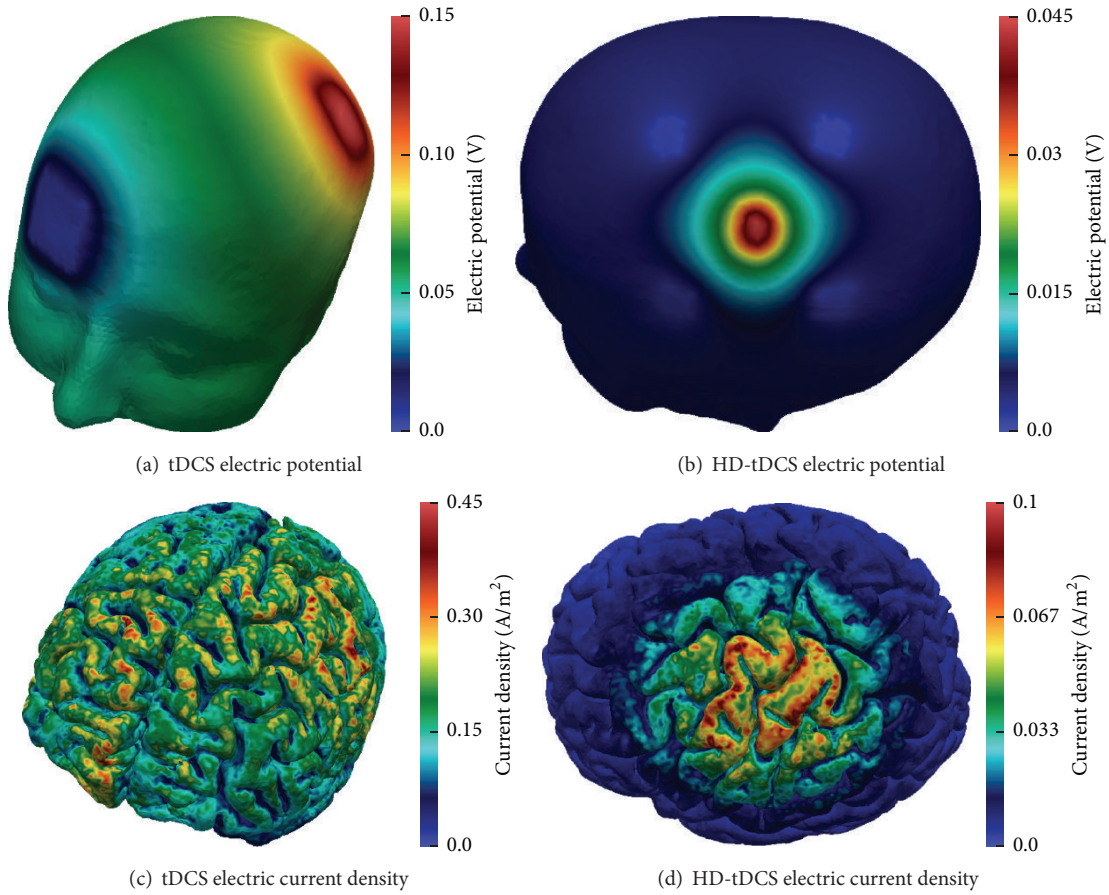
(a) Isotropic conductivities (b) Anisotropic conductivities

Figure 4: Simulation 1, current density results viewed from above with the nasion facing up. Anode was placed at C3 and cathode at C4.

To simulate DBS with the existing TES software framework, no existing code required modification. Rather, the only addition was a new subclass of class `Source` that defines $f(\vec{x})$ (1a) for DBS; in total, less than ten lines of code were added. Code 6 displays this new subclass definition, `Source_DBS`, and highlights of its `valuePt` function implementation, which in this simplified scenario simply injects 1.0 mA of current from the anode contact into the surrounding tissue. In practice, DBS stimulations are time-varying pulses, and differing stimulation frequencies impact GM and WM tissue conductivities [56]. However, when examining electric field dispersion around a DBS electrode as in this simulation, a constant-valued source term in (1a) can be utilized [57].

3. Results and Discussion

3.1. Simulation 1. Electric current density results on the surface of the brain tissue are displayed in Figure 4. Viewing perspective is from above the head with the nasion facing up. As expected, the largest electric current density values are attained near the anode and cathode locations. Despite an overall similar pattern of electric current distribution, the simulated current densities, particularly in the motor cortex ipsilateral to the anode electrode, are significantly different between the isotropic and anisotropic simulations. For example, the maximal anisotropic current density value (0.434 (A/m^2)) is approximately 18.2% higher than in the isotropic case (0.367 (A/m^2)). Similar discrepancies are observed throughout the top surface of the brain, including a substantial impact on the paracentral lobule contralateral to the anode. In addition, these discrepancies extend to the GM and WM interiors. These results reinforce the importance of using anisotropic conductivities to most accurately model TES administrations and in addition showcase the capabilities of the object-oriented framework to support both isotropic and anisotropic TES simulations.



(a) tDCS electric potential

(b) HD-tDCS electric potential

(c) tDCS electric current density

(d) HD-tDCS electric current density

Figure 5: Simulation 2, electric potential and current density results. The tDCS montage positioned the anode at C3 and the cathode over the contralateral supraorbital. A 4×1 configuration was used for HD-tDCS with a single anode positioned over C3.

3.2. Simulation 2. Electric potential and electric current density results are displayed in Figure 5. The tDCS montage results in a noticeably larger range of electric potential values (Figures 5(a) and 5(b)). However, the focus of the tDCS current density on the targeted region, in this case the motor cortex under C3, is much less than the HD-tDCS electrode montage (Figures 5(c) and 5(d)). Specifically, the brain tissue outside of the square formed by the HD-tDCS cathodes is virtually unstimulated, whereas tDCS results in a much greater electric current dispersion.

One limitation of this HD-tDCS electrode arrangement is the lower concentration of electric current that reaches the brain tissue. Specifically, the maximal electric current density in the brain from HD-tDCS is just approximately 23% of that achieved with traditional tDCS. The culprit for this effect is the shunting of the electric current around the poorly conducting skull; the HD-tDCS montage used in this simulation yields a minuscule amount of current that penetrates the skull to reach the CSF and brain tissues (Figure 6). Potential remedies for this behavior are a greater anode stimulation or positioning the cathodes at greater distances from the anode [3].

This simulation demonstrates the framework's ability to support varying TES electrode montages, including high-definition electrode configurations. Results of this simulation

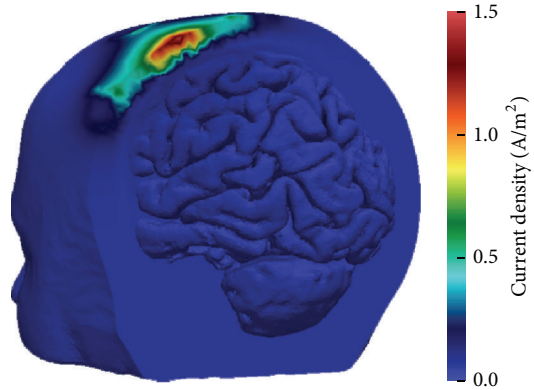
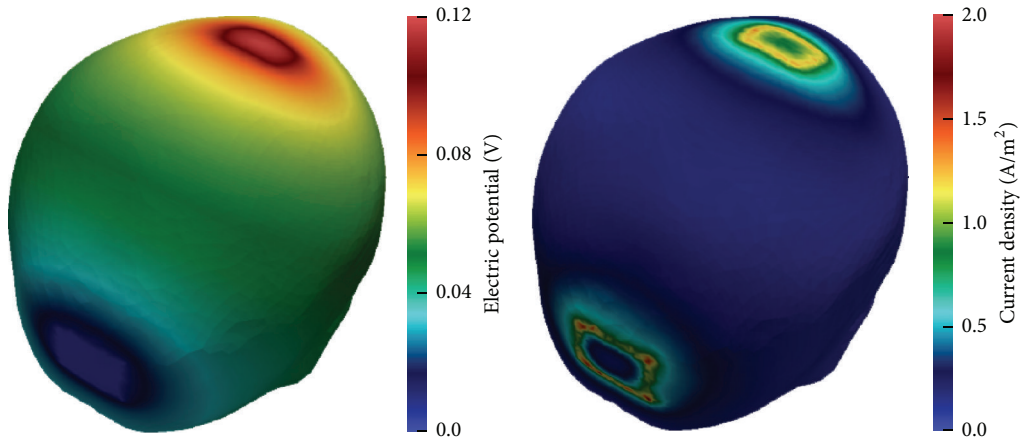


Figure 6: Simulation 2, HD-tDCS electric current density. Cross section is through two diagonally positioned cathodes.

corroborate that HD-tDCS offers greater electric current focality; however, its net stimulation of brain tissue is potentially much lower than tDCS.

3.3. Simulation 3. Electric potential and current density results on the head surface are displayed in Figure 7. Viewing



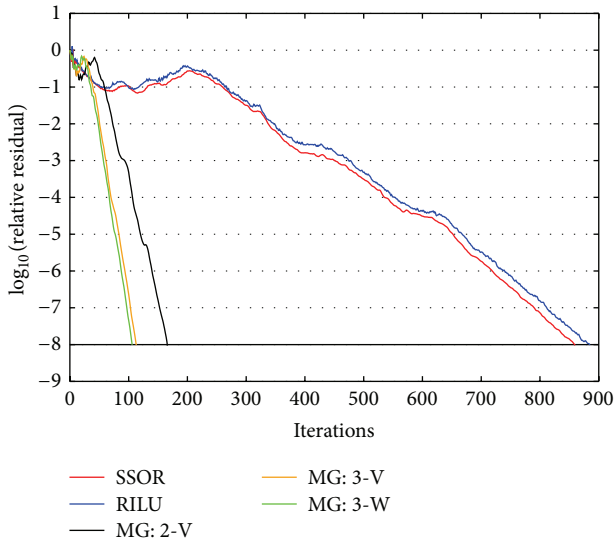
(a) Electric potential (b) Current density

FIGURE 7: Simulation 3, electric potential and current density results viewed from the back of the head. The anode was positioned at CZ and the cathode at OZ.

perspective is from behind the head, and as anticipated, maximal and minimal electric potential and current density values occur at the anode and cathode, respectively. The shunting of the electric current around the skull, due to its low electrical conductivity, results in the segregated current density pattern around the anode and cathode centers (Figure 7(b)). This phenomenon has been previously observed [1]; however, it is highlighted by the very fine mesh resolution used in this simulation.

Figure 8 displays convergence history curves and performance metrics for each CG preconditioning strategy. With no preconditioning, the CG method will eventually convergence but will accomplish this extremely slowly, requiring more than 6 hours to solve the linear system. The SSOR and RILU preconditioners demonstrate comparable performances, both in solution time and number of iterations. Multigrid preconditioning with two grid levels has far fewer iterations with 166 than both SSOR and RILU and yet has a greater run time. This observation can be explained by the fact that a single iteration of MG has embedded iterations on the different mesh refinements [37]. However, three-grid-level MG noticeably accelerates convergence rates, with both the V- and W-cycles outperforming the other preconditioners. Three-grid-level MG with a W-cycle pattern has slightly fewer iterations than its corresponding V-cycle, yet this V-cycle preconditioner is approximately 11.3% faster.

The ability of the framework to support numerically oriented TES research is presented in this simulation example. The results indicate that the CG method combined with an appropriately configured MG preconditioner is highly efficient in solving the linear systems produced in TES computational simulations.



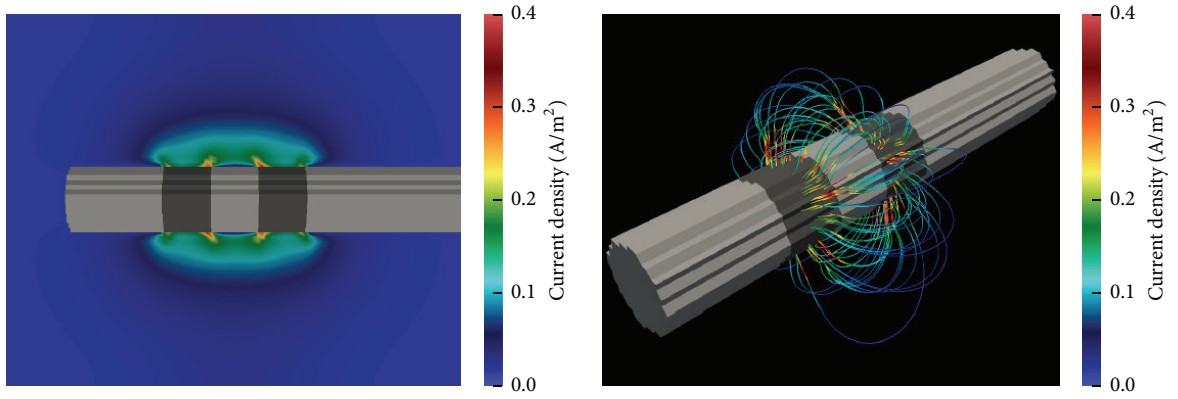
(a) Convergence history

Preconditioner	Iterations	Time (min)
None	>5000	>360
SSOR	860	113.4
RILU	885	117.5
MG 2-grid V-cycle	166	126.4
MG 3-grid V-cycle	113	**57.4**
MG 3-grid W-cycle	**106**	64.7

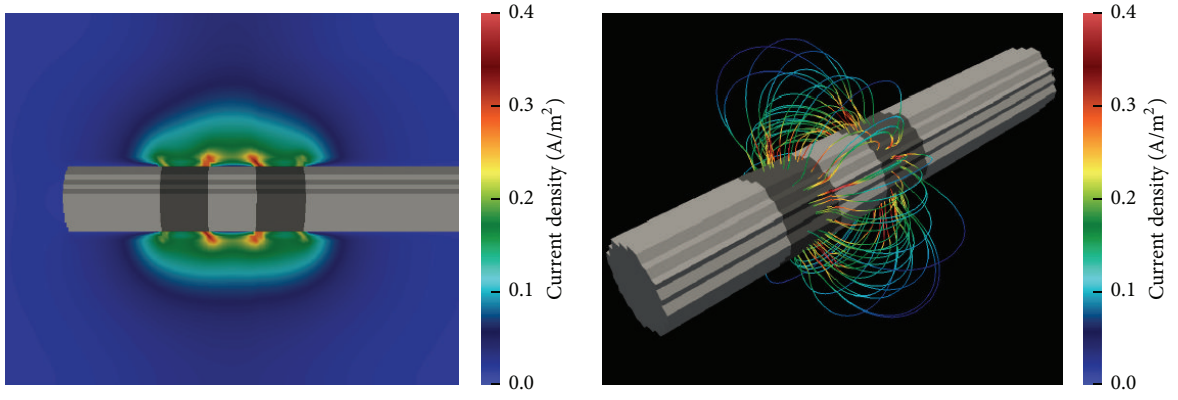
(b) Numerical iterations and linear system solve time. Boldface values indicate best convergence results

FIGURE 8: Convergence performances of the preconditioned conjugate gradient methods.

3.4. Simulation 4. Figure 9 displays the electric current densities produced by the DBS electrode using isotropic (Figure 9(a)) and anisotropic (Figure 9(b)) conductivities. In both cases, the majority of the current density is proximal to the electrode, indicating that accurate placement of electrodes in DBS procedures is paramount [56]. While the maximal current densities of the isotropic and anisotropic scenarios are similar, other differences between them are observable. First, the current density around the electrode is nonsymmetric in the anisotropic case, as is expected with directionally dependent conductivity data. In addition,



(a) Isotropic



(b) Anisotropic

FIGURE 9: Simulation 4, electric current density magnitudes and field lines. Current densities in the figures on the left are from a coronal cross section through the electrode center.

anisotropic conductivities result in a greater electrical current intensity adjacent to the electrode and more dispersion into the neighbouring tissue. This is also observed in the electrical field lines, where anisotropic conductivities produce a more intense and further-reaching current.

With a straightforward extension, DBS was simulated with the object-oriented TES framework. The entirety of the framework is reused in the DBS simulation, which demonstrates how object-oriented design can produce efficient and scalable software implementations. This particular example further reinforces the importance that anisotropic conductivities play in accurately modeling neurostimulation.

While this DBS scenario utilizes a constant source term to assess electrode electric field dispersion [57], alternative applications may demand the use of time-dependent pulses, such as those that examine temporal neuronal responses to nonconstant electric current administrations. In these cases, different stimulation frequencies have an impact on the electrical impedance; specifically, WM and GM conductivities are frequency dependent where an increase in stimulation frequency yields an increase in electrical conductivity.

Applications such as these are supported by the software framework. First, since isotropic tissue conductivity values are specified within the configuration input file, and not hard-coded in the software, alternative DBS pulses can be accurately simulated by modifying these values. For anisotropic data, modifications to support frequency-dependent conductivities can be made within the `Conductivity` classes. For example, the SimNIBS software package volume normalizes the WM and GM anisotropic conductivities such that the mean conductivity value of each tensor is maintained to the associated tissue's isotropic value [35]. Therefore, for different DBS frequencies, tensors could be updated to be normalized to different isotropic values within the framework's `Conductivity` components. To implement the time-varying source itself, the existing definition of the `valuePt` function in class `Source` contains an argument for time, namely, `dpreal t`. Thus, implementations of `valuePt` in the `Source_DBS` class can be based on time, and system (1a)–(1d) can then be iteratively solved for with time-based source values.

4. Conclusions

Computational simulations of neurostimulation are a valuable tool that enable researchers to investigate this form of brain therapy *in silico*, and as simulations become more refined, their utility to medical and biomedical research grows. While prebuilt simulation software programs can simplify and expedite TES model implementation, they possess

application and portability limitations. In addition, recreating custom software as applications and research objectives change is inefficient, error-prone, and tedious to maintain.

Since the mathematics and physiology that govern all forms of neurostimulation are the same, a single, well-designed software code can support a versatile range of neurostimulation simulations. In this paper, we have presented one such software framework, described its design and implementation, and demonstrated its abilities to support diverse neurostimulation research objectives. The cornerstone of the design stage was to encapsulate general neurostimulation concepts into modular software objects. In doing so, a multitude of TES simulation areas are supported, and in addition, alternative forms of neurostimulation, for example, DBS, can be simulated with the same software.

Simulation results show the importance of using anisotropic conductivities in both TES and DBS. This is especially important for simulations utilized in selecting patient-specific neurostimulation parameters. In addition, results demonstrate the ability of HD-tDCS to focus the electrical current on a specific brain region; however, this capability must be balanced with a potentially low concentration of electric current actually reaching the target. The object-oriented TES framework is an ideal tool for this scenario, enabling different permutations of HD-tDCS electrode montages and stimulation strengths to be conveniently investigated to identify an optimal configuration for a particular patient's data and therapeutic objectives.

Results also illustrate that appropriately configured multigrid preconditioning can achieve superior convergence rates. It is conceivable that hundreds of simulations could be run to identify an optimal electrode montage and neurostimulation parameters for a particular patient. Hence, efficiently solving the linear system of equations resulting from TES finite element simulations is crucial, and MG can be used in this capacity to greatly decrease simulation run times. Finally, we demonstrated how a minor addition to the object-oriented TES framework enables simulations of DBS. The DBS simulation example that was performed utilizes a constant stimulation source term; however, the software framework description and simulation ensemble presented in this paper motivate how the framework could be used to address DBS research related to electrode placement and parameter values.

In future work, we plan to extend the framework to support anisotropic transcranial magnetic stimulation (TMS), in a similar fashion as was demonstrated for DBS. In addition, we plan to utilize the framework to more thoroughly investigate correlations between HD-tDCS interelectrode distances and electric current distributions.

Appendix

Weak Formulation

We multiply (1a) by a test function $v = v(\vec{x})$ and integrate over Ω to obtain

$$-\int_{\Omega} v\left(\nabla \cdot (\mathbf{M}\nabla\Phi)\right) dx = \int_{\Omega} vf\, dx, \tag{A.1}$$

and using Green's theorem we have

$$\int_{\Omega} \nabla v \cdot \mathbf{M}\nabla\Phi\, dx = \int_{\partial\Omega} v\left(\mathbf{M}\nabla\Phi \cdot \vec{n}\right) ds + \int_{\Omega} vf\, dx. \tag{A.2}$$

Expanding the surface integral gives

$$\begin{aligned}\int_{\Omega} \nabla v \cdot \mathbf{M}\nabla\Phi\, dx = &\int_{\partial\Omega_C} v\left(\mathbf{M}\nabla\Phi \cdot \vec{n}\right) ds \\ &+ \int_{\partial\Omega_A} v\left(\mathbf{M}\nabla\Phi \cdot \vec{n}\right) ds \\ &+ \int_{\partial\Omega_S} v\left(\mathbf{M}\nabla\Phi \cdot \vec{n}\right) ds \\ &+ \int_{\Omega} vf\, dx,\end{aligned} \tag{A.3}$$

and substituting the boundary conditions (1c) and (1d) yields

$$\begin{aligned}\int_{\Omega} \nabla v \cdot \mathbf{M}\nabla\Phi\, dx = &\int_{\partial\Omega_C} v\left(\mathbf{M}\nabla\Phi \cdot \vec{n}\right) ds \\ &+ \int_{\partial\Omega_A} vI\, ds + \int_{\Omega} vf\, dx.\end{aligned} \tag{A.4}$$

For these integrals to exist, we require $v, \Phi \in H^1(\Omega)$ and $f \in L_2(\Omega)$, where

$$\begin{aligned}H^1(\Omega) &= \left\{u \mid u \in L_2(\Omega), \frac{\partial u}{\partial x_i} \in L_2(\Omega)\right\}, \\ L_2(\Omega) &= \left\{p \mid \int_{\Omega} |p|^2\, dx < \infty\right\},\end{aligned} \tag{A.5}$$

and we enforce the Dirichlet boundary condition (1b) on our solution space by further stipulating that $v, \Phi \in H_0^1 = \{u \mid u \in H^1(\Omega), u = 0 \text{ for } \vec{x} \in \partial\Omega_C\}$.

Consequently, we have the following weak formulation.

Given $f(\vec{x}) \in L_2(\Omega)$, find $\Phi(\vec{x}) \in H_0^1(\Omega)$ such that

$$\begin{aligned}\int_{\Omega} \nabla v \cdot \mathbf{M}\nabla\Phi\, dx = \int_{\partial\Omega_A} vI\, ds + \int_{\Omega} fv\, dx, \\ \forall v(\vec{x}) \in H_0^1(\Omega).\end{aligned} \tag{A.6}$$

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to acknowledge Frank Vogel and the entire inuTech team for their assistance with Diffpack. The authors would also like to acknowledge the financial support received from Virginia Tech's Open Access Subvention Fund.

References

[1] A. Datta, X. Zhou, Y. Su, L. C. Parra, and M. Bikson, "Validation of finite element model of transcranial electrical stimulation using scalp potentials: implications for clinical dose," *Journal of Neural Engineering*, vol. 10, no. 3, Article ID 036018, 2013.

[2] A. Datta, V. Bansal, J. Diaz, J. Patel, D. Reato, and M. Bikson, "Gyri-precise head model of transcranial direct current stimulation: improved spatial focality using a ring electrode versus conventional rectangular pad," *Brain Stimulation*, vol. 2, no. 4, pp. 201–207, 2009.

[3] P. Faria, M. Hallett, and P. C. Miranda, "A finite element analysis of the effect of electrode area and inter-electrode distance on the spatial distribution of the current density in tDCS," *Journal of Neural Engineering*, vol. 8, no. 6, Article ID 066017, 2011.

[4] P. S. Boggio, R. Ferrucci, S. P. Rigonatti et al., "Effects of transcranial direct current stimulation on working memory in patients with Parkinson's disease," *Journal of the Neurological Sciences*, vol. 249, no. 1, pp. 31–38, 2006.

[5] P. S. Boggio, L. P. Khoury, D. C. S. Martins, O. E. M. S. Martins, E. C. de Macedo, and F. Fregni, "Temporal cortex direct current stimulation enhances performance on a visual recognition memory task in Alzheimer disease," *Journal of Neurology, Neurosurgery and Psychiatry*, vol. 80, no. 4, pp. 444–447, 2009.

[6] P. S. Boggio, C. A. Valasek, C. Campanhã et al., "Non-invasive brain stimulation to assess and modulate neuroplasticity in Alzheimer's disease," *Neuropsychological Rehabilitation*, vol. 21, no. 5, pp. 703–716, 2011.

[7] R. Ferrucci, M. Bortolomasi, M. Vergari et al., "Transcranial direct current stimulation in severe, drug-resistant major depression," *Journal of Affective Disorders*, vol. 118, no. 1–3, pp. 215–219, 2009.

[8] P. S. Boggio, S. P. Rigonatti, R. B. Ribeiro et al., "A randomized, double-blind clinical trial on the efficacy of cortical direct current stimulation for the treatment of major depression," *International Journal of Neuropsychopharmacology*, vol. 11, no. 2, pp. 249–254, 2008.

[9] P. Homan, J. Kindler, A. Federspiel et al., "Muting the voice: a case of arterial spin labeling-monitored transcranial direct current stimulation treatment of auditory verbal hallucinations," *The American Journal of Psychiatry*, vol. 168, no. 8, pp. 853–854, 2011.

[10] J. Brunelin, M. Mondino, F. Haesebaert, M. Saoud, M. F. Suaud-Chagny, and E. Poulet, "Efficacy and safety of bifocal tDCS as an interventional treatment for refractory schizophrenia," *Brain Stimulation*, vol. 5, no. 3, pp. 431–432, 2012.

[11] A. Antal and W. Paulus, "Transcranial alternating current stimulation (tACS)," *Frontiers in Human Neuroscience*, vol. 7, article 317, 2013.

[12] T. Neuling, S. Wagner, C. H. Wolters, T. Zaehle, and C. S. Herrmann, "Finite-element model predicts current density distribution for clinical applications of tDCS and tACS," *Frontiers in Psychiatry*, vol. 3, article 83, 2012.

[13] F. Gasca, L. Marshall, S. Binder, A. Schlaefer, U. G. Hofmann, and A. Schweikard, "Finite element simulation of transcranial current stimulation in realistic rat head model," in *Proceedings of the 5th International IEEE/EMBS Conference on Neural Engineering (NER 2011)*, pp. 36–39, IEEE, Cancun, Mexico, May 2011.

[14] P. C. Miranda, M. Lomarev, and M. Hallett, "Modeling the current distribution during transcranial direct current stimulation," *Clinical Neurophysiology*, vol. 117, no. 7, pp. 1623–1629, 2006.

[15] E. M. Caparelli-Daquer, T. J. Zimmermann, E. Mooshagian et al., "A pilot study on effects of 4×1 high-definition tDCS on motor cortex excitability," in *Proceedings of the IEEE Annual International Conference of the Engineering in Medicine and Biology Society (EMBC '12)*, vol. 735, pp. 735–738, San Diego, Calif, USA, September 2012.

[16] S. K. Kessler, P. Minhas, A. J. Woods, A. Rosen, C. Gorman, and M. Bikson, "Dosage considerations for transcranial direct current stimulation in children: a computational modeling study," *PLoS ONE*, vol. 8, no. 9, Article ID e76112, 2013.

[17] H. S. Suh, W. H. Lee, and T.-S. Kim, "Influence of anisotropic conductivity in the skull and white matter on transcranial direct current stimulation via an anatomically realistic finite element head model," *Physics in Medicine and Biology*, vol. 57, no. 21, pp. 6961–6980, 2012.

[18] S. Wagner, S. M. Rampersad, Ü. Aydin et al., "Investigation of tDCS volume conduction effects in a highly realistic head model," *Journal of Neural Engineering*, vol. 11, no. 1, Article ID 016002, 2014.

[19] S. Lew, C. H. Wolters, T. Dierkes, C. Röer, and R. S. MacLeod, "Accuracy and run-time comparison for different potential approaches and iterative solvers in finite element method based EEG source analysis," *Applied Numerical Mathematics*, vol. 59, no. 8, pp. 1970–1988, 2009.

[20] W. Aquino, "An object-oriented framework for reduced-order models using proper orthogonal decomposition (POD)," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 41–44, pp. 4375–4390, 2007.

[21] R. I. Mackie, "An object-oriented approach to fully interactive finite element software," *Advances in Engineering Software*, vol. 29, no. 2, pp. 139–149, 1998.

[22] R. Sampath and N. Zabaras, "An object-oriented framework for the implementation of adjoint techniques in the design and control of complex continuum systems," *International Journal for Numerical Methods in Engineering*, vol. 48, no. 2, pp. 239–266, 2000.

[23] M. Hakman and T. Groth, "Object-oriented biomedical system modeling—the rationale," *Computer Methods and Programs in Biomedicine*, vol. 59, no. 1, pp. 1–17, 1999.

[24] S. C. Lee, K. Bhalerao, and M. Ferrari, "Object-oriented design tools for supramolecular devices and biomedical nanotechnology," *Annals of the New York Academy of Sciences*, vol. 1013, pp. 110–123, 2004.

[25] S. Tuchschmid, M. Grassi, D. Bachofen et al., "A flexible framework for highly-modular surgical simulation systems," in *Biomedical Simulation*, vol. 4072 of *Lecture Notes in Computer Science*, pp. 84–92, Springer, Berlin, Germany, 2006.

[26] A. Doronin and I. Meglinski, "Online object oriented Monte Carlo computational tool for the needs of biomedical optics," *Biomedical Optics Express*, vol. 2, no. 9, pp. 2461–2469, 2011.

[27] H. P. Langtangen, *Computational Partial Differential Equations: Numerical Methods and Diffpack Programming*, Texts in Computational Science and Engineering, Springer, Berlin, Germany, 2003.

[28] H. P. Langtangen and A. Tveito, *Advanced Topics in Computational Partial Differential Equations: Numerical Methods and Diffpack Programming*, Lecture Notes in Computational Science and Engineering, Springer, Berlin, Germany, 2003.

[29] M. Åström, L. U. Zrinzo, S. Tisch, E. Tripoliti, M. I. Hariz, and K. Wårdell, "Method for patient-specific finite element modeling and simulation of deep brain stimulation," *Medical and Biological Engineering and Computing*, vol. 47, no. 1, pp. 21–28, 2009.

[30] T. Budd, *An Introduction to Object-Oriented Programming*, Addison-Wesley, Boston, Mass, USA, 2002.

[31] A. M. Bruaset and H. P. Langtangen, "Diffpack: a software environment for rapid prototyping of PDE solvers," in *Proceedings of the 15th IMACS World Congress on Scientific Computation, Modeling and Applied Mathematics*, pp. 553–558, Berlin, Germany, August 1997.

[32] B. Stroustrup, *The C++ Programming Language*, Addison-Wesley, Upper Saddle River, NJ, USA, 2013.

[33] S. Prata, *C++ Primer Plus*, Addison-Wesley, Upper Saddle River, NJ, USA, 2012.

[34] *MATLAB Version 8.2.0.701 (R2013b)*, MathWorks, Natick, Mass, USA, 2013.

[35] M. Windhoff, A. Opitz, and A. Thielscher, "Electric field calculations in brain stimulation based on finite elements: an optimized processing pipeline for the generation and usage of accurate individual head models," *Human Brain Mapping*, vol. 34, no. 4, pp. 923–935, 2013.

[36] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2003.

[37] W. L. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, Pa, USA, 2000.

[38] K. A. Mardal, G. W. Zumbusch, and H. P. Langtangen, "Software tools for multigrid methods," in *Advanced Topics in Computational Partial Differential Equations: Numerical Methods and Diffpack Programming*, H. P. Langtangen and A. Tveito, Eds., Lecture Notes in Computational Science and Engineering, pp. 97–152, Springer, Berlin, Germany, 2003.

[39] C. Geuzaine and J.-F. Remacle, "Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009.

[40] A. Henderson, J. Ahrens, and C. Law, *The Paraview Guide*, Kitware Inc, Clifton Park, NY, USA, 2004.

[41] T. Williams, C. Kelley et al., "Gnuplot 4.4: an interactive plotting program," March 2011.

[42] A. Opitz, M. Windhoff, R. M. Heidemann, R. Turner, and A. Thielscher, "How the brain tissue shapes the electric field induced by transcranial magnetic stimulation," *NeuroImage*, vol. 58, no. 3, pp. 849–859, 2011.

[43] M. A. Nitsche, L. G. Cohen, E. M. Wassermann et al., "Transcranial direct current stimulation: state of the art 2008," *Brain Stimulation*, vol. 1, no. 3, pp. 206–223, 2008.

[44] M. Okamoto, H. Dan, K. Sakamoto et al., "Three-dimensional probabilistic anatomical cranio-cerebral correlation via the international 10-20 system oriented for transcranial functional brain mapping," *NeuroImage*, vol. 21, no. 1, pp. 99–111, 2004.

[45] E. K. Kang and N.-J. Paik, "Effect of a tDCS electrode montage on implicit motor sequence learning in healthy subjects," *Experimental and Translational Stroke Medicine*, vol. 3, no. 1, article 4, 2011.

[46] A. Datta, J. M. Baker, M. Bikson, and J. Fridriksson, "Individualized model predicts brain current flow during transcranial direct-current stimulation treatment in responsive stroke patient," *Brain Stimulation*, vol. 4, no. 3, pp. 169–174, 2011.

[47] G. Schlaug, V. Renga, and D. Nair, "Transcranial direct current stimulation in stroke recovery," *Archives of Neurology*, vol. 65, no. 12, pp. 1571–1576, 2008.

[48] A. F. DaSilva, M. S. Volz, M. Bikson, and F. Fregni, "Electrode positioning and montage in transcranial direct current stimulation," *Journal of Visualized Experiments*, no. 51, 2011.

[49] A. Antal, D. Terney, C. Poreisz, and W. Paulus, "Towards unravelling task-related modulations of neuroplastic changes induced in the human motor cortex," *European Journal of Neuroscience*, vol. 26, no. 9, pp. 2687–2691, 2007.

[50] D. Q. Truong, G. Magerowski, G. L. Blackburn, M. Bikson, and M. Alonso-Alonso, "Computational modeling of transcranial direct current stimulation (tDCS) in obesity: impact of head fat and dose guidelines," *NeuroImage: Clinical*, vol. 2, no. 1, pp. 759–766, 2013.

[51] A. Antal, K. Boros, C. Poreisz, L. Chaieb, D. Terney, and W. Paulus, "Comparatively weak after-effects of transcranial alternating current stimulation (tACS) on cortical excitability in humans," *Brain Stimulation*, vol. 1, no. 2, pp. 97–105, 2008.

[52] S. Miocinovic, S. Somayajula, S. Chitnis, and J. L. Vitek, "History, applications, and mechanisms of deep brain stimulation," *JAMA Neurology*, vol. 70, no. 2, pp. 163–171, 2013.

[53] L. M. Zitella, K. Mohsenian, M. Pahwa, C. Gloeckner, and M. D. Johnson, "Computational modeling of pedunculopontine nucleus deep brain stimulation," *Journal of Neural Engineering*, vol. 10, no. 4, Article ID 045005, 2013.

[54] S. Miocinovic, M. Parent, C. R. Butson et al., "Computational analysis of subthalamic nucleus and lenticular fasciculus activation during therapeutic deep brain stimulation," *Journal of Neurophysiology*, vol. 96, no. 3, pp. 1569–1580, 2006.

[55] C. C. Mcintyre and T. J. Foutz, "Computational modeling of deep brain stimulation," *Handbook of Clinical Neurology*, vol. 116, pp. 55–61, 2013.

[56] D. Tarsy, *Deep Brain Stimulation In Neurological And Psychiatric Disorders*, Humana Press, Totowa, NJ, USA, 2008.

[57] M. Astrom, *Modelling, simulation, and visualization of deep brain stimulation [Ph.D. thesis]*, Linköping University, Linköping, Sweden, 2011.