A Software Framework for Solving Problems of Bioelectricity Applying High-Order Finite Elements

M. Cole[1], F. B. Sachse[2], D. M. Weinstein[1], M. Kirby[1], S. Parker[1]
[1]Scientific Computing and Imaging Institute
[2]Nora Eccles Harrison Cardiovascular Research and Training Institute,
University of Utah, UT, USA

Abstract— **Electrical activity in biological media can be described in a mathematical way, which is applicable to computer-based simulation. Biophysical based mathematical descriptions provide important insights in the electrical and electrophysiological properties of cells, tissues, and organs. Examples of these descriptions are Maxwell's and Poisson's equations for electromagnetic and electric fields. Commonly, numerical techniques are applied for calculation of electrical fields, e.g. the finite element method. Finite elements can be classified on the order of the underlying interpolation. High-order finite elements provide enhanced geometrical flexibility and can increase accuracy of a solution. Aim of this work is the design of a framework for describing and solving high-order finite elements in the software package SCIRun/BioPSE, which allows geometric modeling, simulation, and visualization for solving bioelectric field problems. Currently, only low-order elements are supported. Our design for high-order elements concerns interpolation of geometry and physical fields. The design is illustrated by an exemplary implementation of one-dimensional elements with cubic interpolation of geometry and field variables.**

Keywords— **Finite element method, bioelectricity, SCIRun, Poisson equation, numerical techniques, high-order elements**

I. Introduction

Neurons and myocytes show a prominent electrical activity, which is closely related to their function. Measurement of the electrical activity is the basis of several diagnostic techniques, e.g. electrocardiography (ECG), electromyography (EMG), and electroencephalography (EEG). These diagnostic techniques are not only used in clinical routine to obtain information concerning the state of a patient, but also serve as working horses in medical research and development.

The electrical activity in biological media can be described in a mathematical way, e.g. with Maxwell's and Poisson's equations for electromagnetic and electric fields, respectively, as well as the Hodgkin-Huxley equations for current flow through a neuron membrane [1], [2]. Different numerical techniques can be applied to solve these equations.

Commonly, finite element, finite difference and boundary element methods are employed for numerical solution of the partial differential equations, which describe electrical fields in biological media. Each of these methods includes some kind of interpolation, wherefore different levels of approximation are applicable. In the finite element methods the interpolation concerns primarily the electrical potential inside of an element and is carried out by so-called shape-functions [3], [4].

A large family of shape-functions is based on polynomial functions for interpolation of the solution function. Commonly, polynomials of order 1 to 3 are found for electrical field problems and the polynomial order corresponds to the number of node points describing the finite element. Higher order elements provide an enhanced freedom to describe the solution function and thus will generally improve the accuracy of a solution, but increase organizational complexity and computational demands of the solution process.

In this work a software framework for applying high-order finite elements to numerically calculated electrical fields in biological media is described. The software framework is developed on basis of SCIRun/BioPSE, which allows geometric modeling, simulation, and visualization for solving bioelectric field problems [5], [6]. Currently, SCIRun/BioPSE includes already low-order finite elements, i.e. of first order, which were available in tetra- and hexahedral meshes.

II. Methodology

A. Stationary Electrical Current Fields

Stationary electrical current fields can be described with Poisson's equation, which is a simplification of the more general Maxwell's equations for electro-magnetic fields. The equation quantifies the flow of current in materials owning electric conductivity. Further material properties are ignored, e.g. electric permittivity and magnetic permeability. A detailed description of the applicability of Poisson's in biological materials is given in [7].

Poisson's equation for stationary current fields is a partial differential equation:

$$\nabla \cdot (\sigma \nabla \phi) = f \text{ in } \Omega \qquad (1)$$

with the scalar potential ϕ, the conductivity tensor σ, the scalar current source density f, and the spatial domain Ω. Commonly, the conductivity tensor σ is of zeroth or second order. Commonly in applications

This work was supported by a grant from the DARPA, executed by the U.S. Army Medical Research and Materiel Command/TATRC Cooperative Agreement, Contract # W81XWH-04-2-0012. Frank B. Sachse thanks for the support by the Richard A. and Nora Eccles Fund for Cardiovascular Research and the Nora Eccles Treadwell Foundation.

of the equation, boundary conditions are added, e.g. Dirichlet and Neumann boundary conditions.

Equation (1) can be transformed to an equivalent integral representation [4], [8], describing the electrical power Π in a unit domain Ω_0:

$$\Pi = \int_{\Omega_0} \left(\frac{1}{2} (\nabla_0 \phi)^T \sigma (\nabla_0 \phi) + f\phi \right) \mathcal{J} \, d\Omega_0 \qquad (2)$$

with the Jacobian of the coordinate transformation $\mathcal{J}$.

B. Finite Element Methods

The ideas underlying the finite element method were introduced with the classical work of Ritz and Galerkin. The first step of application of finite element methods to solve Poisson's equations for electrical current fields is a subdivision of the spatial domain Ω into finite elements. Commonly, in these elements the solution function, i.e. electrical potential ϕ, is interpolated by basis or shape-functions $\boldsymbol{H}$, which are selected dependent on the element's geometry as well as the order and type of interpolation. In the domain $\Omega^{(m)}$ of the m-th element the solution function ϕ is given by:

$$\phi = \boldsymbol{H}^{(m)\,T} \; \boldsymbol{\phi}^{(m)} \qquad (3)$$

with the N-dimensional vector of node variables $\boldsymbol{\phi}^{(m)}$. Commonly, these node variables describe the solution function, i.e. electrical potential, at node points. This assumption leads to requirements for the shape-functions $H_i^{(m)}$ determining their values at the node points $\boldsymbol{x}_i^{(m)}$:

$$H_i^{(m)}(\boldsymbol{x}) = \left\{ \begin{array}{lcl} 1 & : & \boldsymbol{x} = \boldsymbol{x}_i^{(m)} \\ 0 & : & \boldsymbol{x} = \boldsymbol{x}_j^{(m)}, j \neq i \end{array} \right. \qquad (4)$$

A further, pragmatic requirement is, that the sum of the shape-functions $H_i^{(m)}$ at an arbitrary point is given by:

$$\sum_{i=1}^{N} H_i^{(m)}(\boldsymbol{x}) = 1 \qquad (5)$$

In some cases spatial derivatives of the solution function, i.e. electrical field strength, enhance the description at node points. Commonly, the attributed shape-functions fulfill also (4). Different kinds of continuity of the interpolated function at the border of elements can be provided depending on the order of derivatives given at node points. Lagrangian elements, which include no derivatives, guaranty C_0 continuity. N-th order Hermitian elements provide C_n continuity. The nodal concept can be extended by defining variables at edge, face and element level in combination with suitable shape-functions.

Solving electrical field problems as described by (2) necessitates knowledge concerning the gradient of the potential $\nabla\phi$. Spatial derivation of (3) delivers this gradient:

$$\nabla\phi = \nabla \left(\boldsymbol{H}^{(m)\,T} \boldsymbol{\phi}^{(m)} \right) = \left(\nabla \boldsymbol{H}^{(m)\,T} \right) \boldsymbol{\phi}^{(m)} \qquad (6)$$

with the gradient of the shape-function's vector $\nabla \boldsymbol{H}^{(m)}$.

The element-wise definition of (2) and combination with (3) and (6) leads to:

$$\Pi^{(m)} = \int_{\Omega_0} \big(\frac{1}{2} (\nabla_0 \boldsymbol{H}^{(m)\,T} \boldsymbol{\phi}^{(m)})^T \sigma (\nabla_0 \boldsymbol{H}^{(m)\,T} \boldsymbol{\phi}^{(m)}) + f \; \boldsymbol{H}^{(m)\,T} \; \boldsymbol{\phi}^{(m)} \big) \; \mathcal{J} \; d\Omega_0 \qquad (7)$$

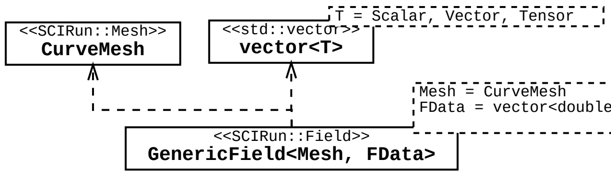
with the transformed gradient operator $\nabla_0 = J^{-1}\nabla$ and the Jacobian matrix J. In a similar manner, conductivity σ and current source density f can be interpolated inside of an element.

For each element a system of linear equations can be be created by derivation of (7) concerning the node variables. The unknowns of this system are the node variables, i.e. the description of the solution function at node points. Finally, the element-wise linear equations are assembled into the system equations, boundary conditions are incorporated and the system is solved with numerical techniques, e.g. conjugate gradient and multigrid methods.

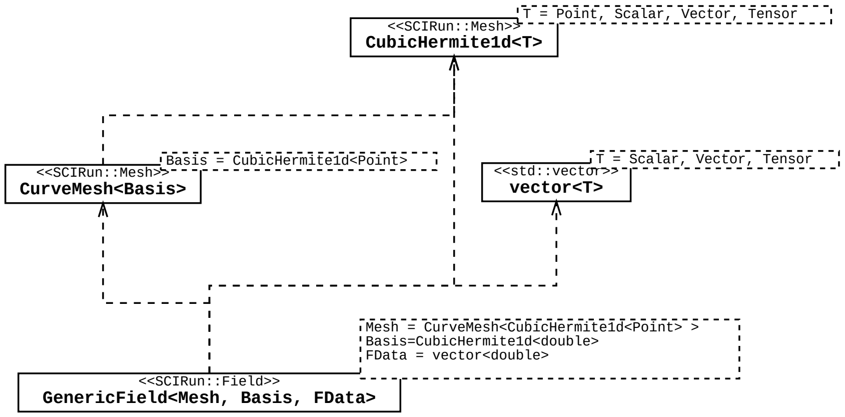
C. Status Quo of SCIRun/BioPSE

SCIRun includes classes for describing meshes and fields, which are extensively used in BioPSE for calculating of electrical fields. The field classes are derived from a mesh class and include a single data collection (Fig. 1a). Field classes are templated on these two parameters. Similar to the concepts implemented in the C++ Standard Template Library [9], the classes fields and meshes were designed to allow general algorithm coding. It is possible to implement an algorithm for fields such that it compiles and executes correctly with any specific type. The concept specifies the interface that all fields and meshes classes must have. It is the uniformity of interface that allows algorithms to be written once for the general case, and compiled for each specific field type. SCIRun has many such examples of generic algorithms. Most of the modules are written as generic algorithms and dynamically compiled at runtime based on the exact field input type [10].

There are currently sixteen field and mesh classes implemented all supporting the mesh concept. These implement very different topologies and dimensions. Fields classes range from structured to partially structured to unstructured. LatticeVol is an example of a structured field class. Nodes in the mesh are specified simply by a spacing and an extents number of nodes in x,y, and z. TetVol is an example of an unstructured field, where explicitly the node positions and all connectivity information is stored. In all cases the mesh handles the topology. The field



(a)



<<SCIRun::Mesh>>
CubicHermite1d<T>
T = Point, Scalar, Vector, Tensor
<<SCIRun::Mesh>>
CurveMesh<Basis>
Basis = CubicHermite1d<Point>
<<std::vector>>
vector<T>
T = Scalar, Vector, Tensor
Mesh = CurveMesh<CubicHermite1d<Point> >
Basis=CubicHermite1d<double>
FData = vector<double>
<<SCIRun::Field>>
GenericField<Mesh, Basis, FData>

(b)

Fig. 1. (a) Old and (b) new version of mesh and field concepts. The concepts differ concerning the availability of an arbitrary interpolation basis in the new version.

owns both a mesh and a set of data that is specified to exist at node, edge, face, or cell centers. In each case within cells the basis was linear, which has influence on some part of the concepts.

D. New Field/Mesh Concept

While generality was achieved in the interface, field and mesh interpolation were forced to be of low order within elements. Allowing the greater accuracy and flexibility of higher order basis functions, we designed new field and mesh concepts, which accommodate in a general way different kinds of basis. Here, meshes were only responsible for the topological organization of the nodes. A template parameter for describing the interpolation basis was added to the mesh classes, and all responsibility for how data changes within an element is calculated in the class basis. Fields similarly added a template parameter for describing the interpolation basis. Field variables can have a different interpolation basis than the mesh. Higher order basis require additional variables stored within the element, so memory savings can be employed by mixing these basis without sacrificing accuracy.

E. Interpolation Basis

An interpolation basis concept defines the role of the interpolation basis classes within interaction of the field, mesh, and interpolation basis classes. This interaction allows for generic algorithms to be compiled for any combination of the field, mesh, and interpolation basis classes. Fig. II-E shows an example of a general algorithm for interpolating a value at a given point in space.

An interpolation basis class has a template parameter of its own, that describes the type of values interpolated within the elements. In the case of meshes the parameter is typically a class describing points. For fields the parameter is commonly of type scalar, vector, or tensor, but can be any concrete type. Interpolation basis classes provide methods to interpolate values and first and second order derivatives, determining the parametric coordinates of a point, piecewise linear approximate edges and faces for an element, as well as interface to get the shape function values separately for a given coordinate for interpolation, first and second derivative. The interpolation basis class stores any additional data required to support the basis functions, apart from the data meshes or fields store, like derivatives for example. The interpolation basis has no information concerning the field or mesh that own them, but need to know values stored in these. To satisfy this some interface in the interpolation basis have an additional template argument. This additional parameter holds the interface to get values held by a field or mesh. Even within mesh type that it is easy to see that the mesh knows how to provide these values. The LatticeVol mesh has to calculate a point, where a TetVol can do a lookup to return the point. A single interpolation basis class type can be used for any mesh having

```
template <class FLD>
void interp(typename FLD::value_type &val, Point p, typename FLD::handle_type fld)
{
  typedef typename FLD::mesh_type Msh;
  typename FLD::mesh_handle_type mh = fld->get_typed_mesh();
  typename Msh::Elem::index_type ei;

  // The mesh finds the element that holds (or is nearest to) point p
  mh->locate(ei, p);

  // This vector of double accomodates any dimension
  vector<double> coords(3, .0L);

  // From the mesh and its basis find the parametric coordinates within the element
  mh->get_coords(coords, p, ei);

  // Ask the field to use its basis to calculate the field variable value at the parametric coordinate
  fld->interpolate(val, coords, ei);
}
```

Fig. 2. Exemplary general algorithm for interpolating a value given some point in space. Similar algorithms are available for interpolation of derivatives and for integration.

the same topology. A LatticeVol has the same topology as a an unstructured volume based on Cubes. Both can use the same set of basis implementations. Changes in dimension require different basis implementations.

III. Results

The design of the framework was tested by exemplary implementations of high order finite elements. The programming language C++ was employed. The previous field and mesh design had no flexibility with regards to interpolation basis as can be seen in Fig. 1a, which shows the template parameters required to create a one-dimensional field with a linear basis. The new design given in Fig. 1b shows a one-dimensional field that has scalar data, i.e. of type double. The field variables are described by a cubic hermitan basis. Also, the mesh uses a cubic hermitan basis for describing its geometry.

Several function were implemented which provide the basis for setup and solution of finite element models. Particularly, efficient methods were developed to calculate first order derivatives, which are fundamental for setting up system matrices.

IV. Discussion and Conclusion

We presented a software framework for high order finite elements to solve problems of bioelectricity. The software framework extends the current capabilities of SCIRun/BioPSE in such a manner that basis interpolation functions can be assigned to meshes and fields.

The software framework supports not only the frequently applied isoparametric elements, but also combinations of different interpolation types for geometry and fields, e.g. low order geometrical interpolation and high order field interpolation. The framework simplifies the modeling and solution processes of problems in bioelectricity due its general interface. Independently of the element type, field values and derivatives can be interpolated, e.g. at Gaussian point for numerical integration.

he design for addition of higher order basis functions in SCIRun maintains the generality that the SCIRun fields and meshes have enjoyed. The framework allows users to create their own basis classes and use them within existing SCIRun field and mesh classes. The generic algorithms that make up the core of most SCIRun modules will compile and run with their basis, assuming that the basis concept was adhered to.

In future work we will implement a variety of classes for three-dimensional interpolation of Lagrange and Hermition type. Additionally, methods for efficient interaction with and visualization of high order finite elements meshes will be developed. The framework will be applied in forward calculations, where the distribution of potentials is determined from electrical sources, e.g. currents in the heart.

References

[1] J. D. Jackson, *Classical Electrodynamics.* Berlin, New York: Wiley Text Books, 3 ed., 1998.

[2] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol*, vol. 177, pp. 500–544, 1952.

[3] K.-J. Bathe, *Finite-Elemente-Methoden.* Berlin, Heidelberg, New York: Springer, 1990.

[4] H. R. Schwarz, *Methode der finiten Elemente.* Stuttgart: Teubner, 3 ed., 1991.

[5] SCIRun: A Scientific Computing Problem Solving Environment. Scientific Computing and Imaging Institute (SCI), University of Utah, http://software.sci.utah.edu/scirun.html, 2002.

[6] BioPSE: Problem Solving Environment for modeling, simulation, and visualization of bioelectric fields. Scientific Computing and Imaging Institute (SCI), University of Utah, http://software.sci.utah.edu/biopse.html, 2002.

[7] R. Plonsey and D. B. Heppner, "Considerations of quasi-stationarity in electrophysiological systems," *Bulletin of mathematical biophysics*, vol. 29, pp. 657–664, 1967.

[8] F. B. Sachse, *Computational Cardiology: Modeling of Anatomy, Electrophysiology, and Mechanics.* LNCS 2966, Berlin, Heidelberg, New York: Springer, 2004.

[9] B. Stroustrup, *The C++ programming language.* Boston: Addison-Wesley, 3 ed., 1997.

[10] M. Cole and S. Parker, "Dynamic compilation of C++ template code," vol. 11, pp. 321–327, IOS Press, 2003.