Proceedings of the 28th IEEE
EMBS Annual International Conference
New York City, USA, Aug 30-Sept 3, 2006

A Software Framework for Solving Bioelectrical Field Problems Based on Finite Elements

F. B. Sachse[1,2], M. J. Cole[3], J. G. Stinstra[3]
[1]Nora Eccles Harrison Cardiovascular Research and Training Institute,
[2]Bioengineering Department, [3]Scientific Computing and Imaging Institute,
University of Utah, UT, USA

Abstract— **Computational modeling and simulation can provide important insights into the electrical and electrophysiological properties of cells, tissues, and organs. Commonly, the modeling is based on Maxwell's and Poisson's equations for electromagnetic and electric fields, respectively, and numerical techniques are applied for field calculation such as the finite element and finite differences methods. Focus of this work are finite element methods, which are based on an element-wise discretization of the spatial domain. These methods can be classified on the element's geometry, e.g. triangles, tetrahedrons and hexahedrons, and the underlying interpolation functions, e.g. polynomials of various order.**

Aim of this work is to describe finite element-based approaches and their application to extend the problem-solving environment SCIRun/BioPSE. Finite elements of various types were integrated and methods for interpolation and integration were implemented. General methods for creation of finite element system matrices and boundary conditions were incorporated. The extension provides flexible means for geometric modeling, physical simulation, and visualization with particular application in solving bioelectric field problems.

Keywords— **Finite element method, bioelectricity, SCIRun, Poisson equation,**

I. Introduction

Electrical activity in biological tissue can be described with Maxwell's and Poisson's equations for electromagnetic and electric fields, respectively. Various analytical and numerical techniques have been developed to solve the underlying systems of partial differential equations. Commonly, finite element, finite difference or boundary element methods are employed for numerical solution. Each of these methods necessitates interpolation schemes, which are major determinants for accuracy and computational demands of field calculations.

The finite element methods are based on interpolation of the solution field inside of an element. Mostly, the interpolation is defined by so-called basis or shape-functions [1], [2], [3]. A large family of interpolation schemes is based on polynomial functions. Typically, polynomials of order one to three are used for electrical field problems and the number of polynomial coefficients corresponds to the number of node variables describing the finite element. Higher order elements provide 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 finite element-based approaches were applied to extend the problem-solving environment SCIRun/BioPSE. The software was developed to support geometric modeling, simulation, and visualization for solving bioelectric field problems [4], [5]. The extension provides means to solve these tasks in meshes of various element types, interpolation functions and spatial dimension.

The work was supported by the NIH NCRR Center for Integrative Biomedical Computing. F. B. Sachse thanks for the kind support by the Richard A. and Nora Eccles Fund for Cardiovascular Research and the Nora Eccles Treadwell Foundation. The authors gratefully acknowledge the help of Drs. R. Kerckhoffs and A. McCulloch for testing the software.

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 electrical conductivity. A detailed description of the applicability of Poisson's in biological materials is given in [6].

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

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

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 0-th or 2nd order. Boundary conditions are added to solve the equation, e.g. Dirichlet and Neumann boundary conditions.

Eqn. 1 can be transformed to an equivalent integral representation [2], [7], describing the electrical power Π in a unit domain Ω_0 with the Jacobian of the coordinate transformation $\mathcal{J}$:

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

B. Finite Element Methods

The ideas underlying finite element methods were introduced with the classical work of Ritz and Galerkin. Commonly, the first step of application of finite element methods is a subdivision of the spatial domain Ω into finite elements. The solution function, i.e. the electrical potential ϕ, is interpolated within the element by 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 approximated by:

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

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

$$H_i^{(m)}(\boldsymbol{x}) = \begin{cases} 1 & : \quad \boldsymbol{x} = \boldsymbol{x}_i^{(m)} \\ 0 & : \quad \boldsymbol{x} = \boldsymbol{x}_j^{(m)}, j \neq i \end{cases} \tag{4}$$

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

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

In some cases spatial derivatives of the solution function enhance the description at node points, which allows to assure continuity of the interpolated function and to directly apply Neumann-type boundary conditions. Commonly, the attributed shape-functions also fulfill eqn. 4. Different kinds of continuity of the interpolated function at the element interfaces can be provided depending on the order of derivatives given at node points. Lagrangian elements, which include no derivatives, guarantee 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.

The following deviation illustrates the application of the finite element method to solve Poisson's equations for electrical current fields. Similar approaches apply to solve other physical field problems, e.g. in mechanics and optics. 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:

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

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:

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

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 is created by derivation of eqn. 7 concerning the node variables. Unknowns of this system are the node variables, i.e. the description of the solution function at node points. Finally, the element-wise 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. Implementation

The source code of SCIRun/BioPSE includes C++ classes for describing meshes and fields, which are applied for management and visualization of geometrical and physical models. Meshes serve for topological organization of the elements. A template parameter of the mesh classes allows specification of an interpolation basis class. Each field class is derived from a mesh class and includes a single data collection. Field classes are templated on these two parameters. Similar to the concepts employed by the C++ Standard Template Library [8], the classes fields and meshes were designed to allow general algorithm coding composition.

Sixteen different mesh classes support the mesh concept in the SCIRun/BioPSE framework. These implement very different topologies and dimensions. The classes describe structured to partially structured to unstructured meshes. Examples of these mesh classes are a LatVol class describing a regular lattice of hexahedral elements, and a TetVol class describing unstructured tetrahedral mesh whose node positions and connectivity are stored explicitly. In all cases the mesh handles the topology. A field owns both a mesh and a set of data that is specified to exist at nodes, edges, faces, and/or cell centers.

Interpolation basis classes have a template parameter 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, depending on the type of physical field. Interpolation basis classes provide methods to interpolate values and first 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 and first derivative. The interpolation basis class stores 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 owns them, but needs to know values stored in these in order to interpolate values.

An interpolation basis concept defines the role of the interpolation basis classes within interaction of

TABLE I
Implemented element types and interpolation functions

Type	**Dimension**	**Interpolation functions**
Curve	1	Linear and quadratic Lagrangian, cubic Hermitian
Triangle	2	Linear and quadratic Lagrangian, cubic Hermitian
Quad	2	Bilinear and biquadratic Lagrangian, bicubic Hermitian
Tetrahedron	3	Linear and quadratic Lagrangian, cubic Hermitian
Prism	3	Bilinear and biquadratic Lagrangian, bicubic Hermitian
Tetrahedron	3	Trilinear and triquadratic Lagrangian, tricubic Hermitian with scale factors

the field, mesh, and interpolation basis classes. This interaction allows for generic algorithms to be compiled for specific combination of the field, mesh, and interpolation basis classes. Appendix A gives an example of a general algorithm for interpolating a value at a given point in space.

Responsibility for how data changes within an element is determined in the interpolation classes. Similarly, a template parameter describes the interpolation basis in the fields classes. Fields can have an interpolation basis that differs from the mesh.

D. Analytical Derivation of Interpolation Functions

Interpolation functions were derived by solving the following equation system for the various element and interpolation types:

$$\phi_i^{(m)} = P(\boldsymbol{x}_i^{(m)}) \tag{8}$$

with the polynomial P with unknown coefficients, the i-th node value $\phi_i^{(m)}$, and the i-th point $\boldsymbol{x}_i^{(m)}$. The coefficients of the polynomial P were determined by symbolic evaluation. Reordering of the solution yields the shape-functions $H_i^{(m)}$. An exemplary implementation for derivation of an one-dimensional interpolation function is given in Appendix B.

III. Results

The SCIRun/BioPSE software framework was extended by introducing finite element approaches. A variety of interpolation basis classes for one- to three-dimensional interpolation of Lagrangian and Hermitian type were developed (Tab. I). The classes are applied as template parameters for meshes and fields. Additionally, methods for efficient interaction with and visualization of finite elements meshes were implemented.

Several methods were implemented which provide a basis for setting up and solving complex finite element models. Particularly, efficient methods were developed to calculate first order derivatives and Jacobians, which are fundamental for computing system matrices in bioelectrical field problems. Independent of the element type, field values and first order derivatives can be interpolated, e.g. at Gaussian points for numerical integration.

An example of the meshes that were generated with the software is depicted in figure 1. The figure shows a model of the heart described by a tricubic Hermitian hexahedral mesh.

IV. Discussion and Conclusion

We presented a software framework based on finite elements to solve problems of bioelectricity. The framework can be applied in forward calculations, where the distribution of potentials is determined from electrical sources, e.g. currents in the heart.

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.

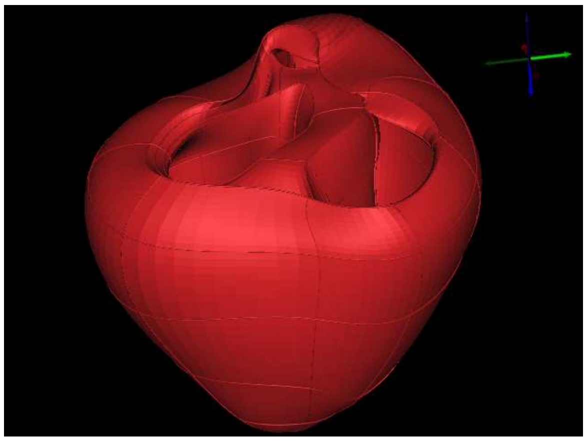
The software framework supports not only the frequently applied isoparametric elements, but also combinations of different interpolation types for geometry and fields, e.g. low order geometrical interpolation and high order field interpolation. The framework simplifies the modeling and solution processes of problems in bioelectricity due its general interface.

This design for the addition of finite-element basis functions in SCIRun maintains the generality of the previous design of SCIRun fields and meshes. The framework allows users to create their own basis classes and use them within existing SCIRun field and mesh classes.

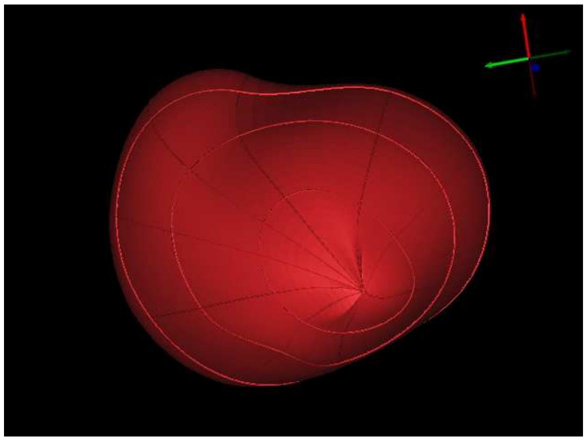
V. Appendix

A. Exemplary General Algorithm

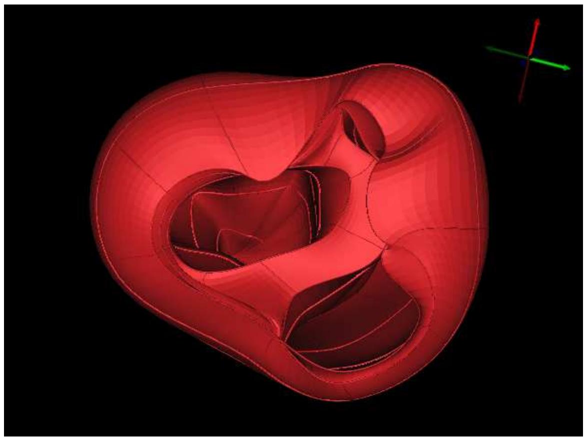
```
template <class FLD>
void interpolate(typename FLD::value_type &val,
                 Point p,
                 typename FLD::handle_type fld)
{
  typedef typename FLD::mesh_type Msh;
  typename FLD::mesh_handle_type
          mh = fld->get_typed_mesh();
  typename Msh::Elem::index_type ei;

  // Find element that holds point p
  mh->locate(ei, p);

  // This vector of double accomodates any dimension
  vector<double> coords(
          Msh::basis_type::domain_dimension(), .0L);

  // Find the local coordinates within the element
  mh->get_coords(coords, p, ei);

  // Calculate the field value at the local coordinate
  fld->interpolate(val, coords, ei);
}
```



(a)



(b)



(c)

Fig. 1. Visualization of cardiac ventricles from (a) frontal, (b) apical, and (c) basal. The ventricles are described by a tricubic Hermitian hexahedron mesh with scale factors.

B. Exemplary Derivation of Interpolation Functions

Interpolation functions were derived with Mathematica [9]. A script for derivation of an one-dimensional cubic Hermitian interpolation based on a cubic polynomial for an one-dimensional curve is described in the following. The script produces C++ code.

```
dim = 2;
XY = {0, 1};
phi = Table[phiv[i], {i, 1, dim}];
gradphi = Table[ex[i], {i, 1, dim}];
c = Table[cv[i], {i, 1, dim * 2}];
f[x_] := c.{1, x, x^2, x^3};

fx[x_] := D[f[xx], xx] /. xx → x
A = Table[{
      f[XY[[i]]] == phi[[i]],
     fx[XY[[i]]] == gradphi[[i]]
    }, {i, 1, dim}];
A = Flatten[A];
Subst = Solve[A, c];

Shapefunctions = Table[{
     CoefficientList[f[x] /. Subst,
        phi[[i]]] [[1]][[2]],
     CoefficientList[f[x] /. Subst,
        gradphi[[i]]][[1]][[2]]
    }, {i, 1, 2}];
Shapefunctions = Flatten[Shapefunctions];
MatrixForm[Simplify[%]]
CForm[%]
```

$$\begin{pmatrix} (-1+x)^2\,(1+2\,x) \\ (-1+x)^2\,x \\ (3-2\,x)\,x^2 \\ (-1+x)\,x^2 \end{pmatrix}$$

```
List(Power(-1 + x,2)*(1 + 2*x),Power(-1 + x,2)*x,(3 -
2*x)*Power(x,2),(-1 + x)*Power(x,2))
```

```
ShapefunctionsDerivatives =
 Table[D[Shapefunctions[[i]], {x}[[j]]],
   {i, 1, dim * 2}, {j, 1, 1}];
MatrixForm[Simplify[ShapefunctionsDerivatives]]
CForm[%]
```

$$\begin{pmatrix} 6\,(-1+x)\,x \\ 1-4\,x+3\,x^2 \\ -6\,(-1+x)\,x \\ x\,(-2+3\,x) \end{pmatrix}$$

```
List(List(6*(-1 + x)*x),List(1 - 4*x +
3*Power(x,2)),List(-6*(-1 + x)*x),List(x*(-2 + 3*x)))
```

References

[1] K.-J. Bathe, *Finite Element Procedures in Engineering Analysis.* Englewood Cliffs/NJ: Prentice Hall, 1982.

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

[3] B. Szabó and I. Babuška, *Finite Element Analysis.* New York: John Wiley & Sons, 1991.

[4] SCIRun: A Scientific Computing Problem Solving Environment. Scientific Computing and Imaging Institute (SCI), University of Utah.

[5] BioPSE: Problem Solving Environment for modeling, simulation, and visualization of bioelectric fields. Scientific Computing and Imaging Institute (SCI), University of Utah.

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

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

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

[9] S. Wolfram, *Das Mathematica Buch.* Bonn: Addison-Wesley, 3 ed., 1997.