 cmes, vol.5, no.4, pp.331-345, 2004

A Discrete Differential Forms Framework for Computational Electromagnetism [1]

P. Castillo, [2] **J. Koning,** [3] **R. Rieben** [4] **and D. White** [5]

Abstract: In this article, we present a computational framework for solving problems arising in electromagnetism. The framework is derived from a modern geometrical approach and is based on differential forms (or *p*-forms). These geometrical entities provide a natural framework for modeling of physical quantities such as electric potentials, electric and magnetic fields, electric and magnetic fluxes, etc. We have implemented an object oriented class library, called FEMSTER. The library is designed for high order finite element approximations. In addition, it can be expanded by including user-defined data types or by deriving new classes. Finally, the versatility of the software is shown through different simulations.

keyword: High order finite element, $H(div)$ and $H(curl)$ - conforming methods, Computational electromagnetism, Object oriented programming.

1 Introduction

The finite element method is the most popular numerical method for structural mechanics and is increasingly popular for numerous other disciplines such as fluid dynamics, particle and radiation transport, thermodynamics, and electromagnetism. The popularity is due to the ease in which complex geometries can be modeled via unstructured computational meshes, the ability to incorporate complex material properties and boundary conditions into the discretization, and the robust foundation of functional analysis. In electromagnetics engineering the most interesting and challenging problems involve either smoothly curved surfaces (radar cross section of aircraft, linear accelerator components), intricate geometric details (microwave integrated circuits, broadband antennas), or material inhomogeneities (electromagnetic fields in the earth, the human body, or man-made metamaterials). The finite element method is well suited to these types of problems. There are many open areas of research in finite element electromagnetics such as mesh generation, fast linear solvers, and efficient and accurate radiation boundary conditions. In this paper we focus our attention on discretization, specifically the recently proposed differential forms based approach for constructing curl-conforming (also known as $H(curl)$ or "edge") elements and divergence-conforming (also known as $H(div)$ or "face") elements.

The equations of electromagnetics can be simply and elegantly cast in the language of differential forms Deschamps (1981), Baldomir (1986), Burke (1985), Bossavit (1998). In this approach the scalar electrostatic potential is a 0-form, the electric and magnetic fields are 1-forms, the electric and magnetic fluxes are 2-forms, and the scalar charge density is a 3-form. The basic operators are the exterior derivative, the wedge product, and the Hodge star. Precise rules (i.e. a calculus) prescribe how these forms and operators can be combined. In this modern geometrical approach to electromagnetics the fundamental conservation laws are not obscured by the details of coordinate system dependent notation.

In the context of the Galerkin procedure applied to the vector Helmholtz equation or to the time-dependent Maxwell's equations, there are significant advantages to curl-conforming finite element basis functions. These basis functions were first proposed by Nédélec (1980), and excellent overviews can be found in the textbooks by Jin (1993) and Bossavit (1998). These advantages include the proper modeling of the jump discontinuity of electric fields across material discontinuities, the

[1] This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory

[2] Department of Mathematics, University of Puerto Rico castillo@math.uprm.edu

[3] Defense Sciences Engineering Division, Lawrence Livermore National Laboratory, koning1@llnl.gov

[4] University of California, Davis and Institute for Scientific Computing Research, Lawrence Livermore National Laboratory, rieben1@llnl.gov

[5] Defense Sciences Engineering Division, Lawrence Livermore National Laboratory, white37@llnl.gov

elimination of spurious modes in eigenvalue computations, and the conservation of charge in time-dependent simulations Rodrigue and White (2001). Originally the development of these basis functions was presented in an ad-hoc manner, with many different variations without a common theme. Recently, a connection between differential forms and the curl-conforming basis functions has been established Hiptmair (1999). In Hiptmair (1999) the theory of differential forms is used to show that the properties of curl-conforming basis functions can be derived from first principles, and that the curl-conforming basis is one member of a family of basis functions that includes the standard scalar C^0 basis and the divergence-conforming basis used in the mixed method for the Stokes problem. In our terminology, a discrete differential *p*-form is a finite element basis function used to discretize a p-form field. Given a physical law expressed in the language of differential forms, it is quite straightforward to discretize the problem using our class library of discrete differential forms.

It should be noted that for a Cartesian grid, and with appropriate quadrature rules for bilinear forms, our finite element approach is equivalent to the classic FDTD scheme. The FDTD scheme is known to be very efficient and it has the important properties of being stable, conserving energy, conserving charge, and correctly modeling the jump discontinuity of fields across material interfaces. A state-of-the-art parallel FDTD code is described in this Special Issue Namburu, Mark, and Clarke (2004).

The second focus of this paper in on high-order discretization. In many electromagnetic design and analysis problems there is little modeling error; the equations are known, the geometry is known, the material properties are known. The only approximation is the numerical method, hence there can be significant benefit to high-order approximations. A high-order discretization can reduce the mesh size, memory usage, and CPU time required to achieve a prescribed error. This is particularly true for electrically large problems due to numerical dispersion. The Galerkin procedure applied to wave equations suffers from numerical dispersion, which means that the computed phase velocity differs from the physical phase velocity and phase error builds up linearly with respect to distance and time. For the popular lowest order edge elements, it is known that the numerical dispersion relation is second order accurate Warren and Scott (1994), Warren and Scott (1995), White (2000). Second order accuracy may seem adequate, but for an electrically large problem the phase error may be such that the global error is 100 percent, although the local truncation error is quite small. The phenomena of the global error being significantly greater than the local (or optimal) error for a Galerkin solution of wave equations is sometimes referred to as the pollution effect. This has been more precisely explained in Babuska, Strouboulis, Upadhyay, and Gangaraj (1995), Babuska, Ihlenburg, Strouboulis, and Gangaraj (1997). The differential forms based approach is not the only approach for deriving higher-order methods. For example, in this Special Issue a higher-order discontinuous Galerkin (DG) method is described Hesthaven and Warburton (2004). The higher-order DG method also has the ability to model complex geometry and also has significantly reduced numerical dispersion compared to standard low-order methods.

To conclude the Introduction, in this paper we review an object oriented class library of discrete differential forms. This library consists of geometric elements, p-form bases of degree 0,1,2 and 3, integration rules, and bilinear forms. All of these classes are of arbitrary order, although due to finite precision effects the maximum order used in practice is limited. We present problems in electrostatics, driven time-harmonic problems, time harmonic eigenvalue problems, and time-domain transient problems expressed in the language of differential forms, and show computed solutions to these problems using our class library of discrete differential forms.

2 Mathematical preliminaries

We begin with the generic boundary value problem stated in the language of differential forms from Hiptmair (2001). We assume a 3-dimensional domain Ω with piecewise smooth boundary $\partial\Omega$ partitioned into Γ_D, Γ_N, and Γ_M. The problem statement is

$$du = (-1)^p \sigma, \;\; d\,j = -\Psi + \Phi \;\; in \;\; \Omega \tag{1}$$

$$T_D u = 0 \;\; on \;\; \Gamma_D, \;\; T_N\, j = 0 \;\; on \;\; \Gamma_N \tag{2}$$

$$j = \star_\alpha \, \sigma, \;\; \Psi = \star_\gamma \, u \;\; in \;\; \Omega \tag{3}$$

$$T_M j = (-1)^p \star^{\beta} T_M u \ \text{ on } \ \Gamma_M. \tag{4}$$

Here u is a $(p-1)$-form, σ is a p-form, j is a $(3-p)$-form, and both Ψ and Φ are $(3-p+1)$-forms, where $1 \leq p \leq 3$. The variable Φ is a source term. In (1) the operator d is the exterior derivative which maps p-forms to $(p+1)$-forms. In the boundary conditions (2) and (4) the symbol T denotes the trace operator, where the trace of a p-form is an integral over a p-dimensional manifold. In (3) and (4) the $\star$ symbol denotes the Hodge-star operator, which converts p-forms to $(3-p)$-forms and typically involves material constitutive properties. Equations (1) and (3) can be combined to yield the general second-order elliptic equation

$$(-1)^p d \star_{\alpha} du = - \star_{\gamma} u + \Phi. \tag{5}$$

We are also concerned with time dependent phenomena. The time derivative does not effect the degree of a form. In the diagram below we show the time-dependent Maxwell's equations, where d denotes the spatial derivative and dt denotes the time derivative and converging arrows denote summation. In these diagrams ϕ is the scalar potential 0-form, the 1-forms A, E, and H are the magnetic vector potential, the electric field, and the magnetic field, respectively; the 2-forms B, D, and J are the magnetic flux density, the electric flux density, and the electric current density, respectively; and ρ is the scalar charge density 3-form. The left diagram encompasses Faraday's law $dE - dB/dt = 0$, Coulomb's law for the magnetic field $dB = 0$, and the fact that the electric field E can be written in terms of potentials as $E = d\phi - dA/dt$. The right diagram encompasses Ampere's law $dH - dD/dt = J$, Coulomb's law for the electric field $dD = \rho$, and the continuity equation $dJ - d\rho/dt = 0$. The two diagrams are connected by the constitutive relations $D = \star_{\varepsilon} E$ and $B = \star_{\mu} H$.

$$\begin{array}{lccccc}
0\text{-forms}: & & \phi & & & \\
 & & \downarrow d & & & \\
1\text{-forms}: & A & \xrightarrow{-dt} & E & & H \\
 & \downarrow d & & \downarrow d & & \downarrow d \\
2\text{-forms}: & B & \xrightarrow{-dt} & 0 & D \xrightarrow{-dt} & J \\
 & \downarrow d & & & \downarrow d & \downarrow d \\
3\text{-forms}: & 0 & & & \rho \xrightarrow{-dt} & 0
\end{array}$$

In the Galerkin finite element procedure we require bilinear forms. These are easily generated from the general second-order equation (5) by taking the wedge product with an $(l-1)$-form v and integrating over the volume Ω,

$$\int_{\Omega} (-1)^l d \star_{\alpha} du \wedge v = - \int_{\Omega} \star_{\gamma} u \wedge v + \int_{\Omega} \Phi \wedge v. \tag{6}$$

Using the integration-by-parts formula

$$\int_{\Omega} d\omega \wedge \eta + (-1)^l \int_{\Omega} \omega \wedge d\eta = \int_{\partial\Omega} \omega \wedge \eta \tag{7}$$

yields the two key symmetric bilinear forms

$$a(u,v) \ = \ \int_{\Omega} \star_{\alpha} (du) \wedge dv, \tag{8}$$

$$b(u,v) \ = \ \int_{\Omega} \star_{\gamma} u \wedge v. \tag{9}$$

With these bilinear forms we can construct a great variety of model equations that can be solved via the finite element method. While above we discussed the standard elliptic boundary value problem, it is a simple matter to introduce temporal derivatives into the model problem and perform temporal evolution using the method-of-lines approach. The key point is that in order for the finite procedure to work, it is necessary to use the proper p-form basis functions when discretizing the above bilinear forms. The essential properties of the p-form basis functions are discussed in Nédélec (1980), Hiptmair (1998), Hiptmair (2001).

3 FEMSTER : a finite element class library

FEMSTER is a modular finite element class library for solving three-dimensional problems arising in electromagnetism using high order approximations. The library is easy to use and provides a framework in which the user is able to add new schemes by reusing the existing classes and is able to expand the library by incorporating new user-defined data types.

An object-oriented programming (OOP) paradigm provides a natural and straightforward way of achieving our goals in a single computational framework. Our implementation benefits from three OOP concepts: *abstract data types, inheritance, and data encapsulation*. We use abstract base classes as computational devices to represent general mathematical objects such as elements, integration rules, polynomial bases, etc. The definition of abstract interfaces models the functionality of the class at the highest level of abstraction, keeping implementation details to the concrete derived classes. Typical methods included in the interface are evaluation of basis functions and their derivatives, local and global coordinate transformations, and generation of mass and stiffness matrices.

The concept of inheritance avoids the *redevelopment* and *testing* of existing code, by *reusing* base class members (data and methods). Moreover it allows the possibility of *extending* the library by including user-defined data types which can be derived from the existing base classes. Finally, by hiding internal details while providing a public interface, data encapsulation prevents unexpected modifications of data, making the code more robust and modular.

The philosophy of the FEMSTER library is derived from the formulation of an abstract conforming finite element method, see Ciarlet (1978). From the implementation point of view, such a formulation is uniquely determined by the $4-$tuple (Σ, P, A, Q) where:

- Σ is a reference element.
- P is a polynomial space defined on Σ.
- A is the set of degrees of freedom.
- Q is a quadrature rule defined on Σ.

This abstract formulation can be easily translated into a practical modular code by using an Object-Oriented Programming (OOP) paradigm. The C++ programming language, Stroustrup (1991), was used in the current implementation. In the following subsections we describe the classes that form the core of the FEMSTER library.

3.1 The class Element3D

This abstract class provides a common interface among various reference elements of different shapes. Currently the library supports three types of 3D elements: tetrahedron, hexahedron, and prism. The class inheritance is shown in Figure 1.

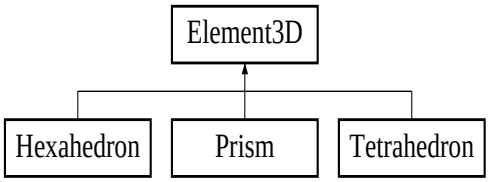

Figure 1 : Element3D class inheritance

The interface, although minimal, provides sufficient geometrical information to compute the local matrices and load vectors of a conforming finite element method. The interface is shown in Table 1.

Table 1 : Interface of the Element3D class

Method	Description
getOrder()	get the order of geometry
getNodes()	get the coordinates of the nodes
setNodes()	set the coordinates of the nodes
jacobian()	get the Jacobian matrix at a point
localToGlobal()	get global coordinates
globalToLocal()	get local coordinates

The geometry of an arbitrary element is uniquely defined by the physical coordinates of its nodes, its geometrical order and a local mapping that transforms the reference element onto the actual element. These are commonly known in the finite element literature as iso-parametric elements, see for example Ciarlet (1978). A block of elements of the same type can be represented by a single reference element. This is more natural from the mathematical point of view since all the elements in that block are topologically equivalent to a single reference element. Geometrical information of a particular element in that

block can then be obtained by setting the physical coordinates of the nodes in the reference element and querying the reference element for the desired information.

3.2 The class IntRule3D

Local integrals are computed numerically by using numerical integration rules. These should be exact for arbitrary high order polynomials and different element shapes. The inheritance diagram for this abstract class is shown in Figure 2.

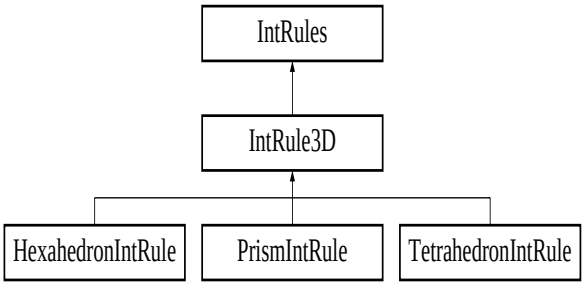

Figure 2 : IntRule3D class inheritance

The library provides integration rules of arbitrary order for all the element shapes. These are based on weighted Gauss-Jacobi quadrature rules. Our low order integration rules are optimal, the higher order rules are based on a tensor product of Gaussian quadratures which while accurate may not be optimal. However, the class has been designed to be extensible, the user can provide their own integration rules if desired. The global interface of the IntRule3D class is shown in Table 2 below.

Table 2 : Interface of the IntRule3D class

Method	Description
getNumPts()	get number of quadrature points
getOrder()	get order of exactness
getPoints()	get array of integration points
getWeights()	get array of integration weights
getRegion()	get region tag
getIntegral()	get approximation of the integral

3.3 The class p-form

3.3.1 Polynomial spaces

The first step in a finite element approximation is to choose the appropriate finite element space P from which the basis will be constructed. Usually this space consists of a particular set of polynomials. Typical examples are the nodal or Lagrangian polynomial space for the standard H^1 conforming method, Ciarlet (1978); the edge elements proposed by Nédélec for $H(curl)$ - conforming methods, Nédélec (1980, 1986); and, the face elements proposed by P.A Raviart and J.M Thomas for $H(div)$ - conforming methods, Raviart and Thomas (1977)

When implementing the finite element space P in the context of differential forms, the explicit formulation of the space depends on the *p*-form and the topology of the reference element Σ . The construction of the finite element space P is not unique, we choose a construction that leads to a simple and efficient implementation. We use polynomials similar to those described in Graglia, Wilton, and Peterson (1997) and Graglia, Wilton, Peterson, and Gheorma (1998) as a *primitive basis*. However these are constructed on the reference element Σ rather than in the physical coordinate system. The actual bases used in the finite element procedure are written as a linear combination of the primitive basis. For example, non-uniform interpolatory functions, moment-based functions, orthogonal functions, etc. can all be expressed in terms of the primitive basis.

3.3.2 Degrees of freedom

The set A of degrees of freedom is a finite subset of P', (i.e. the set of linear functionals from P onto $\Re$ Ciarlet (1978)), and satisfies three important properties; namely

- *Unisolvence*: $\{\alpha_i\}$ is dual to the finite element space P ; i.e. there exists a set $\{w_j\} \subset P$ such that $\alpha_i(w_j) = \delta_{i,j}$.
- *Invariance*: degrees of freedom remain unisolvent upon a change of variables; this implies they are not affected by the pullback operation; i.e. $\hat{\alpha}_i \circ \Phi^* = \alpha_i$ (see section 3.4).
- *Locality*: the trace of a basis function on a subsimplex is determined by degrees of freedom associated *only* with that sub-simplex.

The set A is defined in terms of moment integrals over sub-simplices of the reference element Σ Hiptmair (1999). If we denote a sub-simplex of the reference element Σ of dimension n as Σ_n, then the generalized form for the linear mapping is given by

$$\{\alpha_i\} = \{w \mapsto \int_{\Sigma_n} w \wedge q_n\}, \tag{10}$$

where $p \leq n \leq 3$ and q_n is an $(n-p)$-form *weighting polynomial* of n-variables defined over the sub-simplex Σ_n.

As an example, consider the degrees of freedom for 1-forms. For this case, $p = 1$ and we have the following three sets of moments which will require line integrals over edges weighted by 1-dimensional 0-forms (i.e. 1-dimensional scalar functions), surface integrals over faces weighted by 2 dimensional 1-forms (i.e. vector functions defined in a plane) and volume integrals over the element weighted by 3-dimensional 2-forms (i.e. vector functions defined in a volume).

$$\alpha(\mathbf{w}) = \int_{\hat{e}} (\mathbf{w} \circ \Phi) \cdot \partial\Phi^T(\vec{\mathbf{t}} q),$$
$$\alpha(\mathbf{w}) = \iint_{\hat{f}} (\mathbf{w} \circ \Phi) \cdot \partial\Phi^T(\vec{\mathbf{n}} \times \mathbf{q}),$$
$$\alpha(\mathbf{w}) = \iiint_{\vec{v}} (\mathbf{w} \circ \Phi) \cdot \partial\Phi^T \mathbf{q},$$

where $\vec{\mathbf{t}}$ denotes the unit tangent vector for each of the edges of Σ and $\vec{\mathbf{n}}$ denotes the unit normal vector for each of the faces of Σ . By appropriately using the iso-parametric mapping Φ and its derivative $\partial\Phi$, we can perform the integrals over the reference element, while maintaining generality of the function $\mathbf{w}$ (i.e. $\mathbf{w}$ can be defined over an arbitrary element).

While this formal definition is sufficiently general to define degrees of freedom of arbitrary type, we find that in practice there are simplifications that can be made to this definition for the particular case of *interpolatory* bases that can significantly improve computational performance. In this case, the integral forms of the degrees of freedom are replaced with simple point evaluation operations. For example, the scalar valued 0-form interpolatory bases can have their degrees of freedom reduced to the familiar form

$$\{\alpha_i(w)\} = w(\Phi(x_i)), \tag{11}$$

where x_i is a particular interpolation point in the reference coordinate system. Similarly, the vector valued 1-form interpolatory bases can have their degrees of freedom reduced to the form

$$\{\alpha_i(\mathbf{w})\} = \mathbf{w}(\Phi(x_i)) \cdot \partial\Phi^T(\vec{\mathbf{t}}_i), \tag{12}$$

where x_i is a particular interpolation point in the reference coordinate system and $\vec{\mathbf{t}}_i$ is an "interpolation vector" associated with the point x_i. Table 3 summarizes these simplified linear functionals for interpolatory degrees of freedom.

Table 3 : Simplified Degrees of Freedom for Interpolatory Bases

Form	Linear Functional
0-forms	$\{\alpha_i(w)\} = w(\Phi(x_i))$
1-forms	$\{\alpha_i(\mathbf{w})\} = \mathbf{w}(\Phi(x_i)) \cdot \partial\Phi^T(\vec{\mathbf{t}}_i)$
2-forms	$\{\alpha_i(\mathbf{w})\} = \mathbf{w}(\Phi(x_i)) \cdot \lvert\partial\Phi\rvert\partial\Phi^{-1}(\vec{\mathbf{n}}_i)$
3-forms	$\{\alpha_i(w)\} = \lvert\partial\Phi\rvert w(\Phi(x_i))$

Using this general approach, we can construct a discrete differential p-form basis of order k in the following manner. We begin by generating a primitive basis $W = \{w_j\}$. This primitive basis can be made in a number of different ways depending on the degree of the form and topology of the element. For example, a primitive basis on a reference hexahedron, we can form a primitive basis by taking tensor direct products of 1-dimensional Lagrange interpolatory polynomials. In order to construct a new basis (non-uniform interpolation, hierarchical, etc.) from the primitive basis we first formulate the linear functionals for A using the appropriate weighting polynomials. The choice of weighting polynomials in the formal definition of the degrees of freedom will determine the type of the new basis. For example, by choosing orthogonal weighting polynomials, the new basis will be hierarchical. We then apply the change-of-basis operation to the primitive basis; i.e. we construct the matrix

$$V_{i,j} = \alpha_i(w_j); \;\; w_j \in W \tag{13}$$

This system, which is similar to a Vandermonde matrix, is a linear mapping which expresses the new basis in terms of the primitive basis and will have a rank equal to the dimension of the primitive basis. We know from the definition of the degrees of freedom that the unisolvence property must hold for the new basis; so in order to

satisfy this requirement, we must find the inverse of the Vandermonde matrix. The newly defined basis, which we will denote as F will then have the form:

$$F = V^{-1}W \tag{14}$$

Therefore, the newly defined basis will be a linear combination of the primitive basis that spans the space P (i.e. satisfies the unisolvence property). While it may seem computationally expensive to invert a matrix in order to get the new basis, it should be noted that this is a one time cost as the inverse of the Vandermonde matrix can be stored and used over and over as necessary.

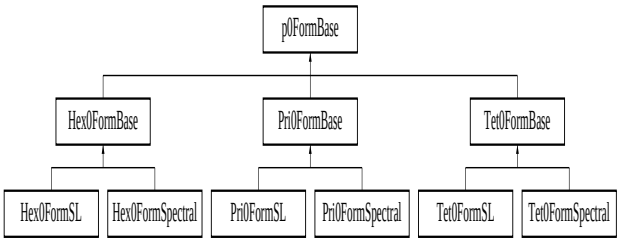

Figure 3 : 0-Form class inheritance

3.3.3 p-Form class interface

We have a class hierarchy for each of the *p*-form bases, the hierarchy for the 0-form class is shown in Figure 3. Concrete classes are presented in the lowest level of the tree. The other *p*-forms have a similar inheritance diagram. Our Silvester-Lagrange (SL) bases are similar to the bases defined in Graglia, Wilton, and Peterson (1997) which use equidistant and shifted equidistant interpolation points. The difference between our SL bases and the bases proposed in Graglia, Wilton, and Peterson (1997) is that ours satisfy the properties in Table 3. These are suitable for low order approximations, *i.e.*, *k* = 1 to 4. It is well known that this particular choice of interpolation points produce badly conditioned mass and stiffness matrices when high order approximations are used. For this reason we have implemented spectral classes that use arbitrary sets of interpolation points, typically Gauss-Lobatto or Chebyshev points Rieben, White, and Rodrigue (2004b). As an example, figure 4 shows the number of iterations required for a conjugate gradient algorithm to solve the linear system (with an error tolerance of 10^{-10}) arising from the discretization of Poisson's equation using a 0-form basis on a hexahedral mesh (see section 4.1). In this example we show the results for two different types of interpolatory bases. Note that the results for the uniform SL basis show exponential growth of iteration count as the approximation order is increased while the spectral (or Extended Chebyshev) basis shows polynomial growth. The user can also experiment by passing their own set of interpolation points into the constructor.

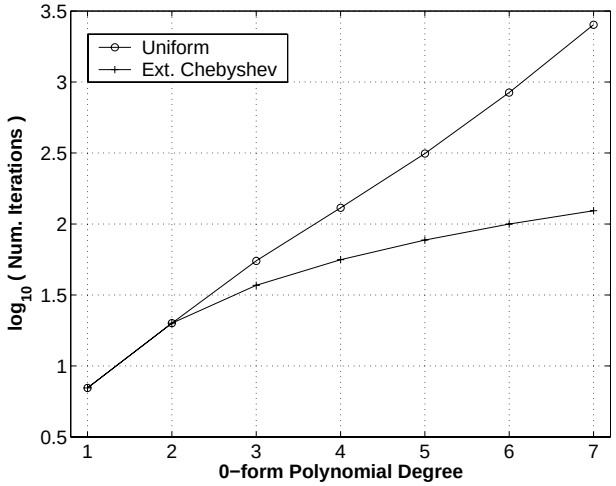

Figure 4 : Iteration count for diagonally scaled conjugate gradient solution of Poisson's equation using two different interpolatory bases.

The interface of a *p*-form includes methods that can be used in the computation of the local matrices and vectors, such as methods to evaluate basis functions and their derivatives at an arbitrary point, to evaluate the interpolate and its derivative and to compute the expansion of the interpolate of a given function. In Table 4 we show part of the interface of a *p*-form class.

The method localEvaluateD() computes the action of a differential operator on the basis functions at a given point. This operator is the exterior derivative and is uniquely determined by the *p*-form, see Abraham, Marsden, and Ratiu (1996), Burke (1985) for a classical geometrical approach. In particular, this operator refers to the gradient for 0-forms; the curl for the 1-forms; and finally, the divergence for the 2-forms.

To facilitate the assembly of global mass and stiffness matrices, the basis functions are locally sorted in the following order : nodal basis functions, edge basis functions, face basis functions, and interior basis functions.

Table 4 : Interface of a p-form class

Method	Description
getOrder()	get the order of the p-form
getDim()	get the dimension of the p-form
setElement()	set the element pointer
clearElement()	clear the element pointer
getConnectivity()	get the connectivity
localEvaluate()	$\phi_i(x), \quad i = 1,\ldots,n$
localEvaluateD()	$d\phi_i(x), \quad i = 1,\ldots,n$
localInterp()	$\Pi(f)(x)$
localInterpD()	$d\Pi(f)(x)$
project()	α_i where $\Pi(f) = \Sigma\alpha_i\phi_i$

The getConnectivity() method returns the number of basis functions per node, per edge, per face, and per cell.

3.4 Bilinear forms

The purpose of this class is to provide an interface to compute mass and stiffness matrices as well as load vectors. We present the derivation of the global stiffness matrix in the context of differential forms. In this setting, several elliptic problems such as div-grad, curl-curl and grad-div can be formulated in a single theoretical framework.

Let T_h be a triangulation of a physical domain Ω using tetrahedral, hexahedral, or prismatic elements. We consider the general bilinear form $a(\cdot,\cdot)$ defined by

$$a(u,v) \;=\; \int_\Omega *_\alpha(du)\wedge dv, \tag{15}$$

where $*_\alpha$ is the Hodge operator associated to a symmetric definite positive tensor α, which typically represents material properties such as electric and magnetic permeabilities and conductivities; and, u and v are both p-forms. Then by using the properties of the Hodge operator and the local change of variables given by the iso-parametric mapping $\Phi(\hat{T}) = T$, we re-write the bilinear $a(\cdot,\cdot)$ from equation (15) as follows

$$\begin{aligned} a(u,v) &= \int_\Omega *_\alpha(du)\wedge dv && (16)\\ &= \sum_{T\in T_h}\int_{T=\Phi(\hat{T})} *_\alpha(du)\wedge dv && (17)\\ &= \sum_{T\in T_h}\int_{\hat{T}} \Phi^*\left(*_\alpha(du)\wedge dv\right)|\Phi| && (18)\\ &= \sum_{T\in T_h}\int_{\hat{T}} *_{\alpha\cdot\Phi}\left(\Phi^*(du)\right)\wedge\Phi^*(dv)|\Phi|. && (19) \end{aligned}$$

Similarly the mass matrix can be obtained using the following bilinear form

$$b(u,v) \;=\; \int_\Omega *(u)\wedge v, \tag{20}$$

and after some manipulations, we have

$$b(u,v) \;=\; \sum_{T\in T_h}\int_{\hat{T}} *\left(\Phi^*(u)\right)\wedge\Phi^*(v)|\Phi|. \tag{21}$$

Equations (19) and (21) show that all calculations for the mass and stiffness matrices are performed on a standard reference element $\hat{T}$ (i.e. the unit cube, tetrahedron, or prism). Results are then transformed to physical mesh elements (of arbitrary curvature) via a set of well defined transformation rules based on the properties of differential forms. These rules are summarized in Table 5 where u denotes a function defined with respect to the global coordinate system and $\hat{u}$ is the same function defined with respect to the local (or reference) coordinate system. Given these transformations the bases need only be evaluated on the reference element and transformed accordingly. This gives rise to a very computationally efficient algorithm for computing finite element approximations. For a given element topology and basis order, the basis functions only need to be computed once. Then, for every element of the same topology in the mesh, the results from the reference element can simply be mapped according to the transformation rules. This can significantly reduce computational time for a typical finite element computation. In addition, integration over the reference element is much simpler and can be done exactly using Gaussian quadrature of the appropriate order.

In addition, several levels of efficiency have been added in the implementation of this class. The local mass and stiffness matrices are symmetric therefore only one triangular block is actually computed and the rest of the

Table 5 : Transformation rules Φ^*

	$\Phi^*(u)$	$\Phi^*(du)$
0-forms	$\hat{u}$	$\partial\Phi^{-1}(d\hat{u})$
1-forms	$\partial\Phi^{-1}\hat{u}$	$\frac{1}{\lvert\partial\Phi\rvert}\partial\Phi^T(d\hat{u})$
2-forms	$\frac{1}{\lvert\partial\Phi\rvert}\partial\Phi^T\hat{u}$	$\frac{1}{\lvert\partial\Phi\rvert}(d\hat{u})$

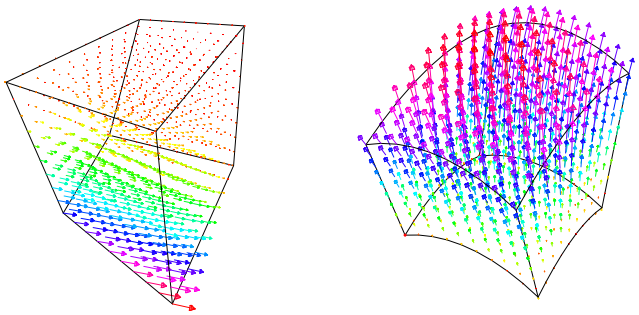

Figure 5 : Visual examples of a 1-form transformation (*left*) and a 2-form transformation (*right*)

entries are copied. For tetrahedrons of order 1, the Jacobian is constant so there is no need to compute it at each integration point.

Below, in Table 6, we show the common interface of a bilinear p-form.

Table 6 : Interface of the Bilinear-pForm class

Method	Description
setpForm()	set a specific 0-form
setIntRule()	set the integration rule
setElement()	set a specific element3D
initialize()	initialize internal data
getMassMatrix()	get the local mass matrix
getStiffnessMatrix()	get the local stiffness matrix
getLoadVector()	get local load vector
getUError()	get the local error
getQError()	get local error of derivative

The getUError() and getQError() methods are provided as testing tools. These can be used to compute the accuracy of the finite element approximation whenever the exact solution of the problem is known.

4 Simulations

4.1 Poisson Equation

The Poisson equation corresponds to the case $p = 1$ in (1)-(4). Here u is a 0-form potential-like quantity and j is a 2-form flux-like quantity. The operator $\star_\alpha$ can be interpreted as the dielectric constant in the case of electrostatics, permeability in the case of magnetostatics, thermal conductivity in the case of heat transfer, etc. The boundary conditions on Γ_D, Γ_N, and Γ_M correspond to the standard Dirichlet, Neumann, and Robbins boundary conditions, respectively.

$$\begin{aligned} -\nabla\cdot(\nabla\phi) &= f \quad \text{in } \Omega, \\ \phi &= s \quad \text{on } \partial\Omega_D, \\ \nabla\phi\cdot\hat{n} &= n \quad \text{on } \partial\Omega_N. \end{aligned} \tag{22}$$

In this numerical example we solve the above problem on a cubic domain using hexahedral elements subject to the Dirichlet boundary condition. We choose an exact solution

$$\phi = \cos(x)\sin(y)\exp(z) \tag{23}$$

and insert this into (22) to form the corresponding source function f. We use Bilinear0Form methods getStiffnessMatrix() and getLoadVector() to form the local stiffness matrix and the local load vector for every element in the mesh. Given these local matrices and local vectors, the standard finite element procedure is used to assemble a global system of the form

$$\mathbf{Ax} = \mathbf{b}, \tag{24}$$

where $\mathbf{A}$ is the global stiffness matrix, $\mathbf{b}$ is the global load vector, and $\mathbf{x}$ is the unknown vector of finite element coefficients. The linear system is solved via a conjugate gradient algorithm.

In Figure 6 we show the computed L_2 error versus element size h on a $\log-\log$ scale for 0-form basis functions of degree 1 through 4. The slopes of the lines (based on a least-squares fit of the data points) are $(2.0000, 2.9939, 3.9914, 4.9692)$ indicating the optimal convergence rate of $k+1$.

4.2 Vector Helmholtz

The vector Helmholtz equation corresponds to the case $p = 2$ in (1)-(4) with the exception that we add a $-\omega^2$

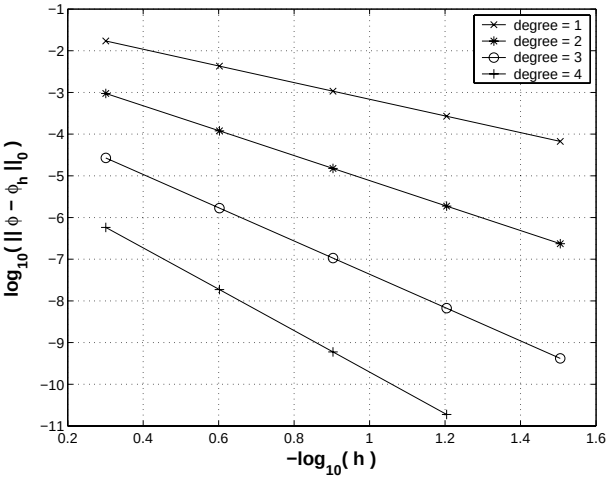

Figure 6 : Polynomial convergence of *h*-refined solutions of the Poisson equation using finite element 0-form basis functions of degree 1 through 4.

term so that the problem corresponds to a hyperbolic wave equation. Here both u and j are 1-form field-like quantities and σ, ψ, and Φ are 2-form flux-like quantities. In the case of Maxwell's equations u and j are the electric and magnetic fields, Ψ and σ are the electric and magnetic flux densities, the source term Φ is the current density, and the star operators $\star_\alpha$ and $\star_\gamma$ correspond to $1/\mu$ and ε, respectively. The boundary conditions on Γ_D and Γ_N correspond to $\hat{n}\times u = 0$ and $\hat{n}\times\nabla\times u = 0$, and the boundary condition on Γ_M is an impedance boundary condition. In standard form we have

$$\nabla\times\frac{1}{\mu}\nabla\times\vec{E} - \varepsilon\omega^2\vec{E} = \vec{f} \tag{25}$$

where f is a source term consisting of electric or magnetic currents, and $\vec{E}$ is the time-harmonic complex-valued electric field. In practice we use a prescribed voltage boundary condition $\hat{n}\times\vec{E} = v(t)$ or a radiation boundary condition $\hat{n}\times\nabla\times\vec{E} = P(E)$ where $P(E)$ is chosen to approximate the Sommerfeld radiation boundary condition.

In this computational experiment we validate the expected rates of convergence for *h*-refinement by choosing a simple problem with a known, smooth solution. The computational domain is a unit cube, discretized via a series of unstructured tetrahedral meshes. We choose an exact solution

$$\vec{E} = \left(0, 0, \left(x-x^2\right)^2\left(y-y^2\right)^2\left(z-z^2\right)^2\right) \tag{26}$$

and insert this into (25) to form the corresponding source function $\vec{f}$. We use Bilinear1Form methods getStiffnessMatrix(), getMassMatrix(), and getLoadVector() to form the local matrices and the local load vector for every element in the mesh. Given these local matrices and local vectors, the standard finite element procedure is used to assemble a global system of the form

$$\left(\mathbf{A} - \omega^2\mathbf{B}\right)\mathbf{x} = \mathbf{b} \tag{27}$$

where **A** is the global stiffness matrix, **B** is the global mass matrix, **b** is the global load vector, and **x** is the unknown vector of finite element coefficients. The linear system is solved via an ILU preconditioned GMRES algorithm.

In Figure 7 we show the computed L_2 error versus element size h on a $\log-\log$ scale for 1-form basis functions of degree 1 through 6. The slopes of the lines (based on least-squares fit of the last three data points) are $(0.98, 1.97, 2.97, 3.97, 4.97, 5.98)$ indicating the optimal convergence. It is interesting to note that for this particular problem using a 6th order basis on a 1440 element mesh yields a solution accurate to 10 significant digits, where a comparable solution using a 1st order basis would require a mesh consisting of billions of elements. Naturally, we cannot expect this type of accuracy for problems with re-entrant corners and associated field singularities, but high-order approximation can be combined with adaptive *h*-refinement for such problems.

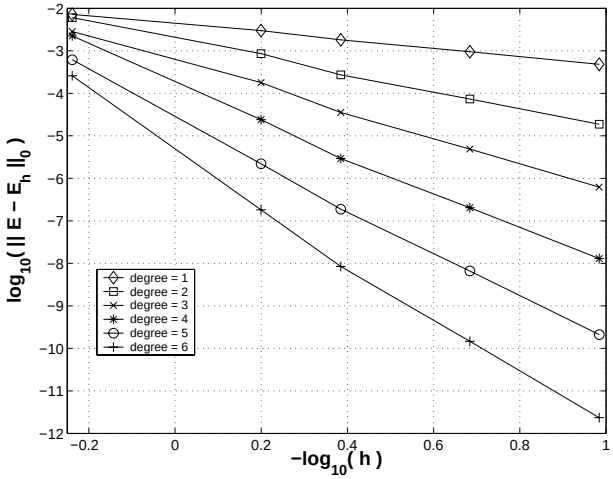

Figure 7 : Polynomial convergence of *h*-refined solutions of the vector Helmholtz equation using finite element 1-Form basis functions of degree 1 through 6.

4.3 Acoustic Eigenvalues

The acoustic problem corresponds to case $p = 3$ (1)-(4) with the exception that the source term Φ is zero and we are seeking the fundamental eigensolutions of the operator. Here u is a 2-form flux-like quantity and j is a 0-form potential-like quantity. In the case of acoustics the operator $\star_\alpha$ is density, in elasticity it corresponds to a combination of the Lamé constants; $\lambda + 2\mu$. The boundary condition on Γ_D is a zero-flux condition, while the condition on Γ_N is a zero-pressure condition. In standard notation we have

$$\begin{aligned} -\nabla\left(\nabla\cdot\vec{U}\right) &= \omega^2\vec{U} \quad in\ \Omega \\ \vec{U}\cdot\hat{n} &= 0 \quad on\ \partial\Omega_D \\ \nabla\cdot\vec{U} &= 0 \quad on\ \partial\Omega_N \end{aligned} \tag{28}$$

where $\vec{U}$ is the velocity and ω is the resonant frequency.

In this computational example we compute the eigenvalues of the above equation on a fixed mesh for various values of polynomial degree k. We choose a problem in which the eigenmodes are known to be smooth, and thus we achieve the expected exponential convergence. The computational domain is a unit cube with the exact eigenvalues given by

$$\omega^2 = \pi^2\left(l^2 + m^2 + n^2\right) \tag{29}$$

with $l, m, n \neq 0$. The domain is discretized using a 6 element tetrahedral mesh, and equation (28) is discretized using 2-form basis functions, with the required discrete bilinear forms computed by the Bilinear2Form methods getMassMatrix() and getStiffnessMatrix(). This results in a generalized linear eigenvalue problem

$$\mathbf{A}\mathbf{x} = {\omega_h}^2\mathbf{B}\mathbf{x} \tag{30}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the global 2-form stiffness and mass matrices, respectively. The vector $\mathbf{x}$ represents the unknown coefficients of the basis function expansion of the eigenmode $\vec{U}$, and ω_h is the computed resonant frequency of the eigenmode.

In this example we use Matlab to compute the entire set of eigenvalues of (30). The model equation (28) has an infinite set of zero-valued eigenvalues, corresponding to solenoidal solutions. The discrete spectrum therefore has a large number of zero-valued (to machine precision) eigenvalues. While this is evidence that our discretization correctly models the kernel of the grad-div operator, these eigenvalues are of no interest to us, so we search the computed spectrum for the first non-zero eigenvalue which according to (29) should have the value $3\pi^2$. In Figure 8 we plot the log of the error $|\omega - \omega_h|$ of the first non-zero eigenvalue versus k, the degree of the finite element approximation. We see the expected exponential convergence. The plateaus in the convergence are due to the symmetry of the fundamental mode. For very large problems in which it is not feasible to use Matlab, it is possible to develop iterative eigenvalue solvers that quickly converge to the smallest non-zero eigenvalues White and Koning (2002).

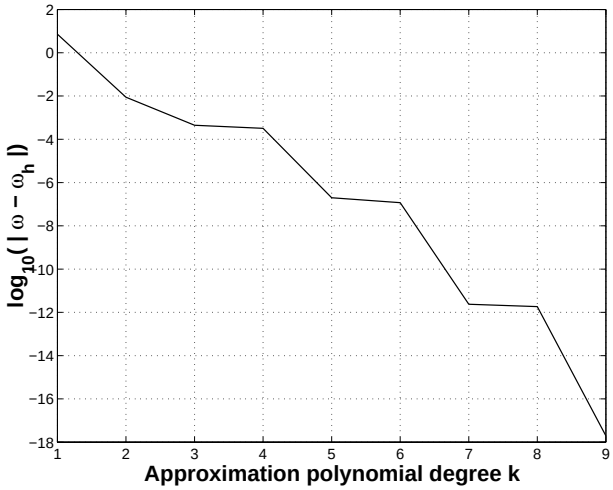

Figure 8 : Exponential p-convergence of the 2-form discretization of the acoustic equation.

4.4 Maxwell's Equations

The main use of the FEMSTER framework is time-domain computational electromagnetics. Typically the second order wave equation for the electric field is solved:

$$\varepsilon\frac{\partial^2}{\partial t^2}\vec{E} = -\nabla\times\mu^{-1}\nabla\times\vec{E} - \frac{\partial}{\partial t}\vec{J} - \sigma\frac{\partial}{\partial t}\vec{E} \ \text{ in } \Omega \tag{31}$$

where ε and μ are the tensor electric permittivity and magnetic permeability respectively, and σ is the electrical conductivity. The boundary conditions on $\vec{E}$ are typically a combination of perfect conducting, impedance, or radiation boundary conditions. The discretized wave equation is

$$\mathbf{B}\ddot{\mathbf{x}} = -\mathbf{A}\mathbf{x} - \mathbf{C}\dot{\mathbf{x}} - \dot{\mathbf{y}} \tag{32}$$

where **A** is the stiffness matrix and **B** and **C** are mass matrices involving the permittivity and conductivity, respectively. These matrices are similar to those used in the frequency domain Helmholtz equation in Section 4.2. In this paper, equation (32) is is discretized using the second-order accurate leap-frog method, the stability and conservation properties are discussed in Rodrigue and White (2001). An alternative higher order time discretization designed specifically for wave equations is described in Rieben, White, and Rodrigue (2004a) The

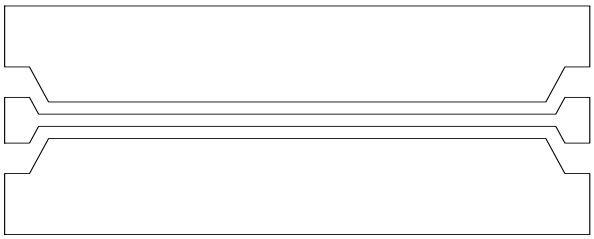

Figure 9 : Coplanar waveguide geometry (not to scale).

first computational example is of a three conductor coplanar waveguide which consists of a 2 micrometer thick metal deposited on silicon. The outer conductors can be considered ground, and the inner conductor is the signal conductor. For high frequency signals, the coplanar waveguide configuration is superior to the traditional micro-strip, as the fields are confined to the small region between the conductors. The domain is discretized using prism element mesh in which the x-y planes are triangle meshes extruded in the z-direction to form the prisms. Figure 9 shows the x-y plane containing the metal. We excite the problem with a time-varying voltage source at the left end of the waveguide. We are interested in how the voltage pulse travels down the guide. We run the simulation for 6000 time steps. The induced currents on the conductors can be measured, and this data can be post-processed to yield input impedance, S-parameters, and other useful characteristics of the waveguide. The voltage input and output for the transmission line is shown in Figure 10.

As a second example, we illustrate the computed electromagnetic fields in a bent single mode optical fiber. A 3-dimensional hexahedral mesh is used for the simulation, with approximately 1.8 million elements with 5.5 million degrees of freedom. Due to the very fine mesh used to resolve the fiber core, 1st order basis functions

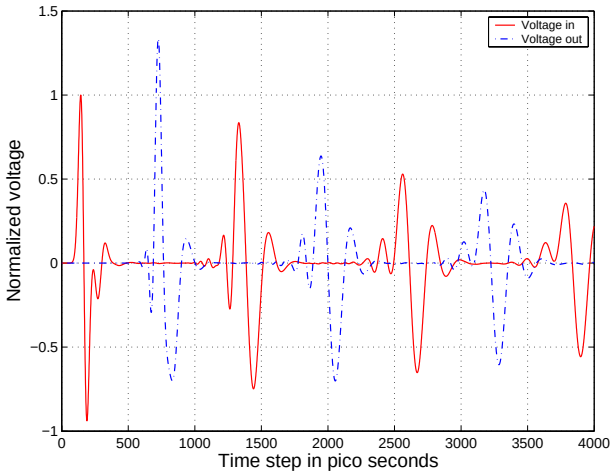

Figure 10 : Input and output voltage for the coplanar waveguide. The solid line is the voltage at the input port, the dashed line is the voltage at the output port. Note the reflections due to the imperfectly matched loads.

are used. A TE01 wave pulse is launched at one end of the 5 micrometer radius fiber core, and the other end is terminated with a Maxwellian perfectly matched layer. The fiber is designed to support a 1.55 micrometer wavelength mode and is 52 micrometers long with a bend radius of 55.13 micrometers. The purpose of the simulation is to determine how much, if any, of the electric field escapes the fiber due to the sharp bend. The simulation is run for 17,000 time steps with $dt = 3.3e-16$ s. This is a full-wave, explicit time-domain simulation. The electric field intensity for a straight and bent optical fiber are shown in Figures 11 and 12.

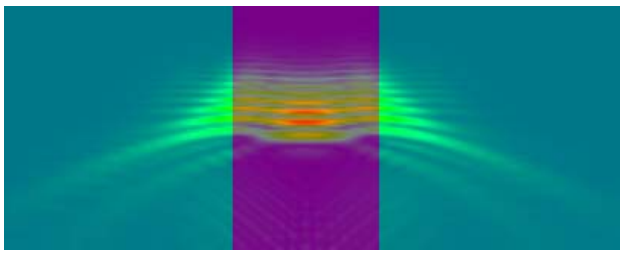

Figure 11 : Slice of three-dimensional straight optical fiber electric field magnitude. This is a snapshot of the field half way through the simulation.

As a final example, we compute the wake fields in a generic accelerator. The accelerator consists of a series of induction cells approximately one meter in diameter. An electron bunch travels down the center of the accelerator,

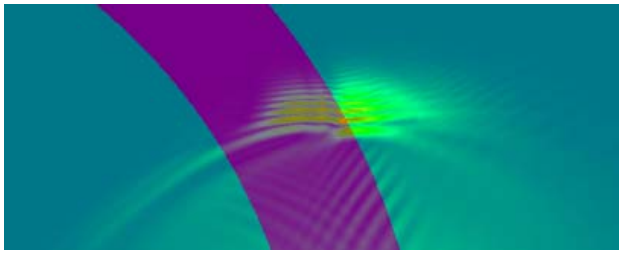

Figure 12 : Slice of three-dimensional bent optical fiber electric field magnitude. This is a snapshot of the field half way through the simulation. Note the electric field leakage into the cladding.

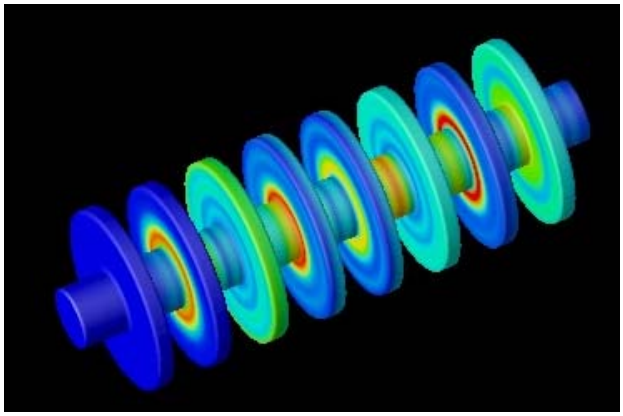

Figure 13 : Snapshot of the magnetic current induced on the walls of a linear accelerator due to a Gaussian electron bunch. At this instant the bunch, traveling from right to left, is approximately 90 percent of the way down the accelerator. Note that the induction cells resonate for a long time after the electron bunch has passed by.

receiving a 1 MEV boost as it passes by each induction cell. The traveling electron bunch generates electromagnetic waves which may resonate within the induction cell for a very long time. These waves are referred to as the wake field. This wake field can interfere with the next electron bunch causing a beam instability. In this simulation we model the electron bunch as a rigid Gaussian beam, i.e. we are simply interested in the wake field and not in the precise motion of the electrons. The accelerator is modeled by a 3-dimensional hexahedral mesh with approximately 500,000 elements, with perfectly conducting walls and a Maxwellian perfectly matched layer at each end. In this simulation we solve for both the electric and magnetic fields in a leapfrog manner similar to the FDTD method, except that we solve a finite element mass matrix at every time step. We do this simply because the magnetic field is the more intuitive field to visualize for this problem. The magnetic current is illustrated in Figure 13.

5 Concluding remarks

In this article we review a software class library of high-order finite element basis functions that is based on the language of differential forms. In our library the basis functions are specified by the form p, by the element type which at present is either a tetrahedron, hexahedron, or prism, and by the order k of the polynomial space. For each p-form we have a separate class hierarchy in which the abstract base class defines the common interface for the derived classes. We have implemented a variety of different degrees of freedom including uniformly spaced interpolation, non-uniform interpolation, and hierarchical moment-based degrees of freedom. The classes were designed to be extensible, users can easily experiment with alternative degrees of freedom by deriving a new class and implementing a single new method.

The language of differential forms is well suited for expressing the laws of electromagnetism. We designed the application program interface of our library to mimic the language of differential forms. This way, given a physical law expressed in differential forms, it is a straightforward process to correctly discretize and solve the problem. The discrete differential forms approach preserves the important symmetry, conservation, and spectral properties of the original continuous operators. We demonstrate how our approach can be applied to electrostatics, eigenvalue problems, the frequency domain Helmholtz equations, and the time-dependent Maxwell equations. The frequency domain solutions are free of spurious modes. The time domain solutions are stable, charge conserving, and energy conserving.

Current research issues in the area of discrete differential forms include fast linear solvers that take advantage of the unique properties of p-forms Hiptmair (1998), Schinnerl, Schoberl, and Kaltenbacher (2000), adaptive hp-refinement Rachowicz and Demkowicz (2002), and applications to nonlinear and multi-physics problems.

References

Abraham, R.; Marsden, J. E.; Ratiu, T. (1996): *Manifolds, Tensor Analysis, and Applications.* Applied Math-

ematical Sciences.

Babuska, I.; Ihlenburg, F.; Strouboulis, T.; Gangaraj, S. K. (1997): A posteriori error estimation for finite element solution of Helmholtz. *Internat. J. Numer. Methods Engrg.*, vol. 40, no. 21, pp. 3883–3900.

Babuska, I.; Strouboulis, T.; Upadhyay, C.; Gangaraj, S. K. (1995): A posteriori estimation and adaptive control of the pollution error in the *h*-version of the finite element method for finite element solution of Helmholtz. *Internat. J. Numer. Methods Engrg.*, vol. 38, no. 24, pp. 4207–4235.

Baldomir, D. (1986): Differential forms and electromagnetism in 3-dimensional Euclidean space R3. *IEEE Proceedings.*, vol. 133, no. 3, pp. 139–143.

Bossavit, A. (1998): *Computational Electromagnetism : variational formulation, complementarity, edge elements.* Academic Press.

Burke, W. (1985): *Applied Differential Geometry : variational formulation.* Cambridge University Press.

Ciarlet, P. G. (1978): *The Finite Element Method for Elliptic Problems*. North-Holland.

Deschamps, G. (1981): Electromagnetics and differential forms. *IEEE Proceedings.*, vol. 69, no. 6, pp. 676–687.

Graglia, R.; Wilton, P.; Peterson, A. (1997): Higher order interpolatory vector bases for computational electromagnetics. *IEEE Trans. Ant. Prop.*, vol. 45, no. 3, pp. 329–342.

Graglia, R.; Wilton, P.; Peterson, A.; Gheorma, I.-L. (1998): Higher order interpolatory vector bases on prism elements. *IEEE Trans. Ant. Prop.*, vol. 46, no. 3, pp. 442–450.

Hesthaven, J. S.; Warburton, T. (2004): High-order accurate methods for time-domain electromagnetics. *CMES: Computer Modeling in Engineering & Sciences*, vol. 5, no. 5, pp. 395-408.

Hiptmair, R. (1998): Multigrid method for 'Maxwell's equations. *SIAM J. Num. Anal.*, vol. 38, no. 1, pp. 204–225.

Hiptmair, R. (1999): Canonical construction of finite elements. *Math. Comp.*, vol. 68, no. 228, pp. 1325–1346.

Hiptmair, R. (2001): Discrete Hodge operators: An algebraic perspective. *J. Electromagnteic Waves Appl.*, vol. 15, no. 3, pp. 343–344.

Jin, J. (1993): *The Finite Element Method in Electromagnetics*. Wiley.

Namburu, R. R.; Mark, E. R.; Clarke, J. A. (2004): Scalable electromagnetic simulation environment. *CMES: Computer Modeling in Engineering & Sciences*, vol. 5, no. 5, pp. 443-454.

Nédélec, J. C. (1980): Mixed Finite Elements in r3. *Numer. Math.*, vol. 35, pp. 315–341.

Nédélec, J. C. (1986): A New Family of Mixed Finite Elements in r3. *Numer. Math.*, vol. 50, pp. 57–81.

Rachowicz, W.; Demkowicz, L. (2002): An hp-adaptive finite element method for electromagnetics. vol. 53, no. 1, pp. 147–180.

Raviart, P.; Thomas, J. (1977): A Mixed Finite Element Method for 2^{nd} Order Elliptic Problems. In Galligani, I.; Mayera, E.(Eds): *Mathematical Aspects of the Finite Element Method*, volume 606 of *Lect. Notes. on Mathematics*, pp. 293–315. Springer Verlag.

Rieben, R.; White, D.; Rodrigue, G. (2004): High order symplectic integration methods for finite element solutions to time dependent maxwell equations. *IEEE Trans. Ant. Prop.* in press.

Rieben, R.; White, D.; Rodrigue, G. (2004): Improved conditioning of finite element matrices using new high order interpolatory bases. *IEEE Trans. Ant. Prop.* in press.

Rodrigue, G.; White, D. (2001): A vector finite element time-domain method for solving Maxwell's equations on unstructured hexahedral grids. *SIAM J. Sci. Comp.*, vol. 23, no. 3, pp. 683–706.

Schinnerl, S.; Schoberl, M.; Kaltenbacher, M. (2000): Nested multigrid methods for the fast numerical computation of 3d fields. *IEEE Trans. Mag.*, vol. 36, no. 4, pp. 1557–1560.

Stroustrup, B. (1991): *C++ Programming Language*. Addison-Wesley, Reading, MA.

Warren, S.; Scott, W. (1994): An investigation of numerical dispersion in the vector finite element method using quadrilateral elements. *IEEE Trans. Ant. Prop.*, vol. 42, no. 11, pp. 1502–1508.

Warren, S.; Scott, W. (1995): Numerical dispersion in the finite element method using triangular edge elements. *Opt. Tech. Lett.*, vol. 9, no. 6, pp. 315–319.

White, D. (2000): Numerical dispersion of a vector finite element method on skewed hexahedral grids. *Commun. Numer. Meth. Engng.*, vol. 16, pp. 47–55.

White, D.; Koning, J. (2002): A novel approach for computing solenoidal eigenmodes of the vector Helmholtz equation. *IEEE Trans. Mag.*, vol. 38, no. 5, pp. 3420–3425.