CAMES, **28**(4): 291–319, 2021, doi: 10.24423/cames.384

A Novel Conversion Technique from Nodal to Edge Finite Element Data Structure for Electromagnetic Analysis

Durgarao KAMIREDDY*, Arup NANDY

Department of Mechanical Engineering
Indian Institute of Technology Guwahati
Guwahati, India; e-mail: arupn@iitg.ac.in
*Corresponding Author e-mail: durga176103010@iitg.ac.in

Standard nodal finite elements in the electromagnetic analysis have a well-known limitation of the occurrence of a spurious solution. In order to circumvent the problem, a penalty function method or a regularization method is used with the potential formulation. These methods solve the problem partially by pushing the spurious mode to the higher end of the spectrum. But it fails to capture singular eigenvalues in the case of the problem domains with sharp edges and corners. To circumvent this limitation, edge elements have been developed for the electromagnetic analysis where degree of freedom is along the edges. But most of the preprocessors develop complex meshes in the nodal framework. In this work, we have developed a novel technique to convert nodal data structure to edge data structure for electromagnetic analysis. We have explained the conversion algorithm in details, mentioning associated complexities with relevant examples. The performance of the developed algorithm has been demonstrated extensively with several examples.

Keywords: FEM, electromagnetics, edge finite elements, eigenvalue analysis.

1. Introduction

The finite element method (FEM) has been widely used for radiation and scattering problems in interior and exterior domains and it has extensive applications in antenna radiations, waveguide transmissions, etc. In order to apply the FEM technique, the domain can be discretized with either edge element or nodal element. The problem of the occurrence of a spurious solution is a well-known limitation of standard nodal finite elements in the electromagnetic analysis. In order to circumvent the problem, the penalty function method and regularization method [22, 28, 29] have been used extensively in the nodal FEM framework. These methods solve the problem partially by pushing the spurious mode to the higher end of the spectrum. In order to capture singular eigenvalue in the case of sharp edges and corners, in the regularization method, we have to use a penalty parameter varying from 0 at the sharp edge to 1 at a large distance from the sharp edge. Also, in order to take care of inherent tangential continuity and normal discontinuity across the material interface, the potential formulation is used in the nodal finite element framework [1, 4, 24–26]. In [19], a two-field variation formulation in electromagnetics can predict the eigenfrequencies very accurately with correct multiplicities. There is no ad hoc term in the mixed formulation as in the penalty function or the regularization method. This method worked very well for all two-dimensional geometries such as non-convex domains with sharp corners, in-homogeneous domains, curved domains, etc. In three dimension, mixed FEM worked for plane structures (structures without any curvature) quite well; there, it worked flawlessly with sharp edges and in-homogeneous domains. But in the case of curved three-dimensional geometries, this mixed formulation failed.

Edge elements were introduced by Whitney, which are also called curl-conforming elements. Nedelec presented the conceptual theory of edge elements [27]. He presented a nonconforming tetrahedron and cube finite elements construction [27] conforming to the H curl and H div spaces. Whitney spaces can act as bases for edge elements in FEM for field type of problems [9] and eddy current problems. In [3, 6, 7, 11, 12, 15, 20, 21] edge elements are used to solve the eigenvalue problems with different shapes. In [10], Bossavit and Vérité solved the 3-D eddy current problems using the combination of FEM and boundary integral element method (BIEM) methods. Edge elements can be easily applied to an exterior domain problem where the coupling of other methods like absorbing boundary condition (ABC) [37], boundary integral (BI) [34], perfectly matched layer (PML) [30] is required. Cendes [12] presented the implementation of the tangential vector finite element method to analyze the dielectric waveguides. In the literature, various methods such as method of moment, spectral-domain methods, finite difference and finite element are adopted for the analysis of dielectric waveguide problems.

Webb [35] broadly discussed the useful properties of edge elements. Vector finite element or edge element [7, 8, 12, 27, 32] was proposed to circumvent the spurious solution problem of nodal FEM. Electromagnetic radiation and scattering problems require special elements where normal discontinuity and tangential continuity exists across material interfaces are met by edge elements. In [5], Barton and Cendes showed that continuity of tangential components of the vector field is sufficient in vector-based FEM to compute the magnetic fields. Another advantage of edge elements is that electric or magnetic fields can be directly computed without any differentiation on potentials. Also, no penalty or regularization term is required in edge element framework. While modeling sharp, perfectly conducting objects, the electric field has to be infinite inside the domain and its direction changes rapidly at the sharp edges and corners. In order to do eigenanalysis of such geometries in nodal framework, singular trial functions are required, whereas due to tangential continuity of edge elements, no such function is required.

Yioultsis and Tsiboukis [39, 40] presented the systematic approach to construct the higher order tetrahedral edge elements and used them to solve the waveguide problems with material discontinuity. General expressions for the shape functions were presented and unknown coefficients were found by following the decoupling procedure. In [31], the formation of higher-order Whitney p-elements was shown. In [23], the construction of higher-order two-dimensional and three-dimensional H1 curl elements are presented to solve the electromagnetic scattering problems. In [16], Graglia *et al.* presented the general approach of interpolatory vector basis functions of various two-dimensional and three-dimensional elements. Edge elements can be constructed by using hierarchical vector basis functions also. Webb [36] proposed hierarchical vector basis functions for higher-order triangle and tetrahedral finite elements. In [33], Seung-Cheol Lee *et al.* implemented higher-order hierarchical vector finite elements in the field of microwave engineering to the wave guiding structures. In hierarchical type implementation, there can be p refinement in some parts of the domain and in some part we can have h refinement. But, in the interpolatory type we can only have one type of refinement in the entire domain. To the best knowledge of the authors, in all the above literature, the detailed conversion strategy from nodal FE input file to edge element data-structure is not available. As most of the available mesh generator packages are based on nodal FEM, it will be very useful if such a conversion algorithm is developed. In the current work, we present a systematic and thorough conversion algorithm.

The rest of this article is organized as follows: in Sec. 2, we present the algorithm in minute details with associated flowcharts and simple conversion examples for different elements such as four-edge quadrilateral, three-edge triangle, twelve-edge quadrilateral and eight-edge triangle. In Sec. 3, we validate our conversion algorithm with several benchmark numerical examples, including all possible complexities such as curved surfaces, sharp edges and corners. We compare our results with available analytical and benchmark solutions from the literature.

2. Conversion algorithm for creating edge data structure from nodal data structure

For electromagnetic analysis in the edge element framework, element data is required in the form of edge data structure. To achieve this, a standalone conversion algorithm is needed where the edge information is generated as output from the supplied nodal data as input. Here, every edge is generated by joining the two nodes of the element. In this section, we are presenting such a conversion algorithm to different edge elements. This type of program is necessary because most of the available commercial mesh generators create and generate nodal element data structure.

2.1. Calculation of $\frac{\partial \xi}{\partial x}$, $\frac{\partial \xi}{\partial y}$, $\frac{\partial \eta}{\partial x}$ and $\frac{\partial \eta}{\partial y}$

The edge shape functions contains the terms of an inverse Jacobian ($\mathbf{\Gamma}$) and its components $\frac{\partial \xi}{\partial x}$, $\frac{\partial \xi}{\partial y}$, $\frac{\partial \eta}{\partial x}$ and $\frac{\partial \eta}{\partial y}$, which can be obtained as discussed below. From the relation:

$$\begin{Bmatrix} \dfrac{\partial f}{\partial \xi} \\ \dfrac{\partial f}{\partial \eta} \end{Bmatrix} = \mathbf{J} \begin{Bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{Bmatrix} \Longrightarrow \begin{Bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{Bmatrix} = \mathbf{J}^{-1} \begin{Bmatrix} \dfrac{\partial f}{\partial \xi} \\ \dfrac{\partial f}{\partial \eta} \end{Bmatrix},$$

where $\mathbf{J}$ is Jacobian can be written as:

$$\mathbf{J} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} \text{ and assume } \mathbf{J}^{-1} = \mathbf{\Gamma} = \begin{bmatrix} \mathit{\Gamma}_{11} & \mathit{\Gamma}_{12} \\ \mathit{\Gamma}_{21} & \mathit{\Gamma}_{22} \end{bmatrix};$$

$$\text{therefore, } \begin{Bmatrix} \dfrac{\partial \xi}{\partial x} \\ \dfrac{\partial \xi}{\partial y} \end{Bmatrix} = \begin{Bmatrix} \mathit{\Gamma}_{11} \\ \mathit{\Gamma}_{21} \end{Bmatrix} \text{ and } \begin{Bmatrix} \dfrac{\partial \eta}{\partial x} \\ \dfrac{\partial \eta}{\partial y} \end{Bmatrix} = \begin{Bmatrix} \mathit{\Gamma}_{12} \\ \mathit{\Gamma}_{22} \end{Bmatrix}.$$

2.2. Four-edge quadrilateral element

For four-node quadrilateral element Fig. 1a shows the local nodal connectivity for which Fig. 1b shows the required local edge connectivity sequence. Edge $\mathbf{e}_1$ is formed by connecting the local node set $(\mathbf{1}, \mathbf{2})$. Similarly, $\mathbf{e}_2$, $\mathbf{e}_3$ and $\mathbf{e}_4$ are

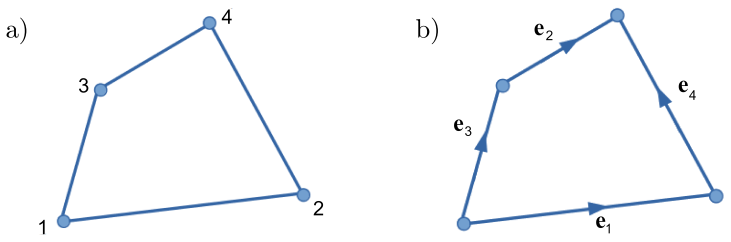


Fig. 1. Quadrilateral element with: a) four nodes, b) four edges.

the other three edges generated by connecting the node sets $\mathbf{(4, 3)}$, $\mathbf{(1, 4)}$ and $\mathbf{(2, 3)}$, respectively. The edge shape functions of four edges are $\mathbf{v}_1 = \frac{l_1}{4}(1-\eta)\nabla\xi$, $\mathbf{v}_2 = \frac{l_2}{4}(1+\eta)\nabla\xi$, $\mathbf{v}_3 = \frac{l_3}{4}(1-\xi)\nabla\eta$ and $\mathbf{v}_4 = \frac{l_4}{4}(1+\xi)\nabla\eta$ [18]. Here l_1, l_2, l_3 and l_4 are the lengths of the edges $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, and $\mathbf{e}_4$, respectively.

2.2.1. Calculation of $\nabla\times$ **E** *with edge element.* The components of $\nabla \times \mathbf{E}$ can be calculated by using the relation:

$$\nabla\times\mathbf{E} = \left[\frac{\partial v_{1y}}{\partial x} - \frac{\partial v_{1x}}{\partial y} \quad \frac{\partial v_{2y}}{\partial x} - \frac{\partial v_{2x}}{\partial y} \quad \frac{\partial v_{3y}}{\partial x} - \frac{\partial v_{3x}}{\partial y} \quad \frac{\partial v_{4y}}{\partial x} - \frac{\partial v_{4x}}{\partial y}\right] \begin{Bmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{Bmatrix} = \mathbf{BE},$$

where E_1, E_2, E_3 and E_4 are the tangential electric fields along the edges 1, 2, 3 and 4, respectively. The components of the **B**-matrix are obtained as below:

$$\begin{bmatrix} \frac{\partial v_{1x}}{\partial x} & \frac{\partial v_{1x}}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial v_{1x}}{\partial \xi} & \frac{\partial v_{1x}}{\partial \eta} \end{bmatrix} \begin{bmatrix} \mathit{\Gamma}_{11} & \mathit{\Gamma}_{21} \\ \mathit{\Gamma}_{12} & \mathit{\Gamma}_{22} \end{bmatrix}.$$

We can derive partial derivative terms $\frac{\partial v_{1x}}{\partial \xi}$, $\frac{\partial v_{1y}}{\partial \eta}$, ..., etc., with finite difference or Mathematica software [38].

2.2.2. Conversion algorithm: Generation of edge connectivity array. To describe the conversion algorithm, we consider a domain is meshed with quadrilateral elements, as shown in Fig. 2a. Figure 2b shows the global nodal connectivity of the meshed domain. Nodal connectivity list for all the elements for the mesh is given in Table 1.

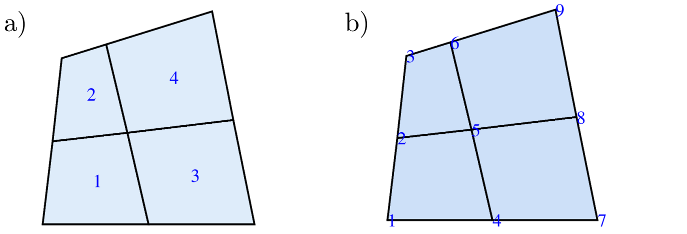


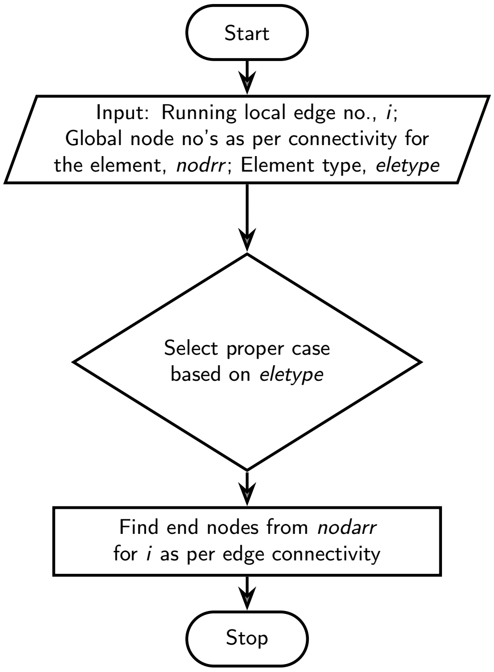
FIG. 2. Discretized domain with: a) global elements, b) global nodes.

In this algorithm, the outermost loop runs over the total number of discretized elements, and the next inner loop runs over the total number of local edges (*nlocedge*) of each element. One global counter e_l is used, which is updated to the last assigned global edge number at the end of each element loop. Inside the

TABLE 1. Element nodal connectivity.

Element number	Nodal connectivity (global node no.)
1	1, 4, 5, 2
2	2, 5, 6, 3
3	4, 7, 8, 5
4	5, 8, 9, 6

loop of *nlocedge*, a subroutine ***edgend***, as shown in Fig. 3, returns global node numbers of two end nodes (*endnd*1 and *endnd*2) of the edge using local nodal connectivity and local edge connectivity information as shown in Fig. 1a and Fig. 1b. We follow the convention that each edge directs from *endnd*1 to *endnd*2.



Start
Input: Running local edge no., *i*; Global node no's as per connectivity for the element, *nodrr*; Element type, *eletype*
Select proper case based on *eletype*
Find end nodes from *nodarr* for *i* as per edge connectivity
Stop

FIG. 3. Flow chart of edgend structure.

For example, two local nodes 4 and 3 (see Fig. 1a) are connected to form the local edge $\mathbf{e}_2$ of the element, as shown in Fig. 1b. These two local node numbers are the position in the nodal connectivity array (*nodarr*) of the element, and this array contains global node numbers, as shown in Table 1. Thus, for example, for second element, global nodes 6 and 3 are two end nodes for edge $\mathbf{e}_2$. Local edges and its corresponding connected nodes of the elements for the discretized domain are tabulated in Table 2.

Table 2. Local edges and its end nodes of elements of discretized domain.

Element number	Local edge number	End nodes (*endnd*1, *endnd*2)
1	1	1, 4
	2	2, 5
	3	1, 2
	4	4, 5
2	1	2, 5
	2	3, 6
	3	2, 3
	4	5, 6
3	1	4, 7
	2	5, 8
	3	4, 5
	4	7, 8
4	1	5, 8
	2	6, 9
	3	5, 6
	4	8, 9

After successful collection of output from the ***edgend*** subroutine, i.e., information about two end nodes of local edge (i), last assigned global edge (e_l) and existing edge connectivity array ($edgearr$) are further supplied into ***edgedata*** structure as shown in Fig. 4. In this data structure, at the end of each iteration (for each local edge), different variables such as *nodeedgenum*, *nodeedge*, *nodeedgexn*, *edgenode*, *edgearr* are updated for every endnode ($nd1$ and $nd2$) of each edge which is discussed below:

1. *nodeedgenum*: Global one-dimensional static array of dimension of maximum number of global nodes (*mx_ nmnode*) in which i-th row stores the number of edges shared by the i-th global node.
2. *nodeedge*: This is two-dimensional array of dimension ($mx\_\ nmnode \times 8$) where 8 columns of i-th row store two informations of 4 connecting edges shared by i-th node. The 1st column stores global edge no. of the 1st connecting edge, second column stores another end node of that connecting edge. The 3rd and 4th column store (edge no., other end node no.) of the second connecting edge. The 5th to 8th columns store a similar set of information for the third and fourth connecting edges. If some node is shared by more than four connecting edges, then additional information from the 5th edge is stored in *nodeedgeexn.* Also this array nodeedge has the direction information of the edge. If the i-th node is the start node

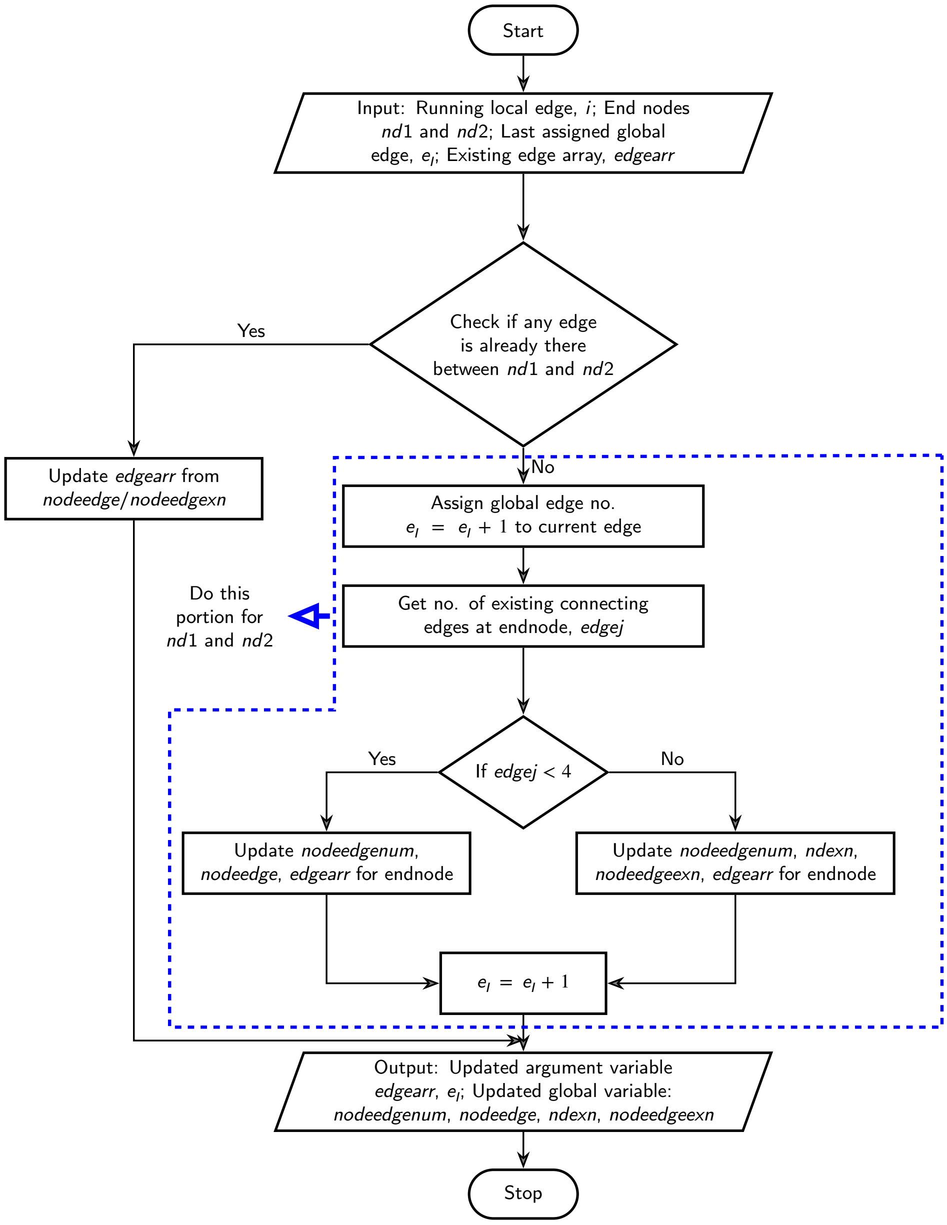

Start
Input: Running local edge, i; End nodes $nd1$ and $nd2$; Last assigned global edge, e_l; Existing edge array, $edgearr$
Check if any edge is already there between $nd1$ and $nd2$
Yes
Update $edgearr$ from $nodeedge/nodeedgexn$
No
Assign global edge no. $e_l = e_l + 1$ to current edge
Do this portion for $nd1$ and $nd2$
Get no. of existing connecting edges at endnode, $edgej$
If $edgej < 4$
Yes
No
Update $nodeedgenum$, $nodeedge$, $edgearr$ for endnode
Update $nodeedgenum$, $ndexn$, $nodeedgeexn$, $edgearr$ for endnode
$e_l = e_l + 1$
Output: Updated argument variable $edgearr$, e_l; Updated global variable: $nodeedgenum$, $nodeedge$, $ndexn$, $nodeedgeexn$
Stop

FIG. 4. Flow chart of edgedata structure.

of the edge, i.e., if the edge is going from the i-th node to another node then edge no. is stored in the odd column as positive integer. If the edge is towards the i-th node from another node, then edge no. is stored as a negative integer. Total no. of negative edges are kept in an account with one counter variable.

3. *nodeedgeexn*: Global two-dimensional array of size (maximum number of nodes in the programme shared by more than 4 edges (mx_nmnnd), $2\times$ maximum additional edges after four edges sharing one node (mx_exedge)). Each row consists of the information about the node having more than four connecting edges. It consists of the information from the fifth edge onwards. Global number of the node whose information are stored in the i-th row of *nodeedgeexn* is stored in the i-th row of *ndexn*, where *nndexn* – no. of nodes associated with more than 4 edges, $ndexn(i)$ – the i-th node number connected with more than 4 edges (according to occurrence), $nodeedgeexn(i,\ 1)$ and $(i,\ 2)$ are 5th edge no. for the node $ndexn(i)$ and another end node of that edge, $(i,\ 3)$ and $(i,\ 4)$ are 6th edge no. for the node $ndexn(i)$ and another end node of that edge and so on.
4. *edgenode*: Global variable of dimension (maximum number of edges in the program, 2) in which the j-th row contains the global number of starting node and ending node of the j-th edge in two columns, respectively.
5. *edgearr*: Local argument variable in the element loop which stores in the process global edge numbers of all the edges of the element according to edge connectivity.

Table 3. Elemental edge connectivity of meshed domain.

Element number	Edge connectivity (global edge no.)
1	1, 2, 3, 4
2	2, 5, 6, 7
3	8, 9, 4, 10
4	9, 11, 7, 12

Table 4. Edgenode array of nodes of the meshed domain.

Global edge(i)	Starting node	End node
1	1	4
2	2	5
3	1	2
4	4	5
5	3	6
6	2	3
7	5	6
8	4	7
9	5	8
10	7	8
11	6	9
12	8	9

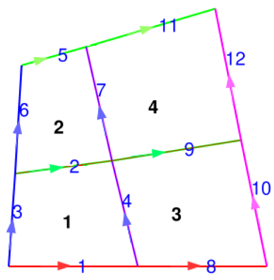


FIG. 5. Element to edge connectivity sequence of the meshed domain.

In ***edgedata*** subroutine, there is a running counter called *edgej* that stores the associated number of edges of the starting end nodes of the current local edge. The information is interchanged between the local *edgej* and the global array *nodeedgenum*. In order to understand the update of the variables, let us first summarize stored values in different variables after the completion of the element loop for the first element.

1. The first, second, fourth and fifth rows of '*nodeedgenum*' array are assigned 'two' because the first element has global nodes 1, 2, 4, and 5 (see Fig. 2b), and after the loop for element 1 each of these nodes is associated with 'two' edges.
2. Last assigned global edge, i.e., e_l with the value 4.
3. *nodeedge* will have the following values as shown in Table 5.
   For the first row (for the global node 1), we have 1 (associated the first edge), 4 (other node of the first edge), 3 (associated the second edge), and 2 (other node of the second edge). The reason for the negative sign in $(2,1)$ position of *nodeedge* is that the first associated edge of node 2, i.e., edge 3 is directing from the other node (1, stored in $(2,2)$) towards the current node 2 (see Fig. 5). For node 1, both the associated edges (1 and 3) are directing from the current node 1 to the respective other nodes. Therefore, those edge numbers are stored with the '+' sign. In the first row (i.e., for the first global edge) of the *edgenode* array, starting node (1) and end node (4) are stored as shown in Fig. 4. Similarly, for the other global edges of the element 1, starting and end nodes are stored in the 2nd, 3rd and 4th rows of this array. Table 4 shows such information for the edges 1 to 4.
4. First row of the edge connectivity array is stored with the edge numbers of the first element, 1, 2, 3 and 4 as shown in the first row of Table 3.

Now for the next entity of the outer element loop, i.e., for element 2, for local edge 1 we have starting and end nodes as 2 and 5 (see Fig. 2b) as per convention of Figs. 1a and 1b. At first, from *nodeedgenum*, we get the total number of already associated edges of the starting node. For our starting node 2, there are

TABLE 5. *Nodeedge* array of nodes after element loop of the first element.

Global node	Associated edge and second node of the edge (*nodeedge* (1:8))							
	1st edge	Other node	2nd edge	Other node	3rd edge	Other node	4th edge	Other node
1	1	4	3	2	0	0	0	0
2	−3	1	2	5	0	0	0	0
3	0	0	0	0	0	0	0	0
4	−1	1	4	5	0	0	0	0
5	−2	2	−4	4	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0

two associated edges. After that, we get the other node numbers of the associated edges from the even columns of row 2 (our current node) of the *nodeedge* array. If any of these other node numbers matches with our current end node (5), then the associated edge no. (available in the respective odd column) will be the global edge no. of that local edge. In this case the edge no. is 2, and we will update the current edge number with global edge number 2. In this local edge loop, we will not update the last assigned global edge (e_l). Also, the first column of *edgearr* is assigned with 2.

For the 2nd local edge, starting and end nodes are 3 and 6, as shown in Fig. 2b. As there is no data available in the 3rd row of the *nodeedge* array, the end node 3 is appearing for the first time. So, a new edge number is assigned just by updating e_l to $e_l + 1$, i.e., with digit 5. Therefore, in the *nodeedge* array, the first and second columns of the third (current end node) row are assigned with 5 (associated edge no.) and 6 (other end node), respectively. For end node number 6, the sixth row of this array is updated with −5 and 3 in the first two columns. Because edge 5 is pointing away from the current end node 6, a negative symbol is assigned to the digit 5. The 5th row of the *edgenode* array is updated with the end node informations (3 and 6) of this new edge (5). In the 2nd column of *edgearr*, this new edge number 5 is assigned. In *nodeedgenum*, the existing number in the third and sixth rows is incremented by 1 because global node 3 and global node 6 become associated with new edge '5' in this local edge loop.

For the third local edge, 2 and 3 are supplied as starting and end nodes (see Fig. 2b) respectively from the *edgend* subroutine. For starting node 2, there are two connected edges 3 and 2. Other end nodes of these connected edges (available in the even columns), i.e., 1 and 5 are not matching with other end node 3. As there is no already existing edge between these two nodes, the last assigned global edge (e_l) is incremented from 5 to 6. After this, *edgearr* is updated with this value in the third row. In the second row of the *nodeedge* array, this new edge data (global edge 6) and its other end node (3) are updated in the fifth and sixth columns. For end node 3, the corresponding row of this array (3rd row) is updated with −6 and 2 in the third and fourth columns. The '−' symbol is assigned to the digit 6 because the edge 6 is pointing away from the current node 3. Also, the second row of the *nodeedgenum* array (related to the 2nd global node) is updated from the previous count 2 to 3. Similarly, the third row (related to the 3rd global node) is updated from 1 to 2. The *edgenode* array is also updated with the end nodes (2 and 3) of the newly formed edge in the first and second columns of the corresponding (sixth) row.

For the fourth local edge, end nodes as 5 (starting node) and 6 (end node), e_l with 6 and existing *edgearr* (2, 5 and 6) are supplied. After checking the end node 6 with the other nodes (2 and 4) available in the fifth row of the *nodedge* array, e_l is updated from 6 to 7. This value is assigned to the running local edge. Now, the fourth column of *edgearr* is updated with 7. Thus we complete the 2nd row of Table 3, which shows the element to edge connectivity array of the second element. Figure 5 shows such element to edge connectivity for all the global elements of the finite element meshed domain.

After this, the *nodeedge* array is updated with the new edge number 7 and its associated other node 6 in the 5th and 6th columns of fifth row. Similarly, in the 6th row corresponding to global node 6, the third and fourth columns are updated with digits −7 and 5, respectively. Also, the existing no. of edges in the fifth and sixth rows of the *nodeedgenum* array are incremented by 1 and updated as 3 and 2. Finally, the global edge number 7 and its end nodes (5 and 6) are stored in the corresponding row (seventh) of *edgenode* array as shown in Table 4.

TABLE 6. *Nodeedge* array of the global nodes of the meshed domain.

Global node	Total no. of conecting edges (*nodeedgenum*)	Associated edge and second node of the edge (*nodeedge* (1:8))							
		1st edge	Other node	2nd edge	Other node	3rd edge	Other node	4th edge	Other node
1	2	1	4	3	2	–	–	–	–
2	3	−3	1	2	5	6	3	–	–
3	2	5	6	−6	2	–	–	–	–
4	3	−1	1	4	5	8	7	–	–
5	4	−2	2	−4	4	7	6	9	8
6	3	−5	3	−7	5	11	9	–	–
7	2	−8	4	10	8	–	–	–	–
8	3	−9	5	−10	8	12	9	–	–
9	2	−11	6	−12	8	–	–	–	–

After updating of all the global variables, program comes out from the inner (local edge) loop as shown in Fig. 6 and enters into the outer (element) loop after incrementing as $ele = ele + 1$. Now, the program runs for local edges of the third element. This process repeats until all the discretized elements are finished. Table 3 shows the edge connectivity array for the entire domain; Table 4 shows the end nodes for all the edges; Table 6 shows the complete *nodeedge* array of all the global nodes of the finite element meshed domain after finishing the outer loop for all four elements.

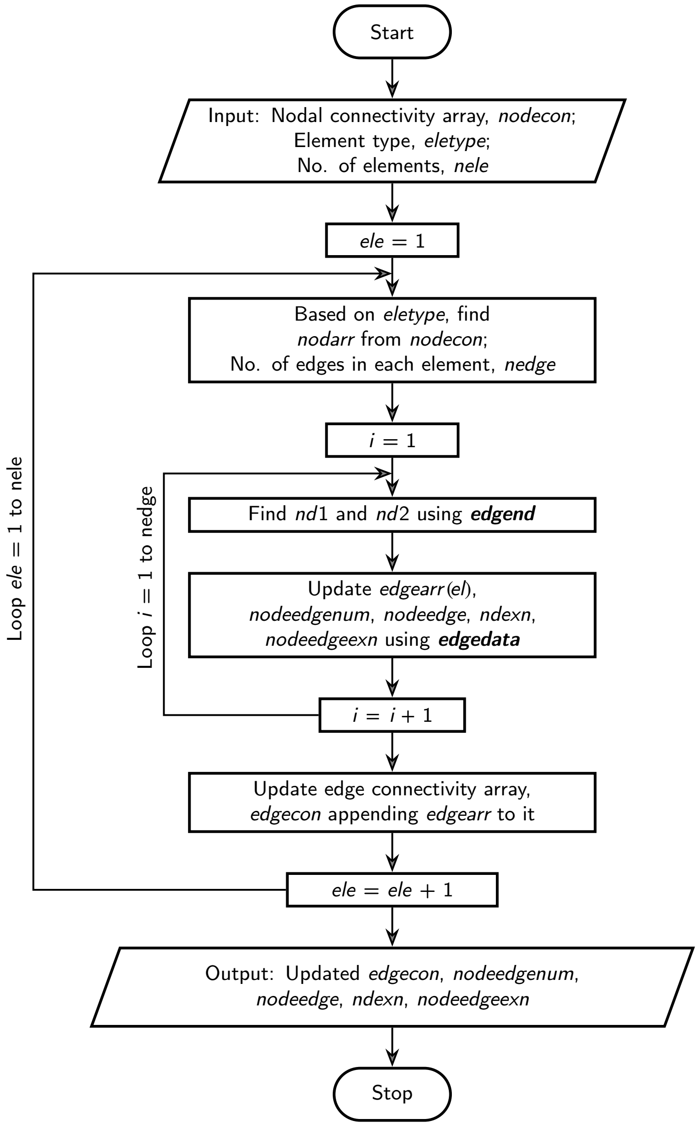

Start
Input: Nodal connectivity array, nodecon;
Element type, eletype;
No. of elements, nele
ele = 1
Based on eletype, find
nodarr from nodecon;
No. of edges in each element, nedge
i = 1
Find nd1 and nd2 using edgend
Update edgearr(el),
nodeedgenum, nodeedge, ndexn,
nodeedgeexn using edgedata
i = i + 1
Update edge connectivity array,
edgecon appending edgearr to it
ele = ele + 1
Output: Updated edgecon, nodeedgenum,
nodeedge, ndexn, nodeedgeexn
Stop
Loop ele = 1 to nele
Loop i = 1 to nedge

FIG. 6. Flow chart of the node to edge structure.

This algorithm can be implemented to other edge elements. These edge elements include 3-edge triangle, 8-edge triangle and 12-edge quadrilateral elements. In the following sections we discuss the implementation of these elements.

2.3. Three-edge triangular element

The lower-order three-edge triangular element is formed from three-node triangle, and Fig. 7a shows the nodal connectivity (local) of the triangular element. For this element the expected edge connectivity is shown in Fig. 7b. With the help of node sets $\mathbf{(1,2)}$, $\mathbf{(2,3)}$ and $\mathbf{(3,1)}$ three edges $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ are formed, respectively. Figure 8a shows one general domain discretized with nodal elements, which can be transformed into a domain discretized with three-edge triangular elements as shown in Fig. 8b. For this triangular element three-edge shape functions are [18] $\mathbf{v}_1 = l_1(\xi\nabla\eta - \eta\nabla\xi)$, $\mathbf{v}_2 = l_2(-\eta\nabla\xi - (1-\xi)\nabla\eta)$ and $\mathbf{v}_3 = l_3((1-\eta)\nabla\xi + \xi\nabla\eta)$, where l_1, l_2 and l_3 are the edge lengths of three edges.

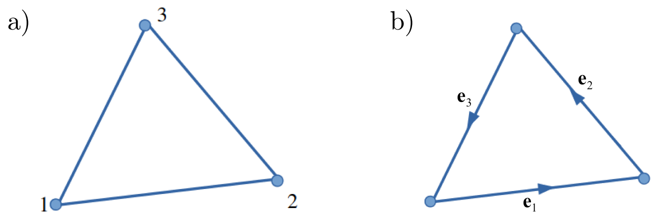


FIG. 7. Triangular element with: a) three nodes, b) three edges.

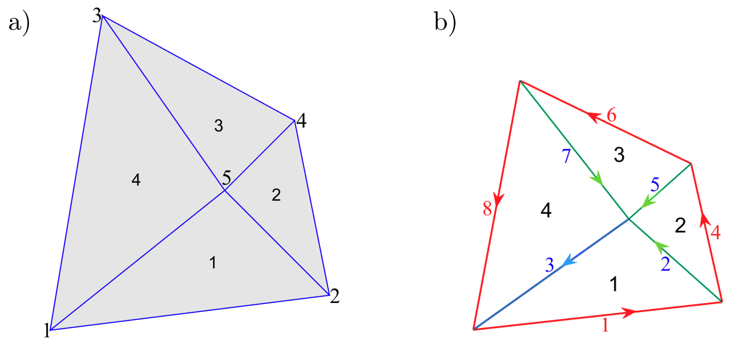


FIG. 8. Discretized domain meshed with:
a) three-node triangular element, b) three-edge triangular element.

2.3.1. Calculation of $\nabla\times\mathbf{E}$. From the relation:

$$\nabla\times\mathbf{E} = \left[\begin{array}{ccc} \dfrac{\partial v_{1y}}{\partial x} - \dfrac{\partial v_{1x}}{\partial y} & \dfrac{\partial v_{2y}}{\partial x} - \dfrac{\partial v_{2x}}{\partial y} & \dfrac{\partial v_{3y}}{\partial x} - \dfrac{\partial v_{3x}}{\partial y} \end{array}\right] \left\{\begin{array}{c} E_1 \\ E_2 \\ E_3 \end{array}\right\} = \mathbf{BE},$$

where E_1, E_2 and E_3 are tangential components of electric fields along the three edges $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$, respectively. The components of the **B**-matrix can be obtained by using the relation:

$$\begin{bmatrix} \dfrac{\partial v_{1x}}{\partial x} & \dfrac{\partial v_{1x}}{\partial y} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial v_{1x}}{\partial \xi} & \dfrac{\partial v_{1x}}{\partial \eta} \end{bmatrix} \begin{bmatrix} \varGamma_{11} & \varGamma_{21} \\ \varGamma_{12} & \varGamma_{22} \end{bmatrix}.$$

We can derive the $\frac{\partial v_{1x}}{\partial \xi}$, $\frac{\partial v_{1x}}{\partial \eta}$, ..., etc., explicitly.

2.3.2. Conversion algorithm. Like four-edge quadrilateral elements, the edge data structure for three-edge triangle can be constructed as described in Subsec. 2.2.2. If any node is shared by more than four edges, then some additional data-structure is required as follows. In this algorithm if any i-th node is shared by more than 4 edges, then the information from the fifth edge onwards is stored in the *nodeedgeexn*, *nndexn* and *ndexn* arrays. In the entire geometry, there will be a few nodes associated with more than four edges. Our *nodeedge* array has a number of rows as a total number of nodes and the number of column as 8 in order to accommodate the first four associated edges, which is very common. *nodeedgeexn* is initialized with a number of rows far smaller than the total number of nodes, as it has 16 columns to accommodate next eight edges. When we come across a node associated with more than four edges, *nndexn* is incremented by 1. Suppose in a flow, we have j-th occurrence of such a node associated with more than four edges. Then, *ndexn(j)* will store the corresponding node number. The j-th row of *nodeedgeexn* will store the information of the associated edge and other node from the fifth edge onwards. This can be understood from the case shown in Figs. 9a and 9b. Node no. 9 of the discretized domain is shared by 8 edges. So the information of the first four connecting edges of node no. 9

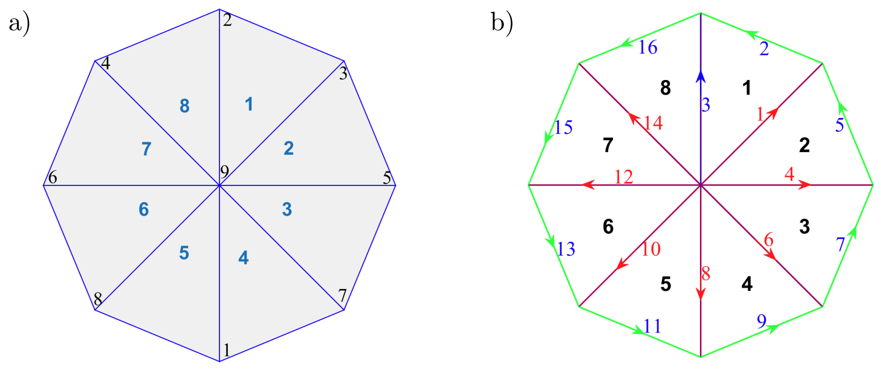


Fig. 9. Discretized circular domain with coarse mesh: a) nodal connectivity of the elements, b) edge connectivity of the elements.

and its other end nodes are stored in the *nodeedge* array as shown in Table 7 after the element loop for the fifth element. But from the fifth to eighth associated edge of node 9, the information is stored in the *nodeedgeexn* array. Table 8 shows update of the information of the *nodeedgeexn* array after the end of the element loop for the fifth element. After the element loop runs for the remaining existing elements, the *nodeedgeexn* array is updated as shown in Table 9. Here, *nndexn* will be 1 and *ndexn*(1) will be 9. We have solved a numerical example of similar domain as discussed in Subsecs. 3.2 and 3.3 to obtain eigenvalues.

Table 7. *Nodeedge* array of nodes after element loop of the fifth element.

Global node	Total no. of conecting edges (*nodeedgenum*)	Associated edge and second node of the edge (*nodeedge* (1:8))							
		1st edge	Other node	2nd edge	Other node	3rd edge	Other node	4th edge	Other node
9	6	1	3	3	2	4	5	6	7
2	2	−2	3	−3	9	–	–	–	–
3	3	−1	9	2	2	−5	5	–	–
5	3	−4	9	5	3	−7	7	–	–
7	3	−6	9	7	5	−9	1	–	–
1	3	−8	9	9	7	−11	8	–	–
8	2	−10	9	11	1	–	–	–	–
–	–	–	–	–	–	–	–	–	–

Table 8. *Nodeedgeexn* array of nodes after element loop of the fifth element.

Global node	Associated edge and second node of the edge (*nodeedgeexn* (1:16))															
	5th edge	a*	6th edge	a*	7th edge	a*	8th edge	a*	9th edge	a*	10th edge	a*	11th edge	a*	12th edge	a*
9	8	1	10	8	–	–	–	–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

a* – other node.

Table 9. *Nodeedgeexn* array of nodes of the discretized domain after complete conversion.

Global node	Associated edge and second node of the edge (*nodeedgeexn* (1:16))															
	5th edge	a*	6th edge	a*	7th edge	a*	8th edge	a*	9th edge	a*	10th edge	a*	11th edge	a*	12th edge	a*
9	8	1	10	8	12	6	14	4	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

a* – other node.

2.4. Eight-edge triangular element

The higher-order triangular edge element (eight-edge triangle) is formed from six-node triangular element. Figure 10a is the local nodal connectivity of the six-node triangular element. Figure 10b represents the desired local edge connectivity of the eight-edge triangular element. Here, eight edges $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, $\mathbf{e}_4$, $\mathbf{e}_5$, $\mathbf{e}_6$, $\mathbf{e}_7$ and $\mathbf{e}_8$ are formed by using the node sets $\mathbf{(1,4)}$, $\mathbf{(4,2)}$, $\mathbf{(2,5)}$, $\mathbf{(5,3)}$, $\mathbf{(3,6)}$, $\mathbf{(6,1)}$, $\mathbf{(6,5)}$ and $\mathbf{(5,4)}$, respectively. l_1, l_2, ..., l_8 are the edge lengths of the element. Here, three edges can be formed on the face of the element. But one edge can be ignored due to independency of three edges. The edge shape functions of the element are [2] $\mathbf{v}_1 = l_1(4\xi-1)(\xi\nabla\eta-\eta\nabla\xi)$, $\mathbf{v}_2 = l_2(4\eta-1)(\xi\nabla\eta-\eta\nabla\xi)$, $\mathbf{v}_3 = l_3(4\eta-1)(\eta\nabla\alpha-\alpha\nabla\eta)$, $\mathbf{v}_4 = l_4(4\alpha-1)(\eta\nabla\alpha-\alpha\nabla\eta)$, $\mathbf{v}_5 = l_5(4\alpha-1)(\alpha\nabla\xi-\xi\nabla\alpha)$, $\mathbf{v}_6 = l_6(4\xi-1)(\alpha\nabla\xi-\xi\nabla\alpha)$, $\mathbf{v}_7 = 4l_7\eta(\alpha\nabla\xi-\xi\nabla\alpha)$ and $\mathbf{v}_8 = 4l_8\xi(\eta\nabla\alpha-\alpha\nabla\eta)$, where $\alpha = 1-\xi-\eta$ and $\nabla\times\mathbf{E}$ can be calculated by using the relation:

$$\nabla\times\mathbf{E} = \left[\frac{\partial v_{1y}}{\partial x} - \frac{\partial v_{1x}}{\partial y} \quad \frac{\partial v_{2y}}{\partial x} - \frac{\partial v_{2x}}{\partial y} \quad \ldots \quad \frac{\partial v_{7y}}{\partial x} - \frac{\partial v_{7x}}{\partial y} \quad \frac{\partial v_{8y}}{\partial x} - \frac{\partial v_{8x}}{\partial y}\right] \begin{Bmatrix} E_1 \\ E_2 \\ \vdots \\ E_7 \\ E_8 \end{Bmatrix} = \mathbf{BE},$$

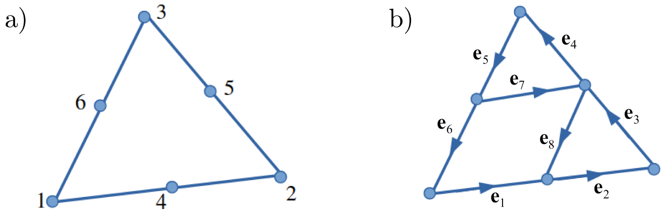


FIG. 10. Higher-order triangular element with: a) six nodes, b) eight edges.

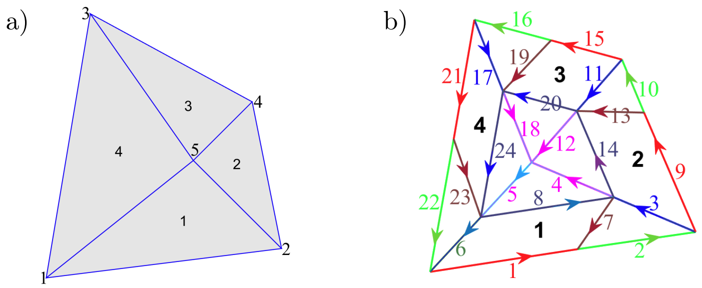


FIG. 11. Discretized quadrilateral domain with: a) six-node triangular element, b) eight-edge triangular element.

where E_1, E_2, ..., E_8 are tangential components of electric fields along the edges $\mathbf{e}_1$, $\mathbf{e}_2$, ..., $\mathbf{e}_8$, respectively. Here, we can obtain the components of the **B**-matrix explicitly. We have used the conversion algorithm, as discussed in Subsec. 2.2.2, to convert one general domain as shown in Fig. 11a discretized with four 6-node triangular elements into the domain meshed with four eight-edge elements shown in Fig. 11b.

2.5. Twelve-edge quadrilateral element

The higher-order quadrilateral edge element with twelve edges is formed from the nine-node quadrilateral element. The edges $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, ..., $\mathbf{e}_{12}$ are formed from the local node sets $\mathbf{(1,2)}$, $\mathbf{(2,3)}$, $\mathbf{(4,5)}$, $\mathbf{(5,6)}$, $\mathbf{(7,8)}$, $\mathbf{(8,9)}$, $\mathbf{(1,4)}$, $\mathbf{(2,5)}$, $\mathbf{(3,6)}$, $\mathbf{(4,7)}$, $\mathbf{(5,8)}$ and $\mathbf{(6,9)}$, respectively. Local nodal connectivity and edge connectivity are shown in Figs. 12a and 12b, respectively. The twelve edge shape functions are [18] $\mathbf{v}_1 = \frac{-l_1}{2}\eta(\eta-1)(\xi-0.5)\nabla\xi$, $\mathbf{v}_2 = \frac{l_2}{2}\eta(\eta-1)(\xi+0.5)\nabla\xi$, $\mathbf{v}_3 = l_3(\eta^2-1)(\xi-0.5)\nabla\xi$, $\mathbf{v}_4 = -l_4(\eta^2-1)(\xi+0.5)\nabla\xi$, $\mathbf{v}_5 = \frac{-l_5}{2}\eta(\eta+1)(\xi-0.5)\nabla\xi$, $\mathbf{v}_6 = \frac{l_6}{2}\eta(\eta+1)(\xi+0.5)\nabla\xi$, $\mathbf{v}_7 = \frac{-l_7}{2}\xi(\xi-1)(\eta-0.5)\nabla\eta$, $\mathbf{v}_8 = l_8(\xi^2-1)(\eta-0.5)\nabla\eta$, $\mathbf{v}_9 = \frac{-l_9}{2}\xi(\xi+1)(\eta-0.5)\nabla\eta$, $\mathbf{v}_{10} = \frac{l_{10}}{2}\xi(\xi-1)(\eta+0.5)\nabla\eta$, $\mathbf{v}_{11} = -l_{11}(\xi^2-1)(\eta+0.5)\nabla\eta$ and $\mathbf{v}_{12} = \frac{l_{12}}{2}\xi(\xi+1)(\eta+0.5)\nabla\eta$, where l_1, l_2, ..., l_{12} are the lengths of the edges of the element.

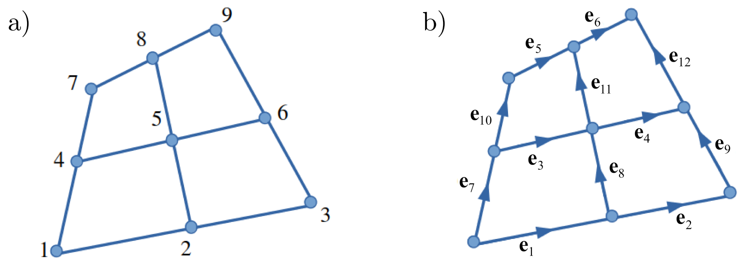


FIG. 12. Higher-order quadrilateral element with: a) nine nodes, b) twelve edges.

$\nabla\times\mathbf{E}$ of the element can be calculated by using the following relation:

$$\nabla\times\mathbf{E}=\left[\frac{\partial v_{1y}}{\partial x}-\frac{\partial v_{1x}}{\partial y}\quad \frac{\partial v_{2y}}{\partial x}-\frac{\partial v_{2x}}{\partial y}\quad \ldots \quad \frac{\partial v_{11y}}{\partial x}-\frac{\partial v_{11x}}{\partial y}\quad \frac{\partial v_{12y}}{\partial x}-\frac{\partial v_{12x}}{\partial y}\right]\begin{Bmatrix} E_1 \\ E_2 \\ \vdots \\ E_{11} \\ E_{12}\end{Bmatrix}=\mathbf{BE},$$

where E_1, E_2, ..., E_{12} are tangential components of electric fields along the edges $\mathbf{e}_1$, $\mathbf{e}_2$, ..., $\mathbf{e}_{12}$, respectively. We can obtain the components of the **B**-matrix explicitly by using the finite difference method or Mathematica software tool [38]. For the general domain shown in Fig. 13a, which shows the domain meshed with four 9-node quadrilateral elements we have used the conversion algorithm to transform into the domain as shown in Fig. 13b. This generated domain is discretized with four 12-edge quadrilateral elements.

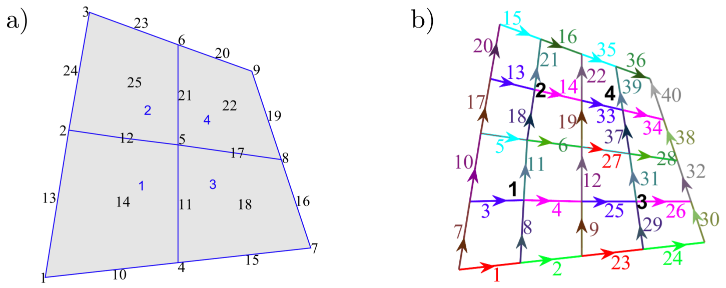


Fig. 13. Discretized domain with: a) nine-node quadrilateral element, b) 12-edge quadrilateral element.

3. Numerical examples

In the frequency domain, the electromagnetic wave equation can be written as [18]:

$$\nabla \times \left(\frac{1}{\mu_r} \nabla \times \mathbf{E} \right) - k_0^2 \epsilon_r \mathbf{E} = -i\omega\mu_0 \mathbf{j}, \tag{1}$$

where $i = \sqrt{-1}$ and $k_0 = \omega/c$ is the wave number in vacuum. From the relations, relative permittivity and relative permeability $\epsilon_r := \epsilon/\epsilon_0$ and $\mu_r := \mu/\mu_0$, where ϵ_0 and μ_0 are the permittivity and permeability for vacuum, and $c = 1/\sqrt{\epsilon_0\mu_0}$ is the speed of light. Assuming current density, $\mathbf{j}$ to be zero, the above equation reduces to

$$\nabla \times \left(\frac{1}{\mu_r} \nabla \times \mathbf{E} \right) = k_0^2 \epsilon_r \mathbf{E}. \tag{2}$$

Equation (2) is used to solve the eigenvalue problems for finding the square of the eigenvalue k_0^2. In order to validate the transformed edge elements we have performed numerical analysis by considering the standard eigenvalue problems. Here, edge elements such as 3-edge triangular, 4-edge quadrilateral, 12-edge quadrilateral and 8-edge triangular elements are represented as T3, Q4, Q12 and T8 respectively. Q4(N) and Q9(N) represents the 4-node quadrilateral and 9-node quadrilateral elements respectively. We assume $\epsilon_r = \mu_r = 1.0$ for all the problems considered in this section.

3.1. Square domain with perfectly conducting boundaries

The square domain of side length π is chosen. For this square domain, all the sides/boundaries are assumed to be perfectly conducting. Here, to conduct the numerical analysis, the domain is discretized with the first-order edge elements (T3 and Q4) and higher-order edge elements (Q12 and T8). Table 10 shows the total number of equations along with the total number of discretized elements for different meshes. For each element, square of eigenvalues is listed in Table 11. Results with the transformed edge elements are in good agreement

TABLE 10. Analysis data of different edge elements for the square domain problem.

Type of element	Total no. of elements	Total free degrees of freedom (equations)
T3	512	736
Q4	256	480
T8	128	1280
Q12	256	1984

TABLE 11. k_0^2 on the square domain for different elements.

Analytical	Edge element			
Benchmark	T3	Q4	Q12	T8
1	0.998066	1.000803	1.000002	0.999992
1	0.999795	1.000803	1.000002	1.000010
2	2.002121	2.001607	2.000004	2.000115
4	3.982881	4.012868	4.000131	4.000089
4	3.982939	4.012868	4.000131	4.000089
5	4.982602	5.013671	5.000133	5.000260
5	5.015107	5.013671	5.000133	5.002108
8	8.032183	8.025735	8.000262	8.006889
9	8.906076	9.065245	9.001478	9.000147
9	8.921107	9.065245	9.001478	9.001707
10	9.950139	10.066048	10.001480	10.005688
10	9.952486	10.066048	10.001480	10.005711
13	12.960172	13.078112	13.001609	13.012005
13	13.133842	13.078112	13.001609	13.037121
16	15.726881	16.206657	16.008194	16.004350
16	15.727173	16.206657	16.008194	16.004383
Number of computed zeros				
–	65	220	217	224

with the analytical results stated in [13]. We can observe that all the elements gave the accurate multiplicity of eigenvalues. The first non-zero eigenvalues of all the elements appeared after stating the number of zeros generated at the machine precision level. These zeros indicate the approximation of null space.

3.2. Circular domain with perfectly conducting surfaces

A circular domain of unit radius with perfectly conducting boundaries is considered to perform eigenanalysis. Our interest to consider this domain is to test the proposed algorithm in handling the data of the additional edges (more than four edges shared at a particular node) as mentioned in Subsec. 2.3.2. In the earlier examples, the whole domains are discretized with only one type of element. But in the present case, the domain is discretized with the combination of first-order edge elements (T3 and Q4) or the combination of higher-order edge elements (Q12 and T8) to perform numerical analysis. Triangular elements are used to mesh the center portion of the domain up to one layer in the r direction. Quadrilateral elements are adopted to discretize the rest of the domain, as shown in Fig. 14. Mesh details of these elements are shown in Table 12, and k_0^2 values are listed in Table 13 and are compared with analytical results reported in [14, 17, 19]. The results of the generated edge elements indicate a strong fit with the analytical results along with the correct multiplicity of eigenvalues.

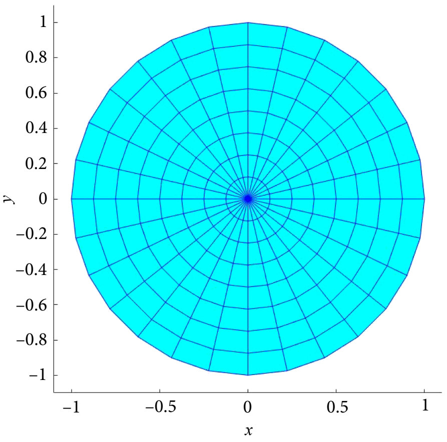


Fig. 14. Discretized circular domain.

Table 12. Analysis data of different edge elements for the circular domain problem.

Type of element	Total no. of triangular elements	Total no. of quadrilateral elements	Total free degrees of freedom (equations)
Q4/T3	30	1770	1430
Q12/T8	20	680	6340

Table 13. k_0^2 on the circular domain for different elements.

Analytical	Edge element	
Benchmark	T3/Q4	T8/Q12
3.391122(2)	3.425827(2)	3.383070(2)
9.329970(2)	9.530267(2)	9.329383(2)
14.680392(1)	14.802788(1)	14.737119(1)
17.652602(2)	18.347489(2)	17.662205(2)
28.275806(2)	28.658911(2)	28.304885(2)
28.419561(2)	30.105560(2)	28.343554(2)
41.158640(2)	45.175889(2)	41.404575(2)
44.970436(2)	45.564095(2)	44.974232(2)
49.224256(1)	49.668776(1)	49.543520(1)
56.272502(2)	64.056343(2)	56.956832(2)
64.240225(2)	65.762272(2)	64.271185(2)
Number of computed zeros		
–	494	629

3.3. Cracked circular domain with perfectly conducting surfaces

In this example, the same circular domain but with the crack running from the center to the side of the circle is taken into consideration, as shown in Fig. 15. Here, the domain's crack is modeled by using the 'double noding' method at the same position. Mesh details are shown in Table 14 including the total number of equations. Table 15 shows the numerical values along with the computed zeros and they are compared with analytical results reported in [14, 17, 19]. The results of the edge elements (T8/Q12) demonstrate close matching with the analytical results.

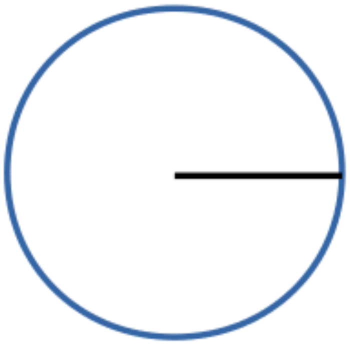


Fig. 15. Cracked circular domain.

Table 14. Analysis data of different edge elements for the cracked circular domain problem.

Type of element	Total no. of triangular elements	Total no. of quadrilateral elements	Total free degrees of freedom (equations)
Q4/T3	30	1770	3631
Q12/T8	20	680	5591

Table 15. k_0^2 on the cracked circular domain for different elements.

Analytical	Edge element	
Benchmark	T3/Q4	T8/Q12
1.358390	1.362745	1.297322
3.391122	3.425901	3.383104
6.059858	6.146835	6.053942
9.329970	9.530267	9.329383
13.195056	13.589143	13.201275
14.680392	–	15.304855
17.652602	18.347489	17.662205
21.196816	21.297238	20.547667
22.681406	23.838848	22.709137
28.275806	28.660201	28.305522
Number of computed zeros		
–	546	659

3.4. Curved L-shape domain with perfectly conducting surfaces

In this example, solved in [13], the domain has three straight and three circular sides of radii 1, 2 and 3 and Fig. 16 shows such domain discretized with Q4 elements. Here, all the boundary edges of the domains are perfectly conducting. This problem is quite complicated due to the existence of singular eigenvalue for the sharp corner and curvature effect. To find the eigenvalues, we discretize the domain with both nodal and edge elements. Here, edge elements include T3, Q4, Q12 and T8 elements, whereas nodal elements are Q4(N) and Q9(N). In Table 16, we present the total number of equations and total number of elements for each type of element. For all the elements, the numerical results obtained are listed in Table 17 along with the number of computed zeros. We compare the numerical results of these elements with analytical values taken from [13]. It can be observed that higher-order edge element Q12 results matches with the benchmark values up to the second decimal and for T8 elements it matches up to the first decimal. In the case of nodal elements, Q4(N) and Q9(N) failed to predict the first singular eigenfrequency. Also with nodal elements, spurious eigenvalues are predicted, which can be seen in the third row of Table 17.

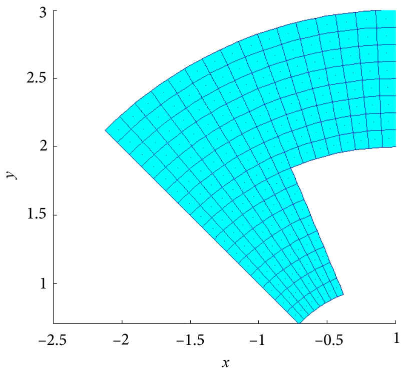


Fig. 16. Mesh for curved L-shape domain.

Table 16. Analysis data of different edge and nodal elements for the curved L-shape domain problem.

Type of element	Total no. of elements	Total free degrees of freedom (equations)
T3	600	860
Q4	300	560
T8	96	448
Q12	108	816
Q4(N)	300	909
Q9(N)	108	1425

Table 17. k_0^2 on the curved L-shaped domain for different elements.

Analytical	Edge element					
Benchmark	Q4(N)	Q9(N)	T3	Q4	Q12	T8
1.818571	–	–	1.797075	1.811631	1.814860	1.807729
3.490576	3.760217	3.689097	3.491215	3.500850	3.490516	3.4954251
–	6.786038	5.033585	–	–	–	–
10.065602	10.050998	10.153471	10.047041	10.151037	10.066760	10.082410
10.111886	10.235756	10.284415	10.101835	10.203259	10.112480	10.127115
12.435537	15.027126	13.845497	12.397735	12.510154	12.429986	12.431268
Number of computed zeros						
–	306	475	101	220	217	224

We want to make a comparison between edge and nodal elements in terms of computational cost and performance of elements in obtaining the solutions during the numerical analysis. For this purpose, we have calculated the percentage of error with analytical values for the first five eigenvalues to both nodal and edge elements. Total no. of equations (which represents the computational cost) along with the percentage of error values for all types of elements are presented in the Table 18. We have chosen the meshes for nodal and edge elements (please see Table 16) such that nodal element meshes result in more no. of equations than respective edge element meshes. It can be seen in Table 18 that for almost all the edge element meshes have less than 1% error with the analytical benchmarks where as there are around 11% and 20% errors for the 5th eigenfrequency with two nodal mesh Q4(N) and Q9(N), respectively. Also, as reported in the literature, the nodal meshes are not able to capture singular eigenvalue 1.818571, which is captured by all the edge elements successfully. Total free degrees of freedom represent the total no. of equations, which is representative of the computational cost.

TABLE 18. Comparison between nodal and edge elements for percentage error with analytical benchmark solutions.

Type of element	Total no. of free degrees of freedom (equations)	% error for 1st eigen-frequencies	% error for 2nd eigen-frequencies	% error for 3rd eigen-frequencies	% error for 4th eigen-frequencies	% error for 5th eigen-frequencies
T3	860	1.182027	0.018306	0.184400	0.099398	0.303984
Q4	560	0.381618	0.294335	0.848782	0.903620	0.600030
T8	448	0.596182	0.138920	0.166985	0.150605	0.034329
Q12	816	0.204061	0.001719	0.011505	0.005874	0.044638
Q4(N)	909	–	5.687342	0.872963	1.706200	11.338151
Q9(N)	1425	–	7.724828	0.145088	1.224994	20.840186

4. Conclusions

Electromagnetic analysis with nodal finite elements has several shortcomings. Nodal FEM cannot model the null space accurately, there are spurious values. With regularization or penalty method this spurious values are shifted towards the higher end. But with this method, an ad hoc penalty parameter is required to be adjusted. Also, with this penalty method singular eigenvalues for the domains with sharp edges and corners cannot be approximated accurately. With the edge finite element method all these limitations are addressed without the use of any ad hoc penalty parameter. Furthermore, in nodal FEM, it is required to decompose electric and magnetic fields into scalar and vector potentials to attain the necessary continuity requirement across elements. After the FEM analysis, fields are calculated from the potentials with additional postprocessing. In edge FEM, we can formulate directly in terms of field variables. But most of the preprocessors in practice generate FEM meshes in terms of nodal connectivities. Hence, in this article we have presented a very useful novel conversion technique transforming the nodal connectivities into edge connectivities. Also, this conversion algorithm generates other necessary data structures in edge formulations such as direction information of the edges, connecting nodes of a particular edges, associated edges of a particular node and respective other node of those edges. This algorithm converts the 4-node quadrilateral into the 4-edge quadrilateral, the 3-node triangle into the 3-edge triangle, the 6-node triangle into the 8-edge triangle, and the 9-node quadrilateral into the 12-edge quadrilateral. For some special geometries, combinations of triangular and quadrilateral elements are more effective. Our conversion algorithm is capable to combine successfully a 3-edge triangle with a 4-edge quadrilateral and a 8-edge triangle with a 12-edge quadrilateral. In Subsecs. 3.2 and 3.3, the successful implementation of such combination was presented with numerical examples. Some special treatment is required in the data structure for the nodes connected to many edges; this is explained in detail in Subsec. 2.3.2 with associated representative examples. This additional data structure is verified with numerical examples in Subsecs. 3.2 and 3.3. The effectiveness of the conversion technique is tested with different standard benchmark examples. These numerical examples include square domain, circular domain, cracked circular domain, curved L-shape domain, etc. Our proposed conversion technique gives accurate k_0^2 values along with correct multiplicity for both convex and non-convex domains. For non-convex domains, singular eigenvalues are predicted without any spurious modes. The perfect match with the benchmark results for different examples in terms of eigenvalues, their multiplicities, singular eigenvalues for the domain with sharp corners and edges, all exhibit the correctness and efficacies of the conversion algorithm. Our edge elements are performing accurately with more computational efficiency than nodal elements.

Acknowledgments

Supported by Science and Engineering Research Board (SERB), and Department of Science and Technology (DST), Government of India, under the project IMP/2019/000276 and VSSC, ISRO through MoU No.: ISRO:2020:MOU:NO: 480.

References

1. M. Agrawal, C.S. Jog, Monolithic formulation of electromechanical systems within the context of hybrid finite elements, *Computational Mechanics*, **59** (3): 443–457, 2017, doi: 10.1007/s00466-016-1356-1.

2. A. Ahagon, T. Kashimoto, Three-dimensional electromagnetic wave analysis using high order edge elements, *IEEE Transactions on Magnetics*, **31**(3): 1753–1756, 1995, doi: 10.1109/20.376375.

3. M. Ainsworth, J.F. Coyle, P.D. Ledger, K. Morgan, Computing Maxwell eigenvalues by using higher order edge elements in three dimensions, *IEEE Transactions on Magnetics*, **39**(5): 2149–2153, 2003, doi: 10.1109/TMAG.2003.817097.

4. Nandy Arup Kumar, Robust Finite Element Strategies for Structures, Acoustics, Electromagnetics and Magneto-hydrodynamics, Ph.D. thesis, Indian Institute of Science Bangalore, Department of Mechanical Engineering, IISc, Bangalore, India, https://etd.iisc.ac.in/handle/2005/2913.

5. M.L. Barton, Z.J. Cendes, New vector finite elements for three-dimensional magnetic field computation, *Journal of Applied Physics*, **61**(8): 3919–3921, 1987, doi: 10.1063/1.338584.

6. D. Boffi, Finite element approximation of eigenvalue problems, *Acta Numerica*, **19**: 1–120, 2010, doi: 10.1017/S0962492910000012.

7. D. Boffi, M. Farina, L. Gastaldi, On the approximation of Maxwell's eigenproblem in general 2D domains, *Computers & Structures*, **79**: 1089–1096, 2001, doi: 10.1016/S0045-7949(01)00003-7.

8. D. Boffi, P. Fernandes, L. Gastaldi, I. Perugia, Computational models of electromagnetic resonators: analysis of edge element approximation, *SIAM Journal on Numerical Analysis*, **36**(4): 1264–1290, 1999, doi: 10.1137/S003614299731853X.

9. A. Bossavit, A rationale for 'edge-elements' in 3-D fields computations, *IEEE Transactions on Magnetics*, **24**(1): 74–79, 1988, doi: 10.1109/20.43860.

10. A. Bossavit, J.-C. Vérité, A mixed FEM-BIEM method to solve 3-D eddy-current problems, *IEEE Transactions on Magnetics*, **18**(2): 431–435, 1982, doi: 10.1109/TMAG.1982.1061847.

11. J.H. Bramble, T. Kolev, J.E. Pasciak, The approximation of the Maxwell eigenvalue problem using a least-squares method, *Mathematics of Computation*, **74**(252): 1575–1598, 2005, doi: 10.1090/S0025-5718-05-01759-X.

12. Z.J. Cendes, Vector finite elements for electromagnetic field computation, *IEEE Transactions on Magnetics*, **27**(5): 3958–3966, 1991, doi: 10.1109/20.104970.

13. M. Dauge, Benchmark computations for Maxwell equations for the approximation of highly singular solutions, 2004, http://perso.univ-rennes1.fr/monique.dauge/core/index.html.

14. A. Elsherbeni, D. Kajfez, S. Zeng, Circular sectoral waveguides, *IEEE Antennas and Propagation Magazine*, **33**(6): 20–27, 1991, doi: 10.1109/74.107352.

15. L.E. Garcia-Castillo, M. Salazar-Palma, Second-order Nédélec tetrahedral element for computational electromagnetics, *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, **13**, 261–287, 2000, doi: 10.1002/(SICI)1099-1204(200003/06)13:2/3<261::AID-JNM360>3.0.CO;2-L.

16. R.D. Graglia, D.R. Wilton, A.F. Peterson, Higher order interpolatory vector bases for computational electromagnetics, *IEEE Transactions on Antennas and Propagation*, **45**(3): 329–342, 1997, doi: 10.1109/8.558649.

17. R.F. Harrington, *Time-Harmonic Electromagnetic Fields*, McGraw-Hill, New York, NY, 1961, https://cds.cern.ch/record/230916.

18. J.-M. Jin, *The Finite Element Method in Electromagnetics*, Third ed., John Wiley & Sons, New Jersey, 2014.

19. C.S. Jog, A. Nandy, Mixed finite elements for electromagnetic analysis, *Computers and Mathematics with Applications*, **68**(8): 887–902, 2014, doi: 10.1016/j.camwa.2014.08.006.

20. D. Kamireddy, A. Nandy, Combination of triangular and quadrilateral edge element for the eigenvalue analysis of electromagnetic wave propagation, *European Journal of Molecular & Clinical Medicine*, **7**(11): 1656–1663, 2020, https://ejmcm.com/article_5696.html.

21. D. Kamireddy, A. Nandy, Creating edge element from four node quadrilateral element, *IOP Conference Series: Materials Science and Engineering*, **1080**, 2021, doi: 10.1088/1757-899x/1080/1/012015.

22. M. Koshiba, K. Hayata, M. Suzuki, Finite-element formulation in terms of the electric-field vector for electromagnetic waveguide problems, *IEEE Transactions on Microwave Theory and Techniques*, **33**(10): 900–905, 1985, doi: 10.1109/TMTT.1985.1133148.

23. J.F. Lee, D.K. Sun, Z.J. Cendes, Tangential vector finite elements for electromagnetic field computation, *IEEE Transactions on Magnetics*, **27**(5): 4032–4035, 1991, doi: 10.1109/20.104986.

24. A. Nandy, C.S. Jog, An amplitude finite element formulation for electromagnetic radiation and scattering, *Computers & Mathematics with Applications*, **71**(7): 1364–1391, 2016, doi: 10.1016/j.camwa.2016.02.013.

25. A. Nandy, C.S. Jog, A monolithic finite-element formulation for magnetohydrodynamics, *Sadhana – Academy Proceedings in Engineering Sciences*, **43**: 1–18, 2018, doi: 10.1007/s12046-018-0905-z.

26. A. Nandy, C.S. Jog, Conservation properties of the trapezoidal rule for linear transient electromagnetics, *Journal of Advances in Mathematics and Computer Science*, **26**(4): 1–26, 2018, doi: 10.9734/JAMCS/2018/39632.

27. J.C. Nedelec, Mixed finite elements in $\mathbb{R}^3$, *Numerische Mathematik*, **35**: 315–341, 1980, doi: 10.1007/BF01396415.

28. R. Otin, Regularized Maxwell equations and nodal finite elements for electromagnetic field computations, *Electromagnetics*, **30**(1–2): 190–204, 2010, doi: 10.1080/02726340903485489.

29. K.D. Paulsen, D.R. Lynch, Elimination of vector parasites in finite element Maxwell solutions, *IEEE Transactions on Microwave Theory and Techniques*, **39**(3): 395–404, 1991, doi: 10.1109/22.75280.

30. U. Pekel, R. Mittra, An application of the perfectly matched layer (PML) concept to the finite element method frequency domain analysis of scattering problems, *IEEE Microwave and Guided Wave Letters*, **59**(8): 258–260, 1995, doi: 10.1109/75.401074.

31. F. Rapetti, A. Bossavit, Whitney forms of higher degree, *SIAM Journal on Numerical Analysis*, **47**(3): 2369–2386, 2009, doi: 10.1137/070705489.

32. C.J. Reddy, M.D. Deshpande, C.R. Cockrell, F.B. Beck, Finite element method for eigenvalue problems in electromagnetics, *NASA Technical Paper*, 3485, http://ecee.colorado.edu/~ecen5004/PDFs/FiniteElementCJReddy.pdf.

33. Seung-Cheol Lee, Jin-Fa Lee, R. Lee, Hierarchical vector finite elements for analyzing waveguiding structures, *IEEE Transactions on Microwave Theory and Techniques*, **51**(8): 1897–1905, 2003, doi: 10.1109/TMTT.2003.815263.

34. X.Q. Sheng, J.M. Jin, C.C. Lu, W.C. Chew, On the formulation of hybrid finite-element and boundary-integral methods for 3-D scattering, *IEEE Transactions on Antennas and Propagation*, **46**(3): 303–311, 1998, doi: 10.1109/8.662648.

35. J.P. Webb, Edge elements and what they can do for you, *IEEE Transactions on Magnetics*, **29**: 1460–1465, 1993, doi: 10.1109/CEFC.1992.720787.

36. J.P. Webb, Hierarchal vector basis functions of arbitrary order for triangular and tetrahedral finite elements, *IEEE Transactions on Antennas and Propagation*, **47**: 1244–1253, 1999, doi: 10.1109/8.791939.

37. J.P. Webb, V.N. Kanellopoulos, Absorbing boundary conditions for the finite element solution of the vector wave equation, *Microwave and Optical Technology Letters*, **2**: 370–372, 1989, doi: 10.1002/mop.4650021010.

38. Wolfram Research, Inc., Mathematica 10.4.1, Champaign, IL (2020), https://www.wolfram.com.

39. T.V. Yioultsis, T. Tsiboukis, Development and implementation of second and third order vector finite elements in various 3-D electromagnetic field problems, *IEEE Transactions on Magnetics*, **33**(2): 1812–1815, 1997, doi: 10.1109/20.582630.

40. T.V. Yioultsis, Multiparametric vector finite elements: a systematic approach to the construction of three-dimensional, higher order, tangential vector shape functions, *IEEE Transactions on Magnetics*, **32**(3): 1389–1392, 1996, doi: 10.1109/20.497506.

Received September 8, 2021; revised version February 11, 2022.