Element Edge Based Discretization for TCAD Device Simulation

Juan E. Sanchez, *Senior Member, IEEE*, and Qiusong Chen

***Abstract*— Technology computer-aided design (TCAD) semiconductor device simulators solve partial differential equations (PDE) using the finite volume method (FVM), or related methods. While this approach has been in use over several decades, its methods continue to be extended, and are still applicable for investigating novel devices. In this paper, we present an element edge based (EEB) FVM discretization approach suitable for capturing vector-field effects. Drawing from a 2D approach in the literature, we have extended this method to 3D. We implemented this method in a TCAD semiconductor device simulator, which uses a generalized PDE (GPDE) approach to simulate devices with the FVM. We describe how our EEB method is compatible with the GPDE approach, allowing the modeling of vector effects using scripting. As an example, this method is applied to solve polarization effects in ferroelectric materials, with examples of a 3D ferro capacitor and a 2D ferroelectric field-effect transistor.**

***Index Terms*— Device Simulation, FeFET, Semiconductor, TCAD**

I. Introduction

TECHNOLOGY computer-aided design (TCAD) semiconductor device simulators solve continuum partial differential equations (PDEs) of the drift-diffusion model for semiconductors on a discretized mesh [1]. The finite volume method (FVM) is the most widely used approach for assembling the PDEs. In the TCAD device simulation literature, the FVM is referred as finite boxes or control volume and is closely related to finite differences [2].

The popularity of the FVM is due to the success of the Scharfetter-Gummel method (SGM) in calculating the electron and hole current densities from exponentially varying carrier densities [3], which exhibits superior numerical stability than other approaches [1], [4]. While the finite element method (FEM) presents advantages in some circumstances [5], [6], the lack of an equivalent to the SGM often leads to hybrid FVM/FEM approaches when solving the device equations with an otherwise FEM solver [7]–[9]. While sharing some of the methods of device simulators, TCAD process simulators are more amenable to FEM-based approaches for the set of PDEs they are meant to solve [6].

Standard simulation models fit well into a node based approach, as volume integration and surface integration of the PDEs are accounted for in the calculation of quantities at the mesh nodes, and at the edges connecting to adjacent mesh nodes.

Advanced simulation models use vector fields and require information from node quantities off of the edge. For mobility models, this requires knowledge of electric field parallel and normal to the direction of current flow [10]. For the ferroelectric effect, the vector components of electric field and polarization are considered [11], [12]. Nodal approaches have been proposed for evaluation of these models, which consider all adjacent nodes to the node of interest [13], [14]. These approaches consider a constant field over the entire control volume bounded by these nodes.

In contrast, an element edge based (EEB) approach has the advantage that the control volume is bounded by the sub volume along each edge of the element being considered. This makes it possible to consider phenomena occurring over smaller scales, such as for impact ionization models [15]. Laux used a 2D EEB approach to model generalized mobility [10] and impact ionization [15], where vector fields are calculated for each edge of a triangular element. In [16], this discretization was compared with other approaches for impact ionization, and it was found to be less susceptible to changes in mesh size than the other methods they considered. This was attributed to the EEB approach using a smaller effective control volume for each element edge. In this paper, we adopt this approach for 2D and extend it for tetrahedral elements in 3D.

Generalized PDE (GPDE) simulators use an equation description as input from the user [4], [6], [14], [17], [18]. The goal is to allow the rapid development of new models and applications for the continuum based approach. Model development time is reduced when the model fits into the existing capabilities of the simulator and is done through a scripting interface. This is in contrast to using a compiled computer language, like C++.

Many GPDE simulators do not consider vector effects, as equation assembly is often restricted to node or edge based quantities [4]. Other simulators do not completely model equation sensitivities to vector effects. Some may provide problem specific operators, limiting the general purpose nature of their approach. One group compared 3 TCAD simulation approaches to model the ferroelectric capacitance effect [14]. They report better convergence for methods where more complete model sensitivities are considered, such as ensuring that the electric field is fully coupled with the field dependent polarization model.

J. E. Sanchez is with DEVSIM LLC, PO Box 50096, Austin, TX 78763, USA (e-mail: jsanchez@devsim.com)

Q. Chen, is with the Department of Materials Science, Fudan University, 220 Handan Road, Shanghai 200433, China (e-mail: chenqiusong@gmail.com)

In this paper, we describe an EEB approach to modeling vector-field effects in DEVSIM, an open source GPDE semiconductor device simulator, for both 2D and 3D device simulation [19]. We describe the approach in Section II. In Section III, present a simulation examples for the hysteresis effects in a ferroelectric capacitor in 3D and a ferroelectric field effect transistor (FeFET) simulation results in 2D.

II. Simulator Methods

A. Equation Assembly

1) Newton Method: The Newton method requires the evaluation of the model equations, as well as the derivatives with respect to the solution variable [1]. This results in a formulation:

$$J\Delta x = -F \tag{1}$$

where J is the Jacobian, ΔX is update to the solution vector, and F is referred to as the right-hand side (RHS) vector. As the method converges at each iteration, $|F| \rightarrow 0$ and $|\Delta x| \rightarrow 0$, to the limits of floating point precision. For TCAD simulation, the nonlinear nature of the PDEs often require a reasonable initial guess, and accurate derivatives, to get convergence or a good convergence rate.

The PDEs considered in this paper fit in the form

$$F^a: \quad \nabla \cdot \overrightarrow{C} + \nabla \cdot \overrightarrow{A} + B = 0 \tag{2}$$

where $\overrightarrow{C}$ is an EEB quantity, $\overrightarrow{A}$ is an edge quantity, and B is a node quantity. The label F^a refers to equation a being considered, as there are multiple simultaneously coupled PDEs in semiconductor simulation[1].

2) Node Assembly: Fig. 1 shows a mesh node in 2D surrounded by triangular elements. The B term is integrated over the `NodeVolume`. Using the Newton method, the RHS entry for this scalar quantity:

$$F^a_i \mathrel{+}= B_i \cdot \texttt{NodeVolume}_i \tag{3}$$

for each node i on the simulation mesh. In this, and subsequent equations, $+=$ represents addition of the term to an existing matrix or vector entry. Similarly, $-=$ represents the subtraction from matrix and vector entries.

Similarly for the Jacobian:

$$J^{a,x}_{i,i} \mathrel{+}= \frac{\partial B_i}{\partial x_i} \cdot \texttt{NodeVolume}_i \tag{4}$$

where x_i is a simulation variable on the same node. The Jacobian entry is placed in the row corresponding to equation a and the column corresponding to simulation variable x.

3) Edge Assembly : Fig. 2 shows the `EdgeCouple` over which flux is integrated for the $\overrightarrow{A}$ term in (2). The calculation of `EdgeCouple` is described in Section II-A.4.

For each edge consisting of nodes $\langle i, j \rangle$

$$F^a_i \mathrel{+}= A_{i,j} \cdot \texttt{EdgeCouple}_{i,j} \tag{5}$$

$$F^a_j \mathrel{-}= A_{i,j} \cdot \texttt{EdgeCouple}_{i,j} \tag{6}$$

[1]The case where B involves a time-derivative is also handled by the simulator. This paper does not consider our approach to time integration methods or boundary conditions.

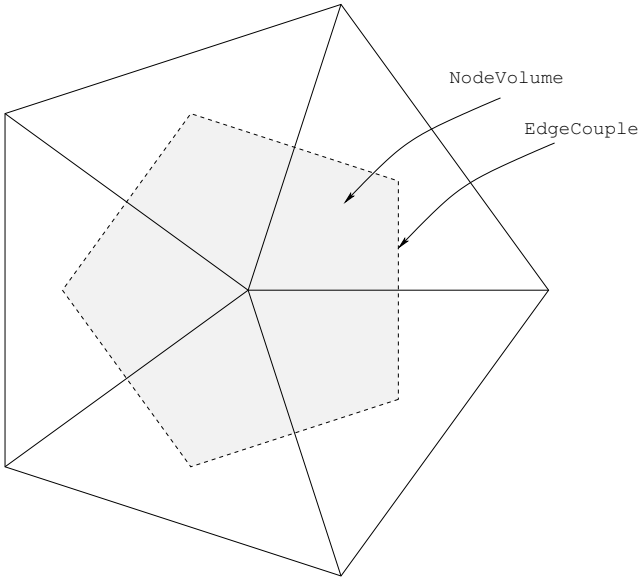

Fig. 1. The 2D mesh cell with an arbitrary number of triangle elements connected to node. The volume of a node is bounded by the perpendicular bisectors of each triangle [20].

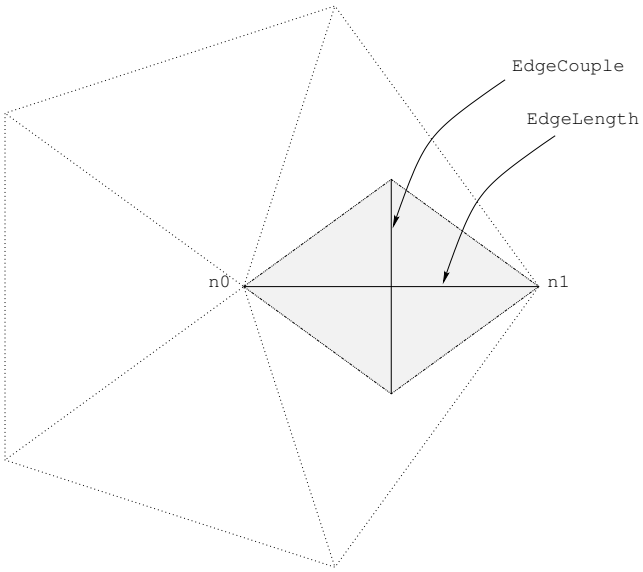

Fig. 2. The 2D mesh cell depicting how flux is integrated along each edge [20]. The `EdgeCouple` is the length of the perpendicular bisectors for the 2 triangles along the edge connecting nodes $\langle$`n0`, `n1`$\rangle$.

where $A_{i,j}$ is the vector quantity evaluated on a mesh edge, and $\texttt{EdgeCouple}_{i,j}$ is the cross section of the mesh edge.

The Jacobian entries are:

$$J^{a,x}_{i,k} \mathrel{+}= \frac{\partial A_{i,j}}{\partial x_k} \cdot \texttt{EdgeCouple}_{i,j} \tag{7}$$

$$J^{a,x}_{j,k} \mathrel{-}= \frac{\partial A_{i,j}}{\partial x_k} \cdot \texttt{EdgeCouple}_{i,j} \tag{8}$$

where x_k is the simulation variable on one of the 2 nodes on the edge.

The data structures for this approach require the location of nodes in the mesh, and the edges connecting them to adjacent nodes. In this approach, the element type or dimension is not relevant to the assembly process, other than to calculate

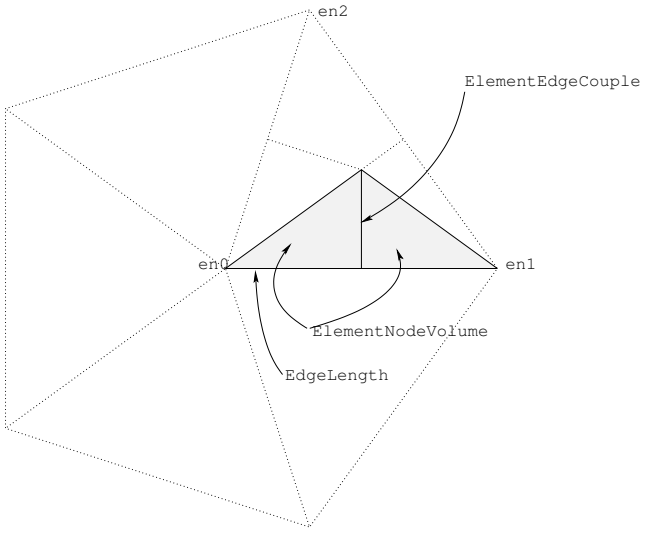

Fig. 3. The mesh cell in 2D. Each triangle is subdivided into 3 subvolumes. The labels for nodes $\langle$en0, en1, en2$\rangle$ are specific to the element edge in the sub element bounded by the solid line [20].

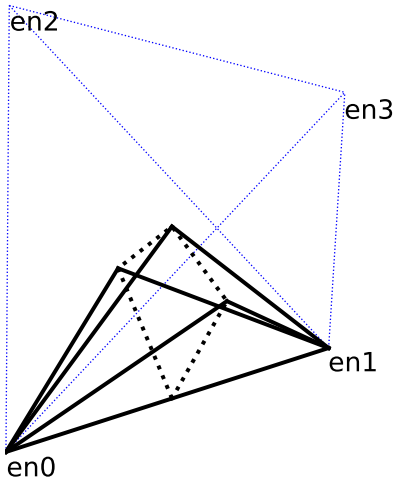

Fig. 4. The element cell in 3D. The tetrahedron is subdivided into 6 subvolumes along each element edge. The element edge has the edge nodes labeled $\langle$en0, en1$\rangle$. The other nodes are labeled $\langle$en2, en3$\rangle$.

EdgeCouple and NodeVolume.

4) Element Edge Assembly: We consider the $\overrightarrow{C}$ contribution to (2), which is an EEB contribution, which is dependent on variables on all of element nodes.

Fig. 3 shows a 2D representation of the volume and area over which the integration occurs. In 2D, the ElementEdgeCouple around each node is from the center of the triangle circumcenter to the center of the edge. Fig. 4 shows the element edge volume in 3D, where the ElementEdgeCouple is from the circumcenter of the tetrahedral element, to the circumcenters of the element triangles connected to the element edge, to the center of the element edge. In both 2D and 3D, the EdgeCouple (Section II-A.3) is calculated by summing up the ElementEdgeCouple for all element edges connected to the edge being considered.

Similar to (5)–(6), the right hand side contribution of the additional term is:

$$F_i^a \mathrel{+}= C_{i,j}^t \cdot \texttt{ElementEdgeCouple}_{i,j}^t \tag{9}$$
$$F_j^a \mathrel{-}= C_{i,j}^t \cdot \texttt{ElementEdgeCouple}_{i,j}^t \tag{10}$$

where index t refers to the element containing the edge. In 2D, the triangle has 3 sub volumes over which to perform the integration. In 3D, the tetrahedron has 6 sub volumes contributing to these equations.

Similar to (7)–(8), the Jacobian entries are:

$$J_{i,k}^{a,x} \mathrel{+}= \frac{\partial C_{i,j}^t}{\partial x_k} \cdot \texttt{ElementEdgeCouple}_{i,j}^t \tag{11}$$
$$J_{j,k}^{a,x} \mathrel{-}= \frac{\partial C_{i,j}^t}{\partial x_k} \cdot \texttt{ElementEdgeCouple}_{i,j}^t \tag{12}$$

where x_k is a simulation variable on one of the nodes on the element. In 2D, this is the 2 nodes, $\langle$en0, en1$\rangle$, on the element edge being considered, and a third node, en2, on a triangular element as shown in Fig. 3. In 3D, there are 2 nodes on the each element edge, $\langle$en0, en1$\rangle$, as well as 2 additional nodes on the tetrahedral element $\langle$en2, en3$\rangle$, as shown in Fig. 4.

5) Edge Volume Assembly : It is also possible to do a volume integration using either the edge based or EEB assembly. This is useful for models such as the density gradient model [21] or impact ionization model [15].

To perform volume integration along an edge we consider

$$F_i^a \mathrel{+}= X_{i,j} \cdot \texttt{EdgeNodeVolume}_{i,j} \tag{13}$$
$$F_j^a \mathrel{+}= X_{i,j} \cdot \texttt{EdgeNodeVolume}_{i,j} \tag{14}$$

where $X_{i,j}$ is an edge model based on quantities at nodes $\langle i,j\rangle$. The $\texttt{EdgeNodeVolume}_{i,j}$ (see Fig. 2) is the volume along the edge connected by the same nodes.

Similarly, for element edges:

$$F_i^a \mathrel{+}= Y_{i,j}^{t,0} \cdot \texttt{ElementNodeVolume}_{i,j}^t \tag{15}$$
$$F_j^a \mathrel{+}= Y_{i,j}^{t,1} \cdot \texttt{ElementNodeVolume}_{i,j}^t \tag{16}$$

where $Y_{i,j}^{t,0}, Y_{i,j}^{t,1}$ are models evaluated on the element edge connected by nodes $\langle i,j\rangle$ on the element t. The $\texttt{ElementNodeVolume}_{i,j}^t$ (see Fig. 3) is the volume along the element edge connected by the same nodes. The superscripts $0,1$ represents that the model may be different for the first or second node of the element edge.

The Jacobian entries are then calculated in a manner analogous to (7)–(8) and (11)–(12)

B. Element Edge Based Fields

1) Calculating the Vector Field: Consider a vector field that we wish to calculate for an edge of a tetrahedral element. We begin by calculating the components along each mesh edge in the device region. For the case of electric field, this is

$$E_{i,j} = \left(\psi_i - \psi_j\right)/L_{j,i} \tag{17}$$

where ψ_i and ψ_j are the potentials at nodes $\langle i,j\rangle$, and $L_{j,i}$ is the distance between them. The derivatives with respect to the node potentials are:

$$\frac{\partial E_{i,j}}{\partial \psi_i} = 1/L_{j,i} \qquad \frac{\partial E_{i,j}}{\partial \psi_j} = -1/L_{j,i} \tag{18}$$

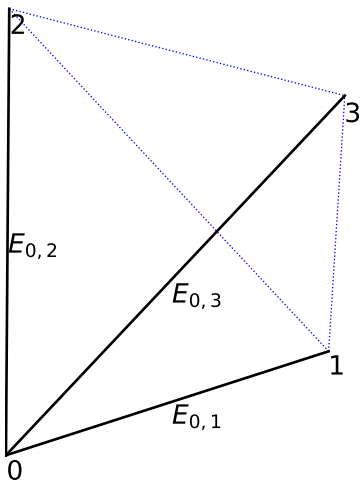

Fig. 5. Depiction on how vector field is calculated from the scalar fields calculated along the **3** edges connected to node **0**.

If the electric field on the element, $\overrightarrow{\mathscr{E}}$, is known, this is equivalent to

$$E_{i,j} = \hat{s}_{i,j} \cdot \overrightarrow{\mathscr{E}} \tag{19}$$

where $\hat{s}_{i,j}$ is the unit vector along the edge connecting nodes $\langle i,j \rangle$. In Fig. 5, we show how these scalar fields are used to calculate the vector field on node 0.

For tetrahedra with nodes $\langle 0,1,2,3 \rangle$, we use the components of the unit vectors to calculate the electric field for all edges connected to node 0

$$\begin{pmatrix} \hat{s}^x_{0,1} & \hat{s}^y_{0,1} & \hat{s}^z_{0,1} \\ \hat{s}^x_{0,2} & \hat{s}^y_{0,2} & \hat{s}^z_{0,2} \\ \hat{s}^x_{0,3} & \hat{s}^y_{0,3} & \hat{s}^z_{0,3} \end{pmatrix} \begin{pmatrix} \overrightarrow{\mathscr{E}}_{0_x} \\ \overrightarrow{\mathscr{E}}_{0_y} \\ \overrightarrow{\mathscr{E}}_{0_z} \end{pmatrix} = \begin{pmatrix} E_{0,1} \\ E_{0,2} \\ E_{0,3} \end{pmatrix} \tag{20}$$

or

$$S_0 \times \overrightarrow{\mathscr{E}_0} = \begin{pmatrix} E_{0,1} & E_{0,2} & E_{0,3} \end{pmatrix}^T \tag{21}$$

so that the computed field is then:

$$\overrightarrow{\mathscr{E}_0} = S_0^{-1} \times \begin{pmatrix} E_{0,1} & E_{0,2} & E_{0,3} \end{pmatrix}^T \tag{22}$$

It should be noted that each row in (20) may swap the order of indexes for each edge, depending on the node ordering of each edge stored in the simulator. This is correct as long as the ordering between the unit vector and the scalar field in (19) are consistent since $E_{i,j} = -E_{j,i}$ and $\hat{s}_{i,j} = -\hat{s}_{j,i}$.

The derivatives with respect to the node potentials are then

$$\frac{\partial \overrightarrow{\mathscr{E}_0}}{\partial \psi_0} = S_0^{-1} \times \begin{pmatrix} \frac{\partial E_{0,1}}{\partial \psi_0} & \frac{\partial E_{0,2}}{\partial \psi_0} & \frac{\partial E_{0,3}}{\partial \psi_0} \end{pmatrix}^T \tag{23}$$

$$\frac{\partial \overrightarrow{\mathscr{E}_0}}{\partial \psi_1} = S_0^{-1} \times \begin{pmatrix} \frac{\partial E_{0,1}}{\partial \psi_1} & 0 & 0 \end{pmatrix}^T \tag{24}$$

$$\frac{\partial \overrightarrow{\mathscr{E}_0}}{\partial \psi_2} = S_0^{-1} \times \begin{pmatrix} 0 & \frac{\partial E_{0,2}}{\partial \psi_2} & 0 \end{pmatrix}^T \tag{25}$$

$$\frac{\partial \overrightarrow{\mathscr{E}_0}}{\partial \psi_3} = S_0^{-1} \times \begin{pmatrix} 0 & 0 & \frac{\partial E_{0,3}}{\partial \psi_3} \end{pmatrix}^T \tag{26}$$

In 3D, we treat the field for the edge between nodes $\langle 0,1 \rangle$ using an average:

$$\overrightarrow{\mathscr{E}_{0,1}} = 0.5 \left(\overrightarrow{\mathscr{E}_0} + \overrightarrow{\mathscr{E}_1} \right) \tag{27}$$

where $\overrightarrow{\mathscr{E}_1}$ is calculated by considering node 1 as the node of interest in (20)–(26).

$$\overrightarrow{\mathscr{E}_1} = S_1^{-1} \times \begin{pmatrix} E_{1,0} & E_{1,2} & E_{1,3} \end{pmatrix}^T \tag{28}$$

The derivative with respect to the node potential is then:

$$\frac{\partial \overrightarrow{\mathscr{E}_{0,1}}}{\partial \psi_k} = 0.5 \left(\frac{\partial \overrightarrow{\mathscr{E}_0}}{\partial \psi_k} + \frac{\partial \overrightarrow{\mathscr{E}_1}}{\partial \psi_k} \right) \tag{29}$$

where k refers to one of the 4 nodes on the element. We assume that the average employed in (27)–(29) is appropriate for creating vectors from edge based models. It can be shown for electric field that $\overrightarrow{\mathscr{E}_0} = \overrightarrow{\mathscr{E}_1}$. In a case like current density, which do not have such a property, then (29) appears to be an acceptable average.

In 2D, we use an edge average from [10], [15], based on the perpendicular bisectors of the triangular elements.

$$\overrightarrow{\mathscr{E}_{0,1}} = d^t_{1,0,2} \cdot \overrightarrow{\mathscr{E}_0} + d^t_{0,1,2} \cdot \overrightarrow{\mathscr{E}_1} \tag{30}$$

where

$$d^t_{i,j,k} = \frac{\texttt{ElementEdgeCouple}^t_{i,k}}{\texttt{ElementEdgeCouple}^t_{i,k} + \texttt{ElementEdgeCouple}^t_{j,k}} \tag{31}$$

so that the averaging is weighted, based on the open angle of the edge pairs associated with each node. The simulator also provides the ability to customize the weighting of $\overrightarrow{\mathscr{E}_0}$ and $\overrightarrow{\mathscr{E}_1}$ in (27) and (30), so that the average may be problem specific.

Once the vector fields are attained, the vector components are used to calculate the components for the element edge assembly in (9)–(12). For example, the electric field in the direction of current flow can be calculated from [10]:

$$E^{\parallel}_{i,j} = \frac{\overrightarrow{J}_{i,j} \cdot \overrightarrow{\mathscr{E}_{i,j}}}{\left| \overrightarrow{J}_{i,j} \right|} \tag{32}$$

This scalar value can evaluated in the model and added to the matrix and RHS vector. Scalar models defined as an edge or element edge model, can also be used in calculations with the vector field components. This will be described in Section II-C, where the approach is applied to the polarization vector.

2) Computational Cost: In practice, the S_0 term in (21) is calculated once for each node of a tetrahedral element and is assigned as S_0 or S_1, based on the element edge of interest. The factorization of S_0 is an $O(n^3)$ operation where $n = 3$ in 3D and $n = 2$ in 2D). Since only geometric information is stored, the factorization is only done once at the beginning of the simulation, and is shared across all model equations and derivatives across all iterations of the simulation. The number of scalar evaluations for edge quantities for models like (17)–(18) would be 1 per edge in the simulation mesh. The cost of the field evaluation, (22), at each iteration is $O(n^2)$ for each element node, and the result is averaged along each connected element edge.

One of the advantages of this approach is the ability to take advantage of high performance math routines, so that the matrix factorization and solution can be done in an efficient manner [22].

In a control volume approach, on an unstructured mesh, the field would be averaged over an arbitrary number of edges

connected to the node of interest. The cost of evaluating the effects of the derivatives would depend on the algorithm in use, but is sometimes limited strictly to nodes on the edge quantity being evaluated. If the derivatives are incomplete, this could lead to additional solver iterations, or possible non convergence.

3) Limitations on Element Type: In our approach, we have restricted the method to triangular elements in 2D and tetrahedral elements in 3D. The original finite box methods restricted themselves to box and cube elements [1], and were later expanded [2]. While it is possible that other element types may be compatible with our method, this is the subject of further investigation.

It is necessary that the tetrahedra are Delaunay, where the center of the circumsphere is inside the element, so that the volume and surface integrals inside the simulator are calculated correctly [23]. While there are methods to accommodate non-Delaunay elements, their presence often leads to less accurate results and convergence difficulties [2].

4) Derivative Accuracy: The best convergence behavior is dependent on having accurate derivatives for the simulation models [1]. In the first simulation example (Section III-A), we compare the convergence behavior when having the full set of element derivatives versus having only a partial set of derivatives available, as is the case with many GPDE simulators [4]. Accurate derivatives are also important for methods using linearization around the device solution, such as small-signal ac [24], and impedance field methods for noise analysis [25] and random dopant fluctuations [26].

C. Generalized PDE Approach

In DEVSIM, models are implemented as symbolic expressions, and they are assembled and solved [18], [20]. Symbolic differentiation is employed to get the required derivatives. The simulator accepts equations in the form of:

Node models on each node (Section II-A.2)
Edge models on each edge (Section II-A.3)
Element edge models on each element edge (Section II-A.4)

The GPDE equations are input by calling commands from a Python script [27]. The simulator is a module loaded by the interpreter and it is written in C++, and utilizes third party libraries for the direct solver and dense matrix operations [18].

In this section, we discuss how to the EEB method is used for modeling polarization effects. A ferroelectric model may be implemented directly, using scripting. For a ferroelectric insulator, the equation is:

$$\nabla \cdot \left(\varepsilon \vec{\mathscr{E}} + \vec{P} \right) = 0 \tag{33}$$

where $\vec{P}$ is the polarization.

We first start with the edge model for electric field and its derivatives

```
EF,  EF:psi@n0,  EF:psi@n1
```

where `EF` is evaluated using an expression for (17) and the rest are calculated using (18) for nodes $\langle 0, 1 \rangle$, respectively. The user defines the model and its derivatives using the symbolic differentiation engine in the simulator.

From the edge model and its derivatives, the `element_from_edge_model` command is used to create element edge models as:

```
EF_x, EF_y, EF_z
```

for each vector component using (27) and their derivatives using (29)

```
EF_x:psi@en0, EF_y:psi@en0 EF_z:psi@en0
EF_x:psi@en1, EF_y:psi@en1 EF_z:psi@en1
EF_x:psi@en2, EF_y:psi@en2 EF_z:psi@en2
EF_x:psi@en3, EF_y:psi@en3 EF_z:psi@en3
```

The first term represents the derivative of `EF_x` with respect to the potential at node 0, and the other terms follow in a similar fashion.

During evaluation, indexes $\langle$`en0, en1`$\rangle$ reference nodes on the element edge and $\langle$`en2, en3`$\rangle$ reference the other element nodes on the tetrahedron. The simulator tracks the element edge indexes on a per element edge basis, so that the same PDE expression will work for each element edge being considered.

The simulator then evaluates the symbolic expression for (33) using

```
(PX+Permittivity*EF_x)*unitx +
(PY+Permittivity*EF_y)*unity +
(PZ+Permittivity*EF_z)*unitz
```

where `unitx`, `unity`, and `unitz` are the unit vector components along the element edge and `Permittivity` is a parameter. The terms `PX`, `PY`, and `PZ` are components which computed in using expressions involving the current and previous solutions for the electric field. We utilized a widely accepted empirical function of hyperbolic tangent to simulate the hysteresis loop [11], [12], [28]. Scripting was used to generate these terms for all of the x, y, and z components and the derivatives of the polarization vector and electric fields.

III. Simulation Examples

A. Ferrocapacitor Simulation

We used TetGen [29] to generate a mesh of a cube and boundary conditions on the top and bottom. TetGen performs a constrained Delaunay tetrahedralization (CDT), which does not guarantee that all elements meet the Delaunay criterion. A script was written to read the mesh and mark the non-Delaunay elements for refinement. Using this background mesh approach, it was possible to achieve a Delaunay mesh.

The mesh and physics are loaded via scripts into the simulator. Fig. 6 shows the compensated charge versus the applied voltage for a ferroelectric capacitor. From the initial zero bias condition, the voltage is ramped over the range between ± 15 V. The hysteresis in the charge due to the polarization is apparent in the figure.

As a test of the accuracy of our approach, we repeated the simulations with derivative terms with respect to $\langle$`en2, en3`$\rangle$ set to 0. This emulates the condition in GPDE simulators which evaluate vector effects over all adjacent nodes, but may restrict their derivative information on edges [4], [14].

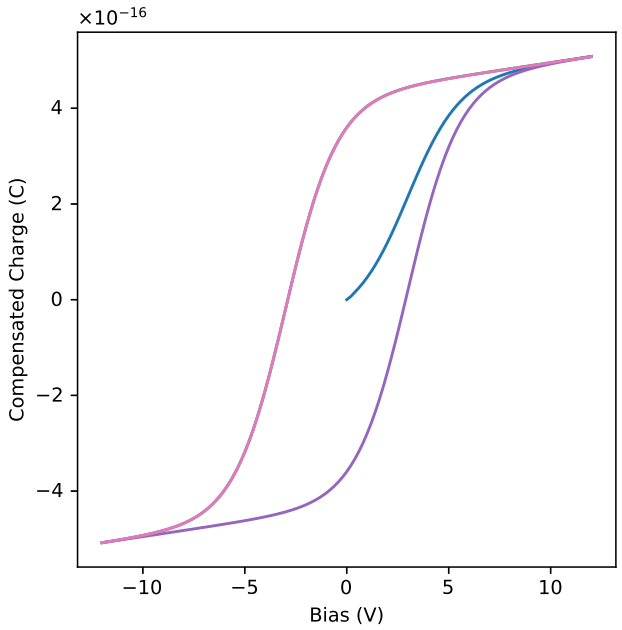

Fig. 6. The ferro capacitor 3D simulation.

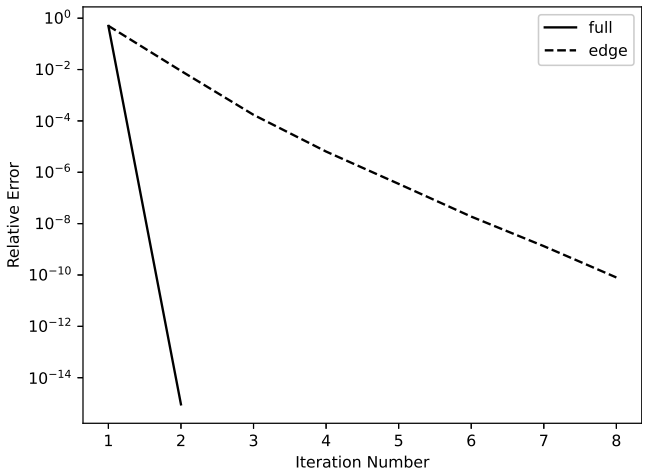

Fig. 7. The relative error versus iteration number. The solid line is for derivatives for all nodes on the element. The dashed line is when only nodes on the edge are considered.

Fig. 7 shows relative error versus iteration number for one of the bias points. With all derivatives accounted for, it takes 2 iterations to get a relative error near 10^{-15}, which is the limit of accuracy for double precision floating point arithmetic. This number of iterations for this equation system is expected, since the equation derivatives are constant with respect to bias. When the off edge derivatives are neglected, the iteration count is higher to to attain a relative error within 10^{-10}.

B. FeFET Simulation

We simulated a 2D p-type FeFET with a top contact/bottom gate architecture. Ohmic contacts were set at the source and drain electrodes. A triangular mesh was generated using Gmsh [30]. Extended precision math mode was enabled in the equation assembly to prevent numerical noise in the off

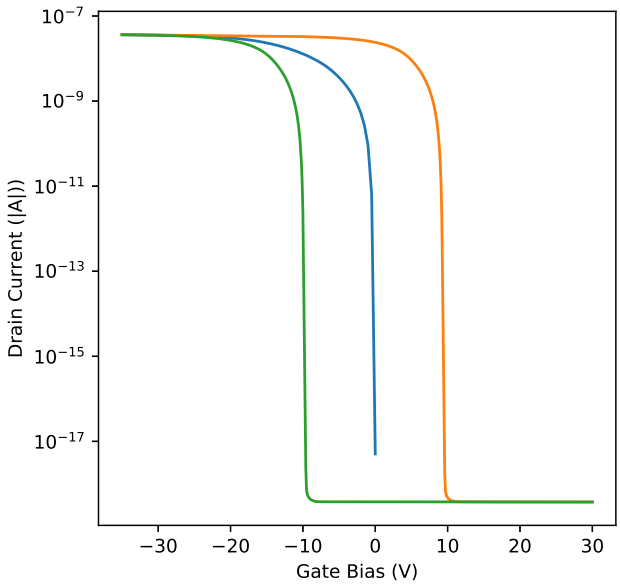

Fig. 8. The p-type FeFET 2D simulation.

state current results. Fig. 8 shows the drain current versus gate voltage. The simulation details and an analysis of the switching processes of FeFET with respect to its structure will be presented in a separate publication [31].

IV. Conclusion

In this paper, we presented an approach for modeling vector effects on element edges. The algorithm evaluates models and their derivatives in both 2D and 3D. We describe how the approach works within a GPDE simulator. Simulation examples for the ferroelectric effect were presented.

References

[1] S. Selberherr, *Analysis and simulation of semiconductor devices*. NY: Springer-Verlag, 1984.

[2] M. R. Pinto, "Comprehensive semiconductor device simulation for silicon ULSI," Ph.D. dissertation, Stanford University, 1990.

[3] D. L. Scharfetter and H. K. Gummel, "Large-signal analysis of a silicon Read diode oscillator," *IEEE Trans Electron Devices*, vol. ED-16, no. 1, pp. 64–77, Jan. 1969, 10.1109/T-ED.1969.16566.

[4] K. M. Kramer and W. N. G. Hitchon, *Semiconductor Devices: A Simulation Approach*. Up Saddle River, NJ: Prentice Hall PTR, 1997.

[5] D. J. Cummings, M. E. Law, S. Cea, and T. Linton, "Comparison of discretization methods for device simulation," in *2009 Int Conf SISPAD*, 2009, pp. 1–4, DOI: 10.1109/SISPAD.2009.5290236.

[6] E. Patrick, N. Rowsey, and M. E. Law, "Total dose radiation damage: A simulation framework," *IEEE Trans Nucl Sci*, vol. 62, no. 4, pp. 1650–1657, 2015, DOI: 10.1109/TNS.2015.2425226.

[7] P. Bochev, K. Peterson, and X. Gao, "A new control volume finite element method for the stable and accurate solution of the drift-diffusion equations on general unstructured grids," *Comput Methods Appl Mech Eng*, vol. 254, pp. 126–145, 02 2013, DOI: 10.1016/j.cma.2012.10.009.

[8] COMSOL, "Analyze semiconductor devices at the fundamental level with the semiconductor module," last accessed 09/15/2020. [Online]. Available: https://www.comsol.com/semiconductor-module

[9] L. Chen and H. Bagci, "Steady-state simulation of semiconductor devices using discontinuous Galerkin methods," *IEEE Access*, vol. 8, pp. 16 203–16 215, 2020, DOI: 10.1109/ACCESS.2020.2967125.

[10] S. E. Laux and R. G. Byrnes, "Semiconductor device simulation using generalized mobility models," *IBM J. Res. Dev.*, vol. 29, no. 3, pp. 289–301, May 1985, DOI: 10.1147/rd.293.0289.

[11] S. L. Miller, R. D. Nasby, J. R. Schwank, M. S. Rodgers, and P. V. Dressendorfer, "Device modeling of ferroelectric capacitors," *J Appl Phys*, vol. 68, no. 12, pp. 6463–6471, 1990, DOI: 10.1063/1.346845.
[12] M. Ghittorelli, T. Lenz, H. Sharifi Dehsari, D. Zhao, K. Asadi, P. W. M. Blom, Z. M. Kovács-Vajna, D. M. de Leeuw, and F. Torricelli, "Quantum tunnelling and charge accumulation in organic ferroelectric memory diodes," *Nat Commun*, vol. 8, no. 1, p. 15741, 2017, DOI: 10.1038/ncomms15841.
[13] Z. Yu, D. W. Yergeau, and R. W. Dutton, "Algorithm for evaluating nodal vector quantities in device simulation and its applications to modeling quantum mechanical effects in sub-50nm MOSFETs," in *Int Symp VLSI Tech, Syst App*, 2003, pp. 261–264, DOI: 10.1109/VTSA.2003.1252603.
[14] T. Ikegami, K. Fukuda, and J. Hattori, "Implementation of automatic differentiation to Python-based semiconductor device simulator," in *2019 Int Conf SISPAD*, 2019, pp. 1–4, DOI: 10.1109/SISPAD.2019.8870377.
[15] S. E. Laux and B. M. Grossman, "A general control-volume formulation for modeling impact ionization in semiconductor transport," *IEEE Trans Electron Devices*, vol. 32, no. 10, pp. 2076–2082, 1985, DOI: 10.1109/T-ED.1985.22241.
[16] O. Triebl and T. Grasser, "Investigation of vector discretization schemes for box volume methods," in *Tech Proc 2007 NSTI Nanotechnology Conf and Trade Show, Vol 3*, 2007, pp. 61–64.
[17] Y. Liu and R. Dutton, "Nano-scale device simulations using PROPHET-Part II: PDE systems," Jan. 2006. [Online]. Available: http://www.nanohub.org/resources/975/
[18] J. E. Sanchez, "Semiconductor device simulation using DEVSIM," in *Open Source TCAD/EDA for Compact Modeling*, W. Grabinski and D. Tomaszewski, Eds. New York: Springer, 2021.
[19] DEVSIM LLC, "DEVSIM TCAD semiconductor device simulator," Available: https://devsim.org.
[20] ——, "DEVSIM User Guide," Available: https://devsim.net.
[21] A. Wettstein, O. Penzin, and E. Lyumkis, "Integration of the density gradient model into a general purpose device simulator," *VLSI Design*, vol. 15, no. 4, pp. 751–759, 2002, DOI: 10.1080/1065514021000012363.
[22] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed. Philadelphia, PA: SIAM, 1999.
[23] B. Haindl, R. Kosik, P. Fleischmann, and S. Selberherr, "Comparison of finite element and finite box discretization for three-dimensional diffusion modeling using AMIGOS," in *1999 Int Conf SISPAD*, 1999, pp. 131–134, DOI: 10.1109/SISPAD.1999.799278.
[24] S. E. Laux, "Techniques for small-signal analysis of semiconductor devices," *IEEE Trans Electron Devices*, vol. 32, no. 10, pp. 2028–2037, 1985, DOI: 10.1109/T-ED.1985.22235.
[25] F. Bonani, G. Ghione, M. R. Pinto, and R. K. Smith, "An efficient approach to noise analysis through multidimensional physics-based models," *IEEE Trans Electron Devices*, vol. 45, no. 1, pp. 261–269, 1998, DOI: 10.1109/16.658840.
[26] K. El Sayed, A. Wettstein, S. D. Simeonov, E. Lyumkis, and B. Polsky, "Investigation of the statistical variability of static noise margins of sram cells using the statistical impedance field method," *IEEE Transactions on Electron Devices*, vol. 59, no. 6, pp. 1738–1744, 2012, DOI: 10.1109/TED.2012.2189860.
[27] Python Software Foundation, "Python," https://python.org.
[28] S. L. Miller and P. J. McWhorter, "Physics of the ferroelectric nonvolatile memory field effect transistor," *J Appl Phys*, vol. 72, no. 12, pp. 5999–6010, 1992, DOI: 10.1063/1.351910.
[29] H. Si, "TetGen, a Delaunay-based quality tetrahedral mesh generator," *ACM Trans. Math. Softw.*, vol. 41, no. 2, Feb. 2015, DOI: 10.1145/2629697.
[30] C. Geuzaine and J.-F. Remacle, "Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities," *Int J Numer Methods Eng*, vol. 79, pp. 1309–1331, 2009, DOI: 10.1002/nme.2579.
[31] Q. Chen, D. Lin, Q. Wang, J. Yang, J. Sanchez, and G. Zhu, "An investigation of the switching processes of ferroelectric field-effect transistors using 2D simulation," *submitted for publication*.