RESEARCH ARTICLE | NOVEMBER 13 2023

Physically constrained learning of MOS capacitor electrostatics

Tejas Govind Indani ; Kunal Narayan Chaudhury ; Sirsha Guha ; Santanu Mahapatra

J. Appl. Phys. 134, 184903 (2023)
https://doi.org/10.1063/5.0168104

Physically constrained learning of MOS capacitor electrostatics

Cite as: J. Appl. Phys. **134**, 184903 (2023); doi: 10.1063/5.0168104
Submitted: 17 July 2023 · Accepted: 25 October 2023 ·
Published Online: 13 November 2023

Tejas Govind Indani,[1] Kunal Narayan Chaudhury,[1,a)] Sirsha Guha,[2] and Santanu Mahapatra[2,a)]

AFFILIATIONS

[1]Department of Electrical Engineering, Indian Institute of Science Bangalore, Bangalore 560012, India
[2]Nano-Scale Device Research Laboratory, Department of Electronic Systems Engineering, Indian Institute of Science Bangalore, Bangalore 560012, India

[a)]**Authors to whom correspondence should be addressed:** kunal@iisc.ac.in and santanu@iisc.ac.in

ABSTRACT

In recent years, neural networks have achieved phenomenal success across a wide range of applications. They have also proven useful for solving differential equations. The focus of this work is on the Poisson–Boltzmann equation (PBE) that governs the electrostatics of a metal–oxide–semiconductor capacitor. We were motivated by the question of whether a neural network can effectively learn the solution of PBE using the methodology pioneered by Lagaris *et al.* [IEEE Trans. Neural Netw. 9 (1998)]. In this method, a neural network is used to generate a set of trial solutions that adhere to the boundary conditions, which are then optimized using the governing equation. However, the challenge with this method is the lack of a generic procedure for creating trial solutions for intricate boundary conditions. We introduce a novel method for generating trial solutions that adhere to the Robin and Dirichlet boundary conditions associated with the PBE. Remarkably, by optimizing the network parameters, we can learn an optimal trial solution that accurately captures essential physical insights, such as the depletion width, the threshold voltage, and the inversion charge. Furthermore, we show that our functional solution can extend beyond the sampling domain.

I. INTRODUCTION

Mathematics is known as the language of physics, and physicists have developed their own dialect over time. Many physical systems have been modeled using differential equations, the origins of which can be traced back to Newton and his second law of motion. A differential equation (DE) involves a function of one or more independent variables and its derivatives. The function typically represents a physical quantity, the derivatives signify rates of change with respect to the independent variables, and the DE describes a relationship between them. For most equations, it is challenging to find analytical solutions, and we have to solve them on a computer. Several numerical methods, such as shooting methods, a finite difference method, a finite element method, and splines,[1,2] have been proposed and studied over the last few decades. Many open-source computing platforms, such as FEniCSx (fenicsproject.org), have been developed that can solve DEs with minimum effort.

Recent years have seen the emergence of a neural network (NN)[3–5] that has impacted diverse science and engineering disciplines. NNs can represent complex functions and are not plagued by the curse of dimensionality.[6–8] In particular, they offer the ability to overcome difficulties, such as dependence on discretization, faced by traditional techniques, such as a finite element and splines. The fundamental challenge with NNs is that their training is slow and computationally demanding. However, recent advances in computing and optimization techniques have helped overcome this obstacle.[3,9,10] On the implementation side, user-friendly software, such as PyTorch[11] and TensorFlow,[12] allows users to seamlessly construct and train neural networks.[13,14]

The powerful functional learning capability of NNs[15–17] has been harnessed for solving DEs.[18–23] The present work was motivated by the approach of Lagaris *et al.*[24] They pioneered the use of NN-based trial solutions for solving DEs, which exactly satisfy the boundary conditions (BCs). The governing equation is used to find a trial solution that best "fits" the DE in a precise sense. The challenge with this approach is that there is no general mechanism for constructing trial solutions, and conceiving trials for nonlinear differential equations can be tricky. A more recent approach, called PINN (Physics-Informed Neural Networks[25]), also uses a neural network to solve DEs. The difference with Ref. 24 is that the BCs are not hard-constrained in PINN; instead, they are combined as a penalty into a loss function along with the DE. While PINNs solve the problem of having to handcraft trial solutions, making them more versatile, the flip side is that there is no guarantee that they satisfy the BCs exactly. In either method, the network parameters (i.e., the weights) are optimized using backpropagation to fit the DE and the BCs (in the case of PINN). More recently, generative learning[26] has been used to solve DEs.[27] However, it is well-known that such generative models are unstable and difficult to train.[28] Though exciting, these efforts have primarily focused on a few standard DEs (Burgers equations, Schrödinger equation, etc.). An open question is whether they can provide accurate solutions to sophisticated and dynamic physical systems.

In this work, we propose to use neural networks to solve the Poisson–Boltzmann equation (PBE) governing the electrostatic behavior of metal–oxide–semiconductor (MOS) capacitors.[29] More specifically, we show that although the PBE is highly nonlinear, it is nonetheless possible to use NNs to construct trial solutions that exactly satisfy the Robin and Dirichlet boundary conditions of the PBE. To the best of our knowledge, this is the first work to explore the possibility of using the method of Ref. 24 to solve the PBE. In our approach, the loss function for a given trial solution is formulated using the residual of the PBE. To construct the loss from the functional trial solution, we need to sample the physical domain, and the number of samples determines the complexity of optimizing the trials. In this regard, we devise a physics-based sampling scheme and introduce device parameters stochastically into the model by virtue of which the model attains exceptional accuracy. We demonstrate that our solution accurately discovers the dependence of threshold voltage on oxide thickness and doping concentration outside the sampling domain, in addition to interpreting the accumulation, depletion, and inversion mechanisms.

II. RESULTS

A. Poisson–Boltzmann equation for MOS

MOS capacitors are the fundamental building blocks of present CMOS (complementary metal–oxide–semiconductor) technology. The schematic of an MOS capacitor is shown in Fig. 1, where an insulating silicon dioxide layer is sandwiched between a metal gate and a semiconducting bulk silicon. We consider the silicon substrate to be p-type, with N_A being the acceptor concentration. The work function of the gate material is chosen so that the flatband voltage always becomes zero. A nonzero flatband voltage shifts all the characteristics on the gate-voltage scale by adding a DC offset, and thus, it does not have any impact on our proposed modeling methodology.

The PBE concerns the electrostatic potential Ψ, which is an important quantity representing a semiconductor's intrinsic energy level. In accordance with the Boltzmann distribution, the charge density $\rho(y)$ at any point y inside the semiconductor is given by[29]

$$\rho(y) = q(p(y) - n(y) - N_A), \tag{1}$$

where

$$n(y) = n_0 e^{\Psi(y)/\Phi_t}, \quad p(y) = p_0 e^{-\Psi(y)/\Phi_t} \tag{2}$$

are the electron and hole concentrations, n_0 and p_0 are their respective values deep inside the bulk, N_A is the concentration of acceptors, q is the elementary charge, and Φ_t is the thermal voltage. On the other hand, the Poisson equation gives us the following relation:

$$\Psi''(y) = -\frac{\rho(y)}{\varepsilon_{si}}.$$

Combining the Poisson and Boltzmann equations, we get the Poisson–Boltzmann equation

$$\Psi''(y) = -\frac{qN_A}{\varepsilon_{si}}\left[e^{-\frac{\Psi(y)}{\Phi_t}} - 1 - e^{-\frac{2\Phi_B}{\Phi_t}}\left(e^{\frac{\Psi(y)}{\Phi_t}} - 1\right)\right], \tag{3}$$

where $\Phi_B = \Phi_t \ln(N_A/n_i)$ is known as bulk potential and $n_i = 10^{16}\,\mathrm{m}^{-3}$ is the intrinsic carrier concentration of silicon.[29]

With the application of the gate voltage V_G, the electrostatic potential, and, consequently, the concentration of electrons and holes changes at every point inside the semiconductor. The precise relation between V_G and Ψ comes from (3) and the boundary condition

$$\Psi'(0) = -\frac{C_{ox}}{\varepsilon_{si}}(V_G - \Psi(0)). \tag{4}$$

Also, we have the natural boundary condition

$$\Psi(t_{si}) = 0. \tag{5}$$

In (4) and (5), t_{si} and t_{ox} are the thicknesses of the semiconductor and the oxide, ε_{ox} and ε_{si} are the permittivities of the oxide and the semiconductor, and $C_{ox} = \varepsilon_{ox}/t_{ox}$ is the oxide capacitance per unit area. The Dirichlet BC (5) represents charge neutrality deep inside the bulk, while the Robin BC (4) originates from the continuity of the electric field at the semiconductor–oxide interface.

In general, the PBE does not have an analytical solution. However, the surface potential $\Psi_s = \Psi(0)$ has an implicit solution given by[29]

$$(V_G - \Psi_s)^2 = \frac{2q\varepsilon_{si}N_A}{C_{ox}^2}\left[\Phi_t e^{-\frac{\Psi_s}{\Phi_t}} + \Psi_s - \Phi_t + e^{\frac{-2\Phi_B}{\Phi_t}}\left(\Phi_t e^{\frac{\Psi_s}{\Phi_t}} - \Psi_s - \Phi_t\right)\right]. \tag{6}$$

This is called the surface potential equation (SPE) and is used as a backbone in all industry-standard, surface-potential-based compact models for bulk-silicon-based MOSFET technology.[30,31]

To solve the PBE numerically, we used the solve_bvp package from scipy.integrate.[32] The normalized solution $\Psi(y)/\Phi_B$ for $t_{ox} = 1\,\mathrm{nm}$ and $N_A = 10^{24}\,\mathrm{m}^{-3}$ is depicted in Fig. 1(b). For $V_G > 0$, Ψ increases monotonically with V_G until Ψ_s reaches the value $2\Phi_B$. Eventually, holes start depleting and electrons accumulate at the surface as suggested by (2). However, once $\Psi_s = 2\Phi_B$, electron concentration at the surface becomes equal to hole concentration deep inside the bulk; we call this the onset of

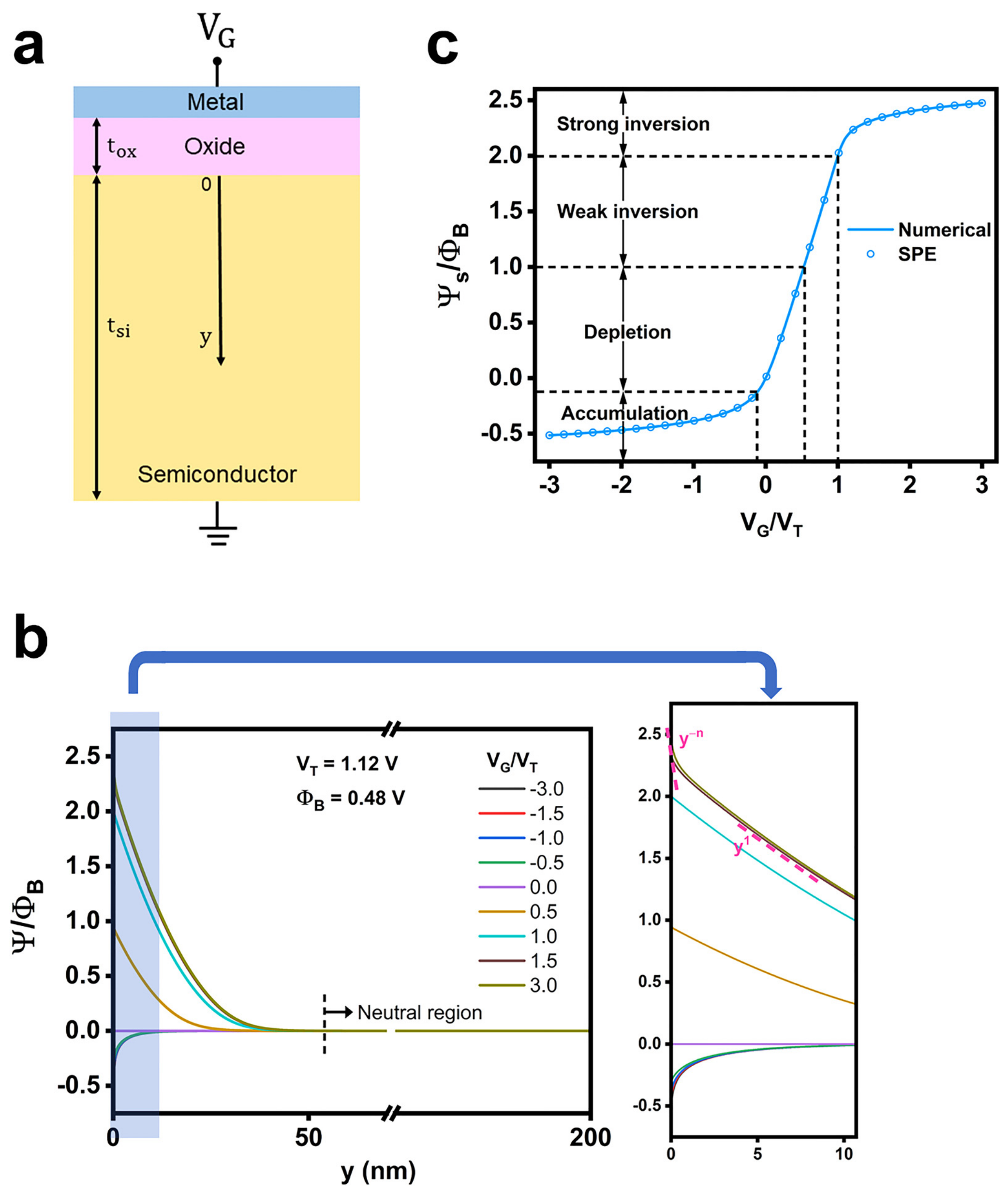

FIG. 1. Anatomy of a MOS capacitor. (a) Schematic representation, where V_G, t_{ox}, and t_{si} denote the applied gate voltage, oxide thickness, and depth of the semiconductor from the surface. (b) Normalized potential distribution inside the semiconductor obtained from the numerical solution of the PBE (the region close to the surface is zoomed on the right). (c) Comparison of the normalized surface potentials obtained from the numerical solution of PBE and SPE. It also shows different regions of operation of a MOS capacitor based on Ψ_s. For all calculations, $t_{ox} = 1$ nm and $N_A = 10^{24}$ m^{-3}.

strong inversion. The gate voltage for which $\Psi_s = 2\Phi_B$ is known as the threshold voltage V_T. This can be analytically derived from (6) with few approximations and is given by

$$V_T = 2\Phi_B + \frac{\sqrt{2q\varepsilon_{si}N_A}}{C_{ox}}\sqrt{2\Phi_B}. \tag{7}$$

When V_G crosses V_T, the inversion layer forms beneath the oxide–semiconductor interface, and as we can see in Fig. 1(b), Ψ becomes highly non-linear in this regime. Similarly, a negative V_G leads to the accumulation of holes at the surface, and we see that $\Psi(y)$ keeps increasing with V_G.

In Fig. 1(b), notice how remarkably $\Psi(y)$ changes with y and V_G:

- For $V_G > V_T$, $e^{\Psi(y)} \approx e^{y^{-n}}$ near the surface, $e^{\Psi(y)} \approx 1$ deep inside the bulk, and $e^{\Psi(y)} \approx e^y$ in between (n is a positive number greater than 1).
- For $0 < V_G < V_T$, $e^{\Psi(y)} \approx 1$ deep inside the bulk and $e^{\Psi(y)} \approx e^y$ elsewhere.
- For $V_G < 0$, $e^{\Psi(y)} \approx e^{y^n}$ near the surface and $e^{\Psi(y)} \approx 1$ elsewhere.

This highly dynamic nature of the solution of the PBE makes the learning problem challenging as well as interesting.

The solution of SPE (6) is compared with the solution from the solve_bvp package in Fig. 1(c). Their excellent agreement confirms the accuracy of the numerical method. It is worth noting that SPE is obtained by setting $t_{si} = \infty$ in (5). However, for the numerical solution of PBE, we took $t_{si} = 200$ nm. We found that with the further increase of t_{si}, the solution for Ψ_s does not change. However, this increases the optimization time of our neural network model (see Sec. II D). From Fig. 1(c), we see that the surface potential changes linearly with V_G for $0 < V_G < V_T$. This is because the exponential terms in (6) become insignificant in this regime, and the Poisson equation effectively becomes the Laplace equation. In the accumulation ($V_G < 0$) and the strong inversion ($V_G > V_T$) regimes, the exponential terms dominate the other terms, and hence, Ψ_s becomes a logarithmic function of V_G. As a result, Ψ_s saturates at $\approx(2\Phi_B + 3\Phi_t)$ and $\approx -3\Phi_t$ in the strong inversion and accumulation regions.

B. Prior works

As mentioned earlier, NNs have been used for solving DEs.[18–25] In particular, our approach is motivated by Refs. 24 and 25. We will discuss these methods and explain how they can be used for solving (3) along with the boundary conditions (4) and (5). To motivate our contribution, we will specifically explain the problem faced in directly applying the method in Ref. 24 to our problem.

In the last few years, PINNs[25] have been used extensively to solve various differential equations.[33] In this approach, the solution of the differential equation is modeled using a neural network. The weights of the network are determined by optimizing a loss function derived from the differential equation and the boundary conditions.

We can use PINN to solve the present PBE as follows. Consider a standard feedforward neural network $\mathcal{N}(y;\theta)$ with input y and parameters θ (see Fig. 2). To keep the exposition simple, we consider just one input at this point; we will later work with several input variables and internal parameters without any fundamental change in the derivation. Ideally, we wish to find θ such that $\mathcal{N}(y;\theta)$ satisfies (3)–(5). More precisely, consider the error function derived from (3),

$$E_1(y,\theta) = \mathcal{N}''(y;\theta) + \frac{qN_A}{\varepsilon_{si}}\left[e^{-\frac{\mathcal{N}(y;\theta)}{\Phi_t}} - 1 - e^{-\frac{2\Phi_B}{\Phi_t}}\left(e^{\frac{\mathcal{N}(y;\theta)}{\Phi_t}} - 1\right)\right]$$

and the error functions

$$E_2(\theta) = \mathcal{N}(t_{si};\theta) \text{ and}$$

$$E_3(\theta) = \mathcal{N}'(0;\theta) + \frac{C_{ox}}{\varepsilon_{si}}(V_G - \mathcal{N}(0;\theta))$$

derived from (4) and (5). In the above equations, the derivatives are with respect to y.

Ideally, we wish to find a parameter setting θ such that $E_2(\theta) = 0$ and $E_3(\theta) = 0$ and $E_1(y,\theta) = 0$ for all y belonging to the domain of interest. However, this can be mathematically infeasible. A standard workaround is to use a regression framework. Specifically, we can sample points $y_1, \ldots, y_K$ from the domain of

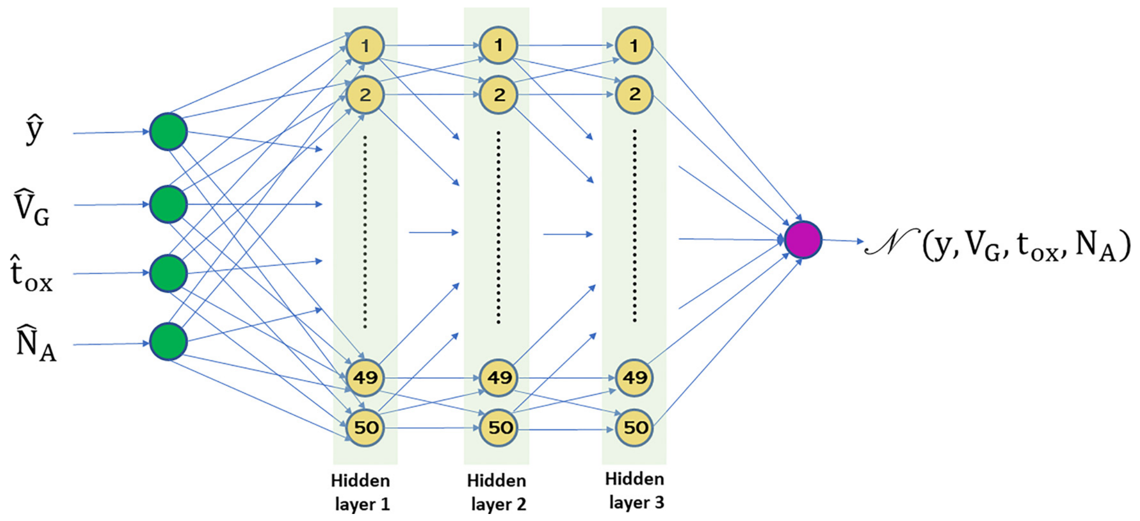

FIG. 2. Architecture of the feedforward neural network used for constructing the trial solution ψ of the Poisson–Boltzmann equation (see the main text for details). We use normalized inputs to the network; e.g., we normalize the input y to $\hat{y} = (y - \mu)/\sigma$, where the mean μ and the standard deviation σ are computed over the set of sampled values of y. We used 50 neurons per layer, which we found to be optimal for our problem (also see Fig. S6).

interest and formulate the loss function as

$$\mathscr{L}(\theta) = \frac{1}{K}\sum_{k=1}^{K} E_1(y_i, \theta)^2 + \lambda_1 E_2(\theta)^2 + \lambda_2 E_3(\theta)^2, \tag{8}$$

where $\lambda_1, \lambda_2 > 0$ are used to combine the loss components. The optimal parameter setting θ^* is obtained by minimizing $\mathscr{L}(\theta)$ over θ. This is precisely the idea behind PINN.[25]

We can minimize $\mathscr{L}(\theta)$ using standard NN training. In particular, if θ are the weights of the neural network, then $\mathscr{L}(\theta)$ is differentiable in θ and, hence, can be optimized using gradient descent and backpropagation.[11,12] PINN is quite versatile and can, in principle, be used to tackle any DE. On the flip side, incorporating boundary conditions as a penalty in the loss function means that they do not necessarily have to be met exactly. More precisely, if θ^* is the minimizer of (8), then we cannot guarantee that $E_2(\theta^*) = 0$ and $E_3(\theta^*) = 0$. To do so, we need to enforce them as hard constraints in the optimization problem; i.e., we would have to solve the optimization problem

$$\min_{\theta} \frac{1}{K}\sum_{k=1}^{K} E_1(y_i, \theta)^2 \quad \text{s.t.} \quad E_2(\theta) = 0, \; E_3(\theta) = 0. \tag{9}$$

Unfortunately, this is a highly challenging problem. Indeed, due to the nonlinear nature of the neural network, it is difficult to characterize the feasible set $\{\theta : E_2(\theta) = 0, \; E_3(\theta) = 0\}$. We demonstrate in Fig. S1 that the PINN approach fails to learn the solution to PBE for the very same reason.

Lagaris *et al.*[24] proposed an ingenious approach to circumvent this problem, which we explain here using our PBE. Similar to PINN, we start with a neural network $\mathcal{N}(y, \theta)$. However, we do not use this to directly represent the solution of (3). Instead, we consider a trial solution of the form

$$\psi(y; \theta) = F(y, \mathcal{N}(y; \theta), \mathcal{N}'(y; \theta), \ldots), \tag{10}$$

where the function $F(y, u, v, \ldots)$ is such that the boundary conditions (3) and (5) are satisfied. Specifically, for any choice of θ, we must have

$$\psi(t_{si}; \theta) = 0, \qquad \psi'(0; \theta) = -\frac{C_{ox}}{\varepsilon_{si}}(V_G - \psi(0; \theta)). \tag{11}$$

It is not obvious if such an F exists, i.e., if we can construct trial solutions of the PBE in the first place. It was shown in Secs. III and IV of Ref. 24 how such trial solutions can be constructed for Dirichlet and Neumann BCs. For our PBE, while (5) is a simple Dirichlet BC, (4) is a Robin BC involving the function Ψ and its derivative at $y = 0$. We make the case that it is highly nontrivial to devise trial solutions for Robin BCs using the ideas discussed in Ref. 24.

C. Trial solution for PBE

The previous discussion brings us to the ingenuity of this work, namely, the construction of a trial solution of the form (10) that conforms to (5) and (4). The difficulty comes from the Robin BC, which is the form of

$$\psi'(0; \theta) + a\psi(0; \theta) + b = 0, \tag{12}$$

where $a = -C_{ox}/\varepsilon_{si}$ and $b = V_G C_{ox}/\varepsilon_{si}$. Since $a \neq 1$, a possible trial solution is

$$\psi(y; \theta) = (y - 1)(y^2\mathcal{N}(y; \theta) + b/(a - 1)) \tag{13}$$

corresponding to $F(y, u) = (y - 1)(y^2 u + b/(a - 1))$. It is not difficult to verify that (13) satisfies (5) and (12). Keeping in mind the exponential term in (3), we can alternatively consider the following trial solution:

$$\psi(y; \theta) = (y - 1)(y^2 e^{-\mathcal{N}(y;\theta)} + b(a - 1)^{-1}), \tag{14}$$

where $F(y, u) = (y - 1)(y^2 e^{-u} + b/(a - 1))$. Again, it can be verified that (14) satisfies (5) and (12).

The problem with (13) and (14) is that the value at $y = 0$ is pivoted in either case; namely, we have $\psi(0, \theta) = -b/(a - 1)$, irrespective of θ. In other words, the potential at the boundary $y = 0$ (and around it by continuity) is clamped and cannot be controlled using the network parameter θ. This limits the generalization capability of the trial solution. We note that this problem does not arise with the trial solutions in Ref. 24, where just Dirichlet and Neumann BCs are considered.

Considering the form of (12), we wish to come up with a trial solution that has both $\mathcal{N}(0; \theta)$ and $\mathcal{N}'(0; \theta)$. Moreover, considering the nature of the solution in Fig. 1, we wish to have an exponential dependence on $\mathcal{N}(y; \theta)$ as in (14). This will help in modeling the steep variation in the potential near the surface, as well as the saturation in the neutral region. Considering the above points, we propose a trial solution of the form

$$\psi(y; \theta) = G\big(y, \mathcal{N}(y; \theta), \mathcal{N}(0; \theta), \mathcal{N}'(0; \theta)\big), \tag{15}$$

where we define $G(y, u, v, w)$ to be

$$G(y, u, v, w) = V_G\left(\frac{\gamma}{1 + \gamma + t_{si}w}\right)\left(1 - \frac{y}{t_{si}}\right)e^{-(u-v)} \tag{16}$$

and $\gamma = C_{ox}t_{si}/\varepsilon_{si}$. Thus, the complete expression of the trial solution is

$$\psi(y; \theta) = V_G\left(\frac{\gamma}{1 + \gamma + t_{si}\mathcal{N}'(0; \theta)}\right)\left(1 - \frac{y}{t_{si}}\right) \cdot \exp\left(-(\mathcal{N}(y; \theta) - \mathcal{N}(0; \theta))\right). \tag{17}$$

Our trial solution depends not just on the output of the neural network $\mathcal{N}(y; \theta)$ as in (13) and (14), but also on $\mathcal{N}(0; \theta)$ and $\mathcal{N}'(0; \theta)$. The latter terms play an important role in the Robin BC (12). Indeed, it can be verified (see the Appendix) that (17) satisfies

(11). Importantly, note that

$$\psi(0;\theta) = V_G\left(\frac{\gamma}{1+\gamma+t_{si}\mathcal{N}'(0;\theta)}\right).$$

Thus, unlike the trial solutions (13) and (14), the potential at the boundary can be controlled using the network parameter. In (17), we considered a single input y to the model, assuming other quantities, such as V_G, t_{ox} and N_A, to be fixed. However, we want a more versatile model that can predict the potential function $\Psi(y)$ for different settings of V_G, t_{ox}, and N_A. This can be done simply by defining the input vector

$$\xi = (V_G, t_{ox}, N_A)$$

and constructing a neural network that takes both y and ξ as input and returns the predicted potential (see Fig. 2). Subsequently, we modify the specification of the trial function in (15) to

$$\psi(y,\xi;\theta) = G\big(y, \mathcal{N}(y,\xi;\theta), \mathcal{N}(0,\xi;\theta), \mathcal{N}'(0,\xi;\theta)\big),$$

where G is given by (16) and $\mathcal{N}'(0,\xi;\theta)$ denotes the derivative of $\mathcal{N}(y,\xi;\theta)$ with respect to y at 0. We will work with this trial solution in the rest of the paper.

D. Loss function and model optimization

In this section, we explain how the loss function is computed and how it is used to determine the optimal trial solution. The optimality in question is in an averaged sense, namely, with respect to different settings of ξ. Moreover, we also need to sample the y-domain as done in (8) and (9). The problem is that even if we consider just 100 samples for each variable, we would end up with $(100)^4 = 10^8$ samples; i.e., the loss function would have 10^8 terms. We will use the device physics to optimize the data volume. The precise sampling strategy is as follows:

1. We sample y according to the nonlinearity offered by $\Psi(y)$. In particular, we sample 500 points in $[0, 10]$ nm, 1000 points in $[10, 50]$ nm, 1000 points in $[50, 130]$ nm, and 500 points in $[130, 200]$ nm.
2. We take 29 samples of t_{ox} in the range $[0.8, 1.5]$ nm.
3. We take 41 uniform samples of N_A in the range $[0.1, 1]\times 10^{24}\,\mathrm{m}^{-3}$.
4. We choose V_G in the range $(-3V_T, 3V_T)$ for the fact that in the semiconductor technology roadmap, the supply voltage is kept three times that of the threshold voltage.[34] As V_T varies with N_A and t_{ox}, we take the largest V_T over all possible values of N_A and t_{ox} mentioned above. This turns out to be $1.2\,V$; therefore, we take 1200 samples of V_G in the range $[-3.6, 3.6]\,V$.

It is clear from the above discussion that the number of samples increases significantly with t_{si}. As mentioned in Sec. II A, t_{si} should be chosen large enough so that there is a sufficient neutral region to make the electric field $\Psi'(y)$ decay beyond the depletion width and become insignificant at $y = t_{si}$. It, thus, poses a great challenge to optimize the model for a low value of N_A, which increases the depletion width. We, therefore, limit the lowest value of N_A at $0.1\times 10^{24}\,\mathrm{m}^{-3}$ based on the available computational resources. At the same time since we are using $C_{ox} = \varepsilon_{ox}/t_{ox}$ in the boundary condition, our model does not face a similar challenge for high-ε gate dielectric.

Of the four inputs, the magnitude of three inputs is highly varying: t_{ox}, $y \sim 10^{-9}$, and $N_A \sim 10^{24}$. Therefore, we propose normalizing these (using their mean and standard deviation) before feeding them into the network. Moreover, since the normalized inputs are dimensionless, (17) does not suffer from a dimension mismatch. The above sampling strategy produces a massive data set of size $3000\times 29\times 41\times 1200 \sim 10^9$. To control this, we break the data into two parts. We consider all possible combinations y and V_G and denote this by $\mathscr{D}$. The size of $\mathscr{D}$ is $3000\times 1200 \sim 10^6$. On the other hand, we randomly sample points from all possible combinations of t_{ox} and N_A; we denote the reduced set by $\mathscr{D}'$. We consider the loss function

$$\mathscr{L}(\theta) = \sum |\, E(y, V_G, t_{ox}, N_A;\theta)|, \tag{18}$$

where

$$E(y,\xi;\theta) = \psi''(y,\xi;\theta) + \frac{qN_A}{\varepsilon_{si}}\left[e^{-\frac{\psi(y,\xi;\theta)}{\Phi_t}} - 1 - e^{-\frac{2\Phi_B}{\Phi_t}}\left(e^{\frac{\psi(y,\xi;\theta)}{\Phi_t}} - 1\right)\right], \tag{19}$$

and the sum in (18) is over $(y, V_G)\in\mathscr{D}$ and $(t_{ox}, N_A)\in\mathscr{D}'$. Note that unlike Refs. 24 and 25, we use the L_1 metric in (18). Regarding the choice of activation function[35] for the neural network, though ReLU is ubiquitously used,[9] we cannot use it for our problem because, since the second derivative of ReLU is zero, this will force $\mathcal{N}''(y,\xi;\theta)$ to vanish. Instead, we use a hyperbolic tangent (tanh) as the activation function. This ensures that the proposed trial $\mathcal{N}(y,\xi;\theta)$ is differentiable in y and θ. We need the derivative in y to define $\psi(y,\xi;\theta)$ in the first place. On the other hand, we need the derivative with respect to θ for gradient-based learning. We do not use activation in the output layer because the output of tanh is limited to $(-1, 1)$, whereas the predicted potential should be allowed to assume any real value and not just values in $(-1, 1)$. For minimizing $\mathscr{L}(\theta)$, we use the Adam optimizer[10] with a batch size of $20\,k$. The initial learning rate is 0.001 and is reduced by 96% after every 1000 epochs. We run the optimization steps for $150\,k$ epochs, which took 16 days on a NVIDIA A100 GPU. The evolution of the loss is shown in Fig. S2. We generally noticed that gradient-based optimization of (18) using gradient descent is slow in practice and can be speeded up if we optimize $\log\mathscr{L}(\theta)$ instead of $\mathscr{L}(\theta)$.

E. Validation and physical insights

We validated our model against the numerical solution of PBE and SPE. The results are shown in Figs. 3(a)–3(c). We observe that the maximum absolute error in the surface potential is around 1 mV, and the maximum error in predicting $\Psi(y)$ is 5 mV. We see from Fig. 3(c) that our model can accurately interpret the accumulation, depletion, and weak and strong inversion mechanisms. Figure 3(d) shows that our model can also apprehend the effect of device parameter variation. In fact, the model has captured the

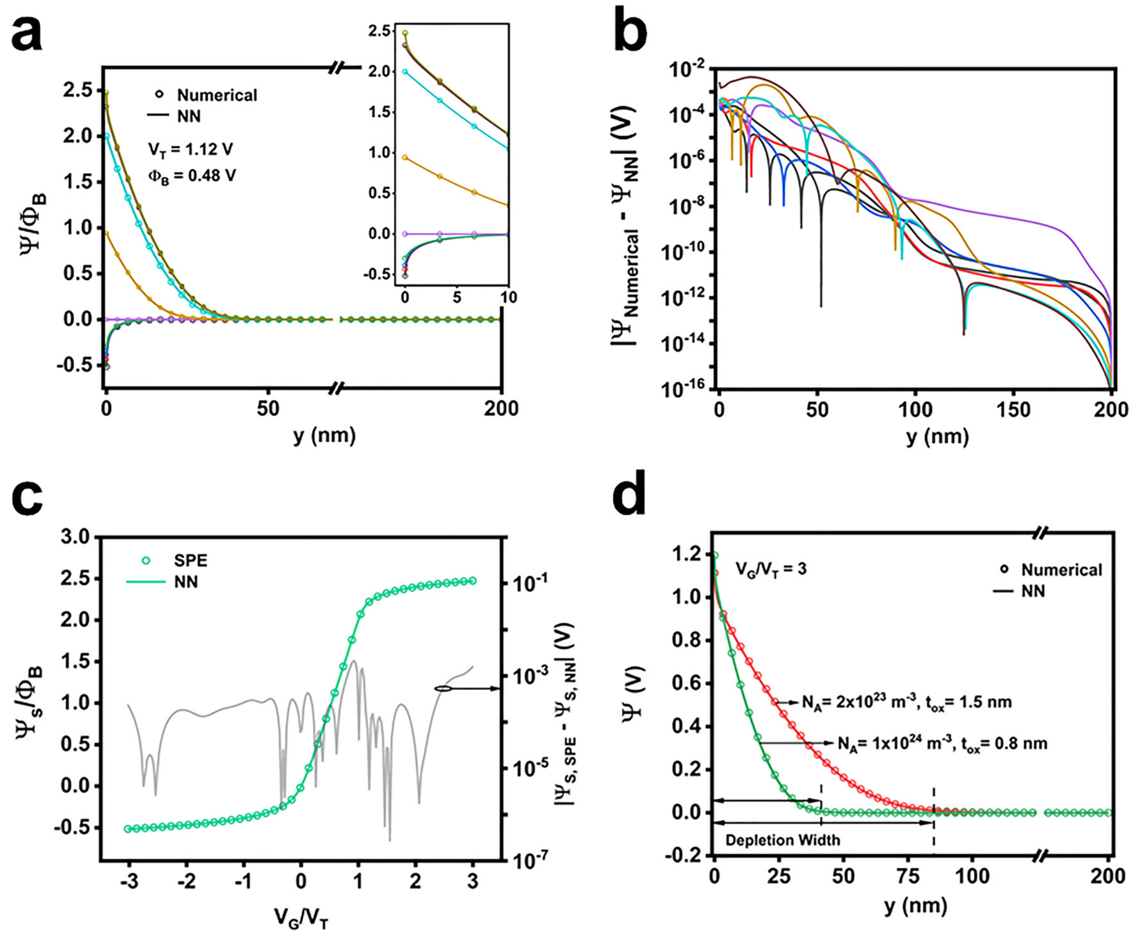

FIG. 3. Neural network prediction vs numerical solution. (a) Comparison for various normalized gate voltage (V_G/V_T); the corresponding absolute errors are shown in (b). (c) and (d) Similar comparison for the normalized surface potential. For (a)–(c), $N_A = 10^{24}\,\mathrm{m}^{-3}$ and $t_{ox} = 1\,\mathrm{nm}$. (d) Prediction from our model for two diverse settings of N_A and t_{ox}.

fundamental physics of change in depletion width with doping concentration. We present predictions for various combinations of N_A and t_{ox} in Fig. S3. We use our model to draw further insights into MOS capacitor device physics. Figure 4(a) shows the variation of inversion charge density with the applied gate voltage. The inversion charge per unit area is calculated as

$$Q = -\frac{qn_i^2}{N_A}\int_0^{t_{si}} e^{\frac{\Psi(y)}{\Phi_t}}\,dy. \tag{20}$$

We see that our model successfully predicts inversion charge density characteristics that vary exponentially with gate voltage in the sub-threshold regime ($V_G < V_T$) and linearly in the strong inversion regime ($V_G > V_T$). This is a signature of MOS capacitor characteristics. It is worth noting that the drain current in a MOS transistor also follows a similar trend. We further try to estimate physical properties, such as the threshold voltage from (7) and the sub-threshold slope factor η_0 from the $Q{-}V_G$ characteristics using the formula

$$\eta_0 = \frac{1}{2.3\Phi_t}\cdot\frac{dV_G}{d\log|Q|}.$$

Figures 4(b) and 4(c) reveal that our model accurately captures the variations of these two parameters as a function of N_A and t_{ox}. Reduction of t_{ox} results in enhancement of oxide capacitance C_{ox}, increasing the gate control over the inversion layer. As a result, both V_T and η_0 decrease with the decrease in t_{ox}. At the same time, a higher value of N_A increases the depletion charge and the depletion capacitance, which effectively increases the values of V_T and η_0. It is worth noting that though our model is optimized to fit the PBE, it can accurately predict a variety of other physical phenomena, such as threshold voltage, sub-threshold slope factor, variation of inversion charge with gate voltage, and so on.

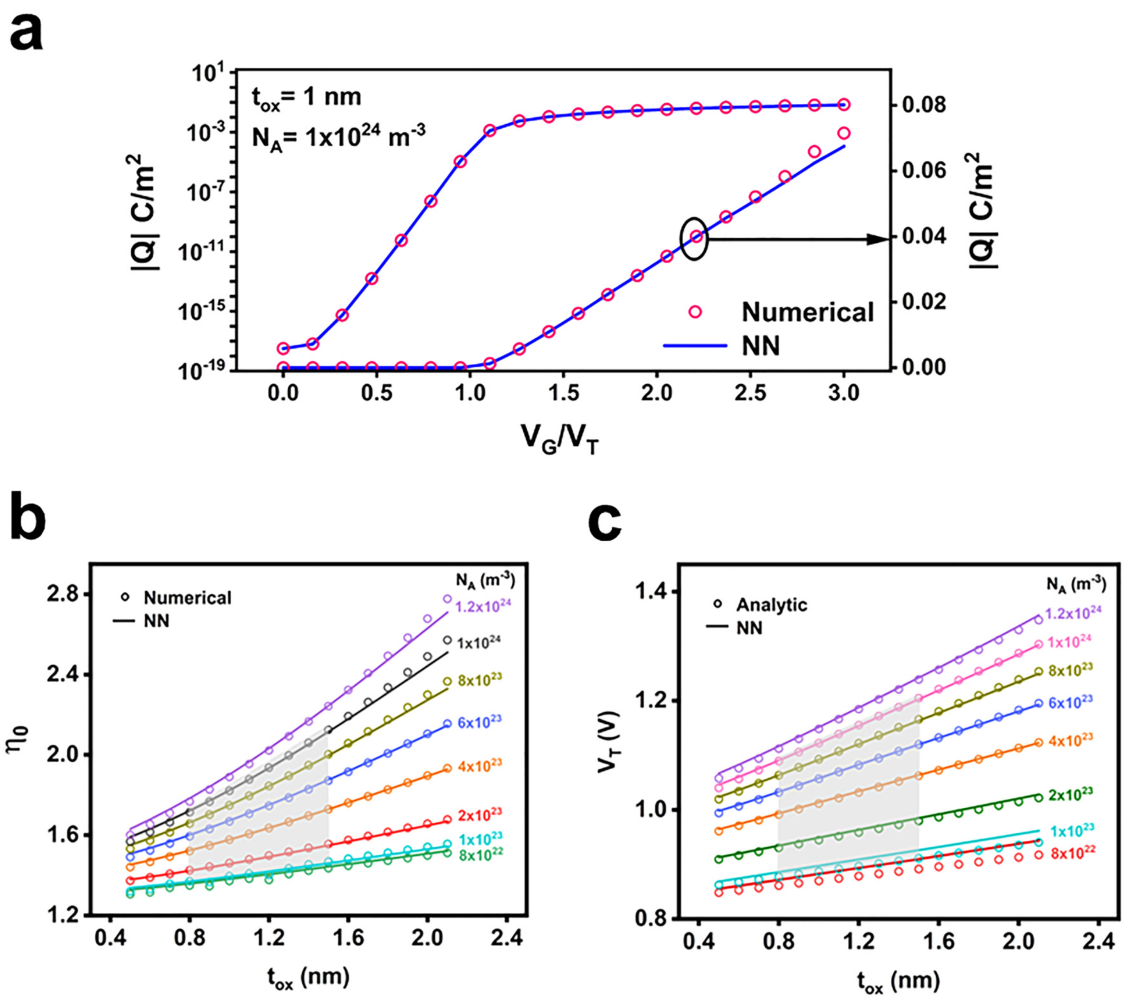

FIG. 4. Calculation of inversion charge density, sub-threshold slope factor, and threshold voltage. (a) Inversion charge per unit area as obtained from (20) as a function of the normalized V_G. Variation of (b) a sub-threshold slope factor and (c) a threshold voltage as a function of t_{ox} for different N_A. The shaded region represents the sampling domain.

We now examine if the proposed model captures the device physics outside the sampling domain. In Figs. 4(b) and 4(c), we observe that our model can capture the variation imposed by N_A and t_{ox} outside the sampling domain quite well. However, V_T and η_0 are estimated from the linear region of Ψ_s–V_G characteristics, which is easier to learn since the exponential terms in PBE are insignificant there. Therefore, we test our model by extrapolating it outside its sampling domain on the V_G scale, as shown in Fig. 5. We also observe the performance of our model for the prediction of low-frequency gate capacitance given by

$$C_{GG} = C_{ox}\left(1 - \frac{d\Psi_s}{dV_G}\right).$$

We see that the extrapolation is decent for $V_G < 0$ ($\approx 1V$ until $V_G/V_T = -4$); however, it is incremental for $V_G > 0$. The origin of such a discrepancy can be explained using Fig. 1(c) and (3). Since our sampling domain on the V_G scale is limited to $[-3V_T, 3V_T]$, the model is being introduced to a larger accumulation regime [where the $\exp(-\Psi/\Phi_t)$ factor dominates] and a smaller strong inversion regime [where the $\exp((\Psi - 2\Phi_B)/\Phi_t)$ term dominates]. Therefore, the model is better able to fit the accumulation region than the strong inversion region.

F. Discussion

Though a substantial portion of the sampling regime appears to be accurately captured by our NN model, the accuracy at the lowest doping level does not meet expectations, as seen in Figs. 4(c) and S4. To understand this issue better, we conducted another experiment with much lower doping concentration. We optimized the NN for fixed values of $t_{ox} = 1.5\,\text{nm}$ and $N_A = 0.05 \times 10^{24}\text{m}^{-3}$.

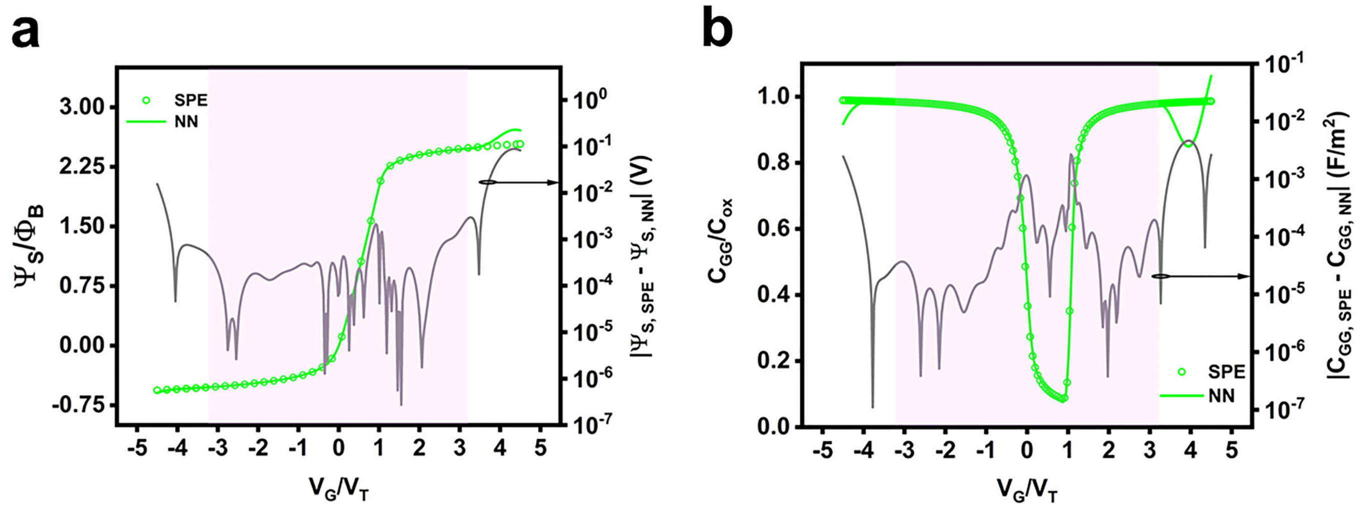

FIG. 5. Model extrapolation on the V_G scale. (a) Normalized surface potential and (b) normalized gate capacitance. The shaded region denotes the sampling domain. The device parameters are the same as in Fig. 1.

t_{si} is chosen to be 250 nm to accommodate the deeper depletion width, and the sampling is done accordingly. We find that the NN prediction matches near-exactly with the numerical results (see Fig. S5). We can conclude that our model inherently is not limited by low substrate doping. However, while getting optimized over a very large range of N_A (which also leads to a large variation of depletion width where the Poisson equation behaves as a Laplace equation), our model is not able to learn the solution for smaller N_A accurately. This issue requires further investigation.

III. CONCLUSION

In this study, we demonstrated how a feedforward neural network can solve the Poisson–Boltzmann equation governing an MOS capacitor. In particular, we used the method of trial solutions proposed by Lagaris *et al.*,[24] where the novelty was the specification of a trial solution that conforms to the mixed (Dirichlet + Robin) boundary conditions of the Poisson–Boltzmann equation. Comparing with the solution predicted by traditional numerical methods, we confirm that the proposed model can learn the relationship between the input parameters (y, V_G, t_{ox}, N_A) and the potential $\Psi(y)$ with exceptional accuracy. We find that the neural network can interpret several important aspects of MOS device physics, such as the doping-dependent depletion width, variation of threshold voltage with oxide thickness and doping, and the low-frequency capacitance–voltage characteristics. The extrapolation capability of the model shows that it does not just memorize the results from the optimization regime but is also able to capture the device physics.

SUPPLEMENTARY MATERIAL

See the supplementary material for details of PINN model performance, the evolution of the proposed model during optimization, and various other results, including experiments with the number of neurons per layer.

ACKNOWLEDGMENTS

The authors acknowledge the generous corporate social responsibility grant from GroupM Media India Private Limited for the M.Tech. AI program. The Nvidia A100 GPUs used in this work were procured through this grant.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

S.M. conceived the problem statement. T.G.I. developed the machine learning model under the supervision of K.N.C. S.G. developed the solution of the surface potential equation and assisted in composing the figures. All the authors contributed to the manuscript preparation.

Tejas Govind Indani: Data curation (lead); Formal analysis (lead); Investigation (lead); Methodology (lead); Validation (lead); Visualization (lead); Writing – original draft (equal); Writing – review & editing (equal). **Kunal Narayan Chaudhury:** Conceptualization (equal); Supervision (equal); Validation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Sirsha Guha:** Methodology (supporting); Visualization (supporting); Writing – original draft (supporting); Writing – review & editing (supporting). **Santanu Mahapatra:** Conceptualization (equal); Supervision (equal); Validation (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available within the article and its supplementary material. Code and ML models used to generate the data can be obtained from https://github.com/tejas-73/Physically-constrained-learning-of-MOS-capacitor-electrostatics.

APPENDIX: BOUNDARY CONDITIONS VERIFICATION

In this part, we give detailed calculations to show that the proposed trial solution (17) satisfies the boundary conditions in (11). It is obvious that $\psi(t_{si};\theta) = 0$ for all θ. On the other hand, we can write (17) as

$$\psi(y;\theta) = \alpha\, e^{-\mathcal{N}(y;\theta)}\left(1 - \frac{y}{t_{si}}\right), \tag{A1}$$

where

$$\alpha = \left(\frac{\gamma\, e^{\mathcal{N}(0;\theta)}}{1+\gamma+t_{si}\mathcal{N}'(0;\theta)}\right)V_G.$$

Differentiating (A1) with respect to y, we have

$$\psi'(y;\theta) = -\alpha\, e^{-\mathcal{N}(y;\theta)}\left(\frac{1}{t_{si}} + \left(1-\frac{y}{t_{si}}\right)\mathcal{N}'(y;\theta)\right).$$

In particular,

$$\psi'(0;\theta) = -\frac{\alpha}{t_{si}}\, e^{-\mathcal{N}(0;\theta)}\left(1+t_{si}\mathcal{N}'(0;\theta)\right). \tag{A2}$$

On the other hand, we have

$$\begin{aligned} V_G - \psi(0;\theta) &= V_G - \alpha\, e^{-\mathcal{N}(0;\theta)} \\ &= V_G - \frac{\gamma}{1+\gamma+t_{si}\mathcal{N}'(0;\theta)}V_G \\ &= \left(\frac{1+t_{si}\mathcal{N}'(0;\theta)}{1+\gamma+t_{si}\mathcal{N}'(0;\theta)}\right)V_G \\ &= \frac{\alpha}{\gamma}\, e^{-\mathcal{N}(0;\theta)}\left(1+t_{si}\mathcal{N}'(0;\theta)\right). \end{aligned} \tag{A3}$$

Combining (A2) and (A3), we get the second condition in (11).

REFERENCES

[1]G. Evans, J. Blackledge, and P. Yardley, *Numerical Methods for Partial Differential Equations* (Springer Science & Business Media, 2012).
[2]M. Jain and T. Aziz, "Spline function approximation for differential equations," Comput. Methods Appl. Mech. Eng. **26**, 129–143 (1981).
[3]Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**, 436–444 (2015).
[4]L. Deng and D. Yu, "Deep learning: Methods and applications," Found. Trends Signal Process. **7**, 197–387 (2014).
[5]C. M. Bishop and N. M. Nasrabadi, *Pattern Recognition and Machine Learning* (Springer, 2006).
[6]J. Han, A. Jentzen, and E. Weinan, "Solving high-dimensional partial differential equations using deep learning," Proc. Natl. Acad. Sci. U.S.A. **115**, 8505–8510 (2018).
[7]M. Raissi, "Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations," arXiv:1804.07010 (2018).
[8]P. Grohs, F. Hornung, A. Jentzen, and P. von Wurstemberger, *A Proof That Artificial Neural Networks Overcome the Curse of Dimensionality in the Numerical Approximation of Black–Scholes Partial Differential Equations*, Memoirs of the American Mathematical Society (American Mathematical Society, 2023), Vol. 284. https://doi.org/10.1090/memo/1410.
[9]I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016).
[10]D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations* (ICLR, 2014).
[11]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (Curran Associates, Inc., 2019).
[12]M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "TensorFlow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation* (USENIX Association, 2016), pp. 265–283.
[13]A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," J. Mach. Learn. Res. **18**, 1–43 (2018).
[14]A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Advances in Neural Information Processing Systems* (NIPS, 2017).
[15]A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," Computer **29**, 31–44 (1996).
[16]K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural Netw. **2**, 359–366 (1989).
[17]K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," Neural Netw. **3**, 551–560 (1990).
[18]L. Bar and N. Sochen, "Unsupervised deep learning algorithm for PDE-based forward and inverse problems," arXiv:1904.05417 (2019).
[19]Y. Guo, X. Cao, B. Liu, and M. Gao, "Solving partial differential equations using deep learning and physical constraints," Appl. Sci. **10**, 5917 (2020).
[20]I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, "Neural-network methods for boundary value problems with irregular boundaries," IEEE Trans. Neural Netw. **11**, 1041–1049 (2000).
[21]M. Raissi and G. E. Karniadakis, "Hidden physics models: Machine learning of nonlinear partial differential equations," J. Comput. Phys. **357**, 125–141 (2018).
[22]Z. Long, Y. Lu, X. Ma, and B. Dong, "PDE-Net: Learning PDEs from data," in *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research (PMLR, 2018), pp. 3208–3216.
[23]K. Wu and D. Xiu, "Data-driven deep learning of partial differential equations in modal space," J. Comput. Phys. **408**, 109307 (2020).
[24]I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," IEEE Trans. Neural Netw. **9**, 987–1000 (1998).
[25]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[26]I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," Commun. ACM **63**, 139–144 (2020).
[27]B. Bullwinkel, D. Randle, P. Protopapas, and D. Sondak, "DEQGAN: Learning the loss function for PINNs with generative adversarial networks," arXiv:2209.07081 (2022).
[28]M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," arXiv:1701.04862 (2017).
[29]Y. Tsividis, *Operation and Modeling of the MOS Transistor* (McGraw-Hill, Inc., 1987).

[30]T. Ezaki, H. J. Mattausch, and M. Miura-mattausch, *Physics and Modeling of MOSFETS: Surface-Potential Model HiSIM* (World Scientific, 2008).
[31]G. Gildenblat, W. Wu, X. Li, R. van Langevelde, A. J. Scholten, G. D. Smit, and D. B. Klaassen, "Surface-potential-based compact model of bulk MOSFET," in *Compact Modeling: Principles, Techniques and Applications* (Springer, 2010), pp. 3–40.
[32]P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, and P. van Mulbregt and, SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," Nat. Methods **17**, 261–272 (2020).
[33]Z. K. Lawal, H. Yassin, D. T. C. Lai, and A. Che Idris, "Physics-informed neural network (PINN) evolution and beyond: A systematic literature review and bibliometric analysis," Big Data Cogn. Comput. **6**, 140 (2022).
[34]T. Skotnicki and F. Boeuf, "Optimal scaling methodologies and transistor performance," in *High Dielectric Constant Materials: VLSI MOSFET Applications* (Springer, 2005), pp. 143–194.
[35]S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," Int. J. Eng. Appl. Sci. Technol. **6**, 310–316 (2020).