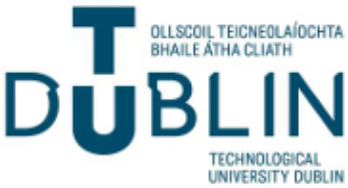


Numerical Solutions of Dopant Diffusion in Semiconductor Structures

by

Onurhan Cansever

D24127964

A thesis submitted in partial fulfilment of the requirements for the MSc. Degree in Electronics and Communications Engineering (TU203) of the School of Electrical and Electronics Engineering of the Technological University Dublin, City Campus

Supervised by

Dr Kevin Berwick

DECLARATION

I, the undersigned, certify that the thesis which I now submit for examination is for the award of Master of Science and is based on my work carried out during my study and has not been taken from the work of others, save to the extent that such work has been cited and acknowledged within the text of my work. This thesis was prepared according to the regulations for postgraduates of the Technological University Dublin and has not been submitted in whole or in part for an award in any other Institute or University. The work reported in this thesis conforms to the principles and requirements of the University's guidelines for ethics in research. The University has permission to keep, lend or copy this thesis in whole or in part, on condition that any use of the material of the thesis is duly acknowledged.

Name: Onurhan Cansever,

Student Number: D24127964

Signature:

Date: 01.09.2025

Acknowledgements

I have made sincere efforts to carry out this project; however, it would not have been possible without the guidance, support and encouragement of several people. I want to use this opportunity to express my heartfelt gratitude to all of them.

I would like to express to my supervisor, Dr. Kevin Berwick, School of Electrical and Electronic Engineering, University of Technology Dublin deep appreciation for his continued support, valuable insights and guidance at various stages of this thesis.

I am also really grateful to Andrew Dillon for providing access to High Performance Computing (HPC) resources, which greatly facilitates the completion of this work.

Finally, I would like to express my sincere thanks to Stewart Killeen (Librarian) for his help in developing a search strategy and teaching me how to navigate the literature databases that significantly enrich the research process.

Table of Contents

Table of Figures

1.Introduction

Diffusion of dopants is a critical stage in semiconductor manufacturing. Dopant atoms are added to change the electrical properties of the silicon wafer. In this way, it is possible to create special zones such as emitter, base and collector in transistors.

Nowadays, things are even more sensitive. The right diffusion profiles are required in everything from microprocessors to memory chips. Because in these nanometer-sized dimensions, the amount of dopant needs to be adjusted very carefully. By understanding and simulating the diffusion process well, engineers can both improve the design of the devices and improve efficiency in production.

1.1 Aim of the Project

The main objective of this project is to develop and study the numerical solutions of dopant diffusion in semiconductor structures. In the study, the diffusion equation is used for different situations: constant source diffusion (predeposition), drive-in diffusion, concentration-dependent diffusion and electric field effects.

For this purpose, finite difference method is used to solve Fick's second law. MATLAB is preferred because it offers strong possibilities for numerical modeling and visualization. In addition, analytical solutions were obtained, and verification was provided by comparing them with numerical results.

Simulations are not limited to only 1D but are also extended to 2D and 3D geometries. Thus, structures closer to real devices, such as the emitter region of a bipolar junction transistor (BJT), could be modeled. High-Performance Computing (HPC) resources have also been used to run more complex simulations and evaluate performance.

The project also explores the use of machine learning techniques to predict junction depth based on process parameters. An interactive MATLAB app was created to allow users to input desired outcomes and receive estimated process values.

1.2 The Diffusion Equation

The process of diffusion occurs when particles are moved from regions with high concentration to regions with less concentration. This process describes how concentration changes over time and is often explained in the literature by Fick's second law. [13]

$$\frac{\partial C}{\partial t} = D\frac{\partial^2 C}{\partial x^2} \qquad (1)$$

Where C is concentration, t is time, D is diffusion coefficient, x is position.

This law defines the change in intensity over time at a given point. The diffusion coefficient D here depends on the type and temperature of the dopant used. In most cases, D grows in the Arrhenius type, that is, exponentially, as the temperature increases. [3]

$$D = D_0 \exp\left(-\frac{E_a}{kT}\right) \qquad (2)$$

1.3 Analytical Solution of the Diffusion Equation

The simplest solution to the diffusion equation happens when the system reaches a steady state, meaning the concentration doesn't change over time. In this case, the equation simplifies to:

$$D\frac{\partial^2 C}{\partial x^2} = 0 \qquad (3)$$

Integrating twice with respect to x, we obtain:

$$C(x) = Ax + B \qquad (4)$$

Where A and B are constants determined by boundary conditions. This shows that the concentration varies linearly with distance under steady-state conditions.

1.3.1 Constant Source Diffusion (Predeposition)

Predeposition is the process where a constant concentration of dopant atoms is maintained at the surface of a semiconductor material, allowing the dopants to diffuse into the material.

$$C(0,t) = C_s, \;\; C(\infty,t) = 0 \qquad (5)$$

Where C_s is the constant source concentration. The analytical solution to Fick's second law under these conditions is:

$$C(x,t) = C_s \,\mathrm{erfc}\left(\frac{x}{2\sqrt{Dt}}\right) \qquad (6)$$

Where erfc is the complementary error function. The diffusion depth increases with time and the dopant dose increases as more impurities diffuse into the wafer. This process is commonly used during the initial stage of semiconductor fabrication to introduce dopants uniformly near the surface. [24]

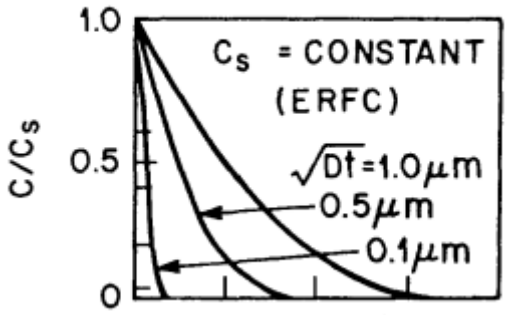


Figure 1: Constant Source Diffusion (erfc versus distance for successive diffusion times.)

The profiles in figure 1 correspond to constant source diffusion (predeposition) and follow the complementary error function solution.

1.3.2 Drive-In Diffusion

Drive-in diffusion initially begins with the presence of a thin layer of impurities on the surface of the substrate, but this concentration gradually decreases over time. Unlike predeposition, the surface concentration is no longer constant, and the dopant atoms redistribute according to a Gaussian profile rather than the erfc profile. [25]

Gaussian profile equations used for analytical solutions are:

$$C(x,t) = \left(\frac{Q}{\sqrt{\pi Dt}}\right)\exp\left(-\frac{x^2}{4Dt}\right) \qquad (7)$$

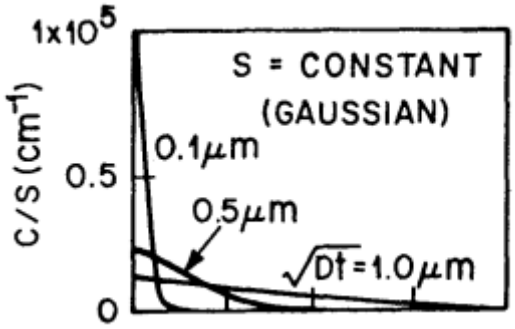


Figure 2: Gaussian profiles for different diffusion steps.

The profile in figure 2 explains the characteristic of the drive-in diffusion step where the surface concentration decreases over time. The expression "S (or Q) = constant" indicates that the total amount of dopant (Q) remains constant during the drive-in process. This condition is consistent with the Gaussian solution.

The main purpose of drive-in diffusion is to achieve a desired junction depth while reducing the surface dopant concentration to prevent high electric field effects.

1.4 Technological Importance of Diffusion

In semiconductor manufacturing, diffusion is one of the most important processes. Diffusion is not only a planned step. Thus, it might occur anytime during the heating process. It is carried out in specialized diffusion furnaces, where silicon wafers are processed under high temperature and controlled atmosphere. Understanding and prediction of diffusion profiles is essential for reliable CMOS technology.

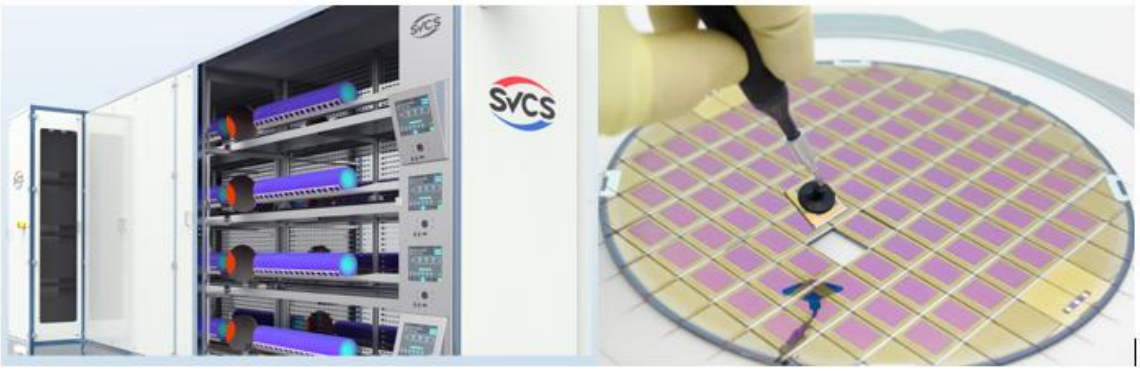


Figure 3: A typical silicon wafer containing multiple integrated circuits (right) and a Horizontal Diffusion Furnace (left).

Precise dopant distribution is important because even small deviations can change the junction depth and overall efficiency emphasizes the technological importance of diffusion in device manufacturing. [28]

1.5 Numerical Solution of the Diffusion Equation

The diffusion equation can be discretized using the finite difference method. The standard numerical formulation is:

$$C_i^{t+1} = C_i^t + \frac{D\Delta t}{\Delta x^2}(C_{i-1}^t - 2C_i^t + C_{i+1}^t) \quad (8)$$

For numerical stability, we impose:

$$\frac{D\Delta t}{\Delta x^2} \leq 0.5 \quad (9)$$

If this condition is not met, the numerical solution becomes unstable, leading to oscillations in concentration values.

Let:

$$\frac{D\Delta t}{\Delta x^2} = \frac{1}{2}$$

we obtain a simplified form:

$$C_i^{t+1} = \frac{1}{2}(C_{i-1}^t + C_{i+1}^t) \quad (10)$$

This equation states that the new concentration at a point is simply the average of its neighboring values. [1]

1.6 Second Order Effects

Modern VLSI structures employ doped regions in which concentration dependent diffusion, electric field effects, dopant segregation and complicated point defect driven diffusion processes take place. All these effects generally require numerical methods to calculate the resulting dopant profiles.

1.6.1 Concentration Dependent Diffusion

Beyond the electric field's influence, a different diffusion influence happens when the doping level exceeds the intrinsic electron level at the diffusion temperature. Fick's first law assumes in general that the diffusion flux is directly proportional in totality and with accuracy to the concentration gradient. However, actual observations of dopant concentration profiles typically display a "box-like" shape, indicating that diffusion occurs more rapidly in regions of higher concentration. [25]

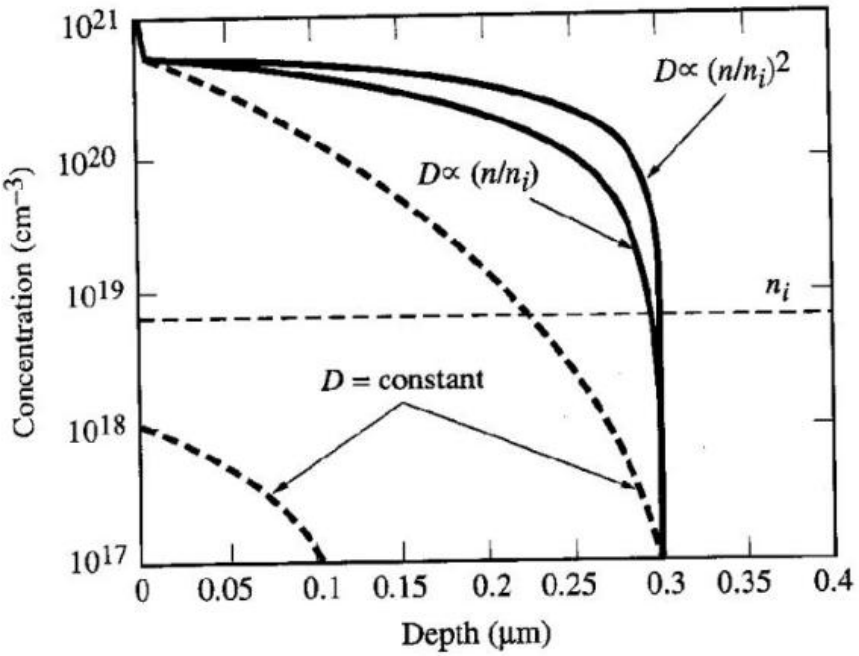


Figure 4: High Concentration Arsenic Diffusion Profile becomes "box-like"

Figure 4 shows how arsenic diffusion changes with concentration. When diffusivity is constant, the profile is smooth and gradual. However, at high concentrations, diffusivity increases with carrier concentration, creating a flat region near the surface and a sharp drop at the junction. This results in a "box-like" profile,

Since D is not constant in such cases, the diffusion coefficient becomes a function of concentration and Fick's equation becomes nonlinear. Therefore, it cannot be solved analytically and must instead be solved numerically:

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left(D_A^{\text{eff}} \frac{\partial C}{\partial x}\right) \quad (11)$$

The effective diffusivity coefficient is calculated using equation 12:

$$D_A^{\text{eff}} = D^0 + D^- \left(\frac{n}{n_i}\right) + D^= \left(\frac{n}{n_i}\right)^2 \quad (12)$$

These individual diffusivity terms can be found using the equation below:

$$D = D.0 \exp\left(-D.\frac{E}{kT}\right) \quad \textbf{(13)}$$

The following table presents concentration-dependent diffusivities for common dopants in single-crystal silicon. The values for D^0 (pre-exponential factor) are given in cm^2/sec while D^E (activation energy) values are provided in electron volts (eV).

	Si	B	In	As	Sb	P
D_0^0	560	0.05	0.6	0.011	0.214	3.85
D^E	4,76	3.5	3.5	3.44	3.65	3.66
D^0	0.95	0.6				
D^E	3.5	3.5				
D^0			31.0	15.0		4.44
D^E			4.15	4.08		4.0
D^0					44.2	
D^E					4.37	

Table 1: Concentration-dependent diffusivities of common dopants in single-crystal silicon.

By rewriting the above equations, the diffusion coefficient measured under extrinsic conditions can be described as:

$$D_A^{\text{eff}} = D_A^* \left(\frac{1+\beta\frac{n}{n_i}+\gamma\left(\frac{n}{n_i}\right)^2}{1+\beta+\gamma} \right) \qquad (14)$$

In semiconductor technology, concentration-dependent diffusion plays a key role in accurately shaping dopant profiles during fabrication. As dopant concentration increases, the diffusion rate also changes, especially at high levels, leading to non-uniform diffusion behavior. This effect is essential for forming shallow junctions with precise control over depth and concentration.

1.6.2 Electric Field Effects

When dopant concentrations in silicon exceed the intrinsic carrier concentration (n_i) at diffusion temperatures, internal electric fields significantly alter diffusion behaviour. This extrinsic effect occurs exclusively in heavily doped regions due to charge separation. [25]

In heavily arsenic-doped (n-type) regions, for example:

- Electrons diffuse rapidly ahead of arsenic ions due to their high mobility.
- This creates a localized positive charge (from ionized As^+ donors left behind).
- The charge separation establishes an internal electric field (ϵ).

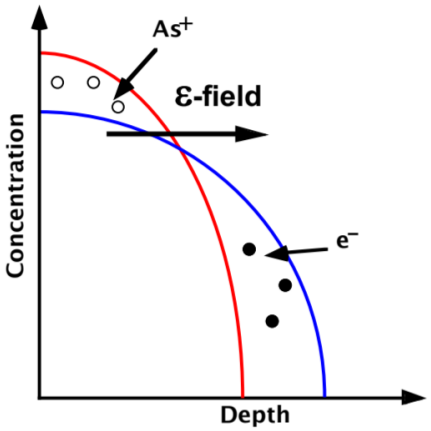


Figure 5: Schematic of Electric Field Effect

This internal electric field has two main consequences:

- It slows down the electron flow by creating a counteracting drift force.
- It pulls dopant ions (like As^+) deeper into silicon, enhancing diffusion.

The total dopant flux **F** under electric field becomes:

$$F = -D\frac{\partial C}{\partial x} - DC\frac{\partial}{\partial x}\ln\frac{n}{n_i} \quad (15)$$

Where D is the diffusivity, C is the dopant concentration, n is the electron concentration, n_i is the intrinsic carrier concentration.

This equation shows that in addition to normal diffusion, there is an extra term due to the electric field.

To simplify modeling, this effect is sometimes written as an enhanced diffusivity:

$$F = -hD\frac{\partial C}{\partial x} \tag{16}$$

where **h** is the enhancement factor due to the electric field:

$$h \equiv 1 + \frac{C}{\sqrt{C^2 + 4n_i^2}} \tag{17}$$

The maximum value of h is 2. So, the electric field can increase dopant diffusion by up to 2x in high-concentration regions.

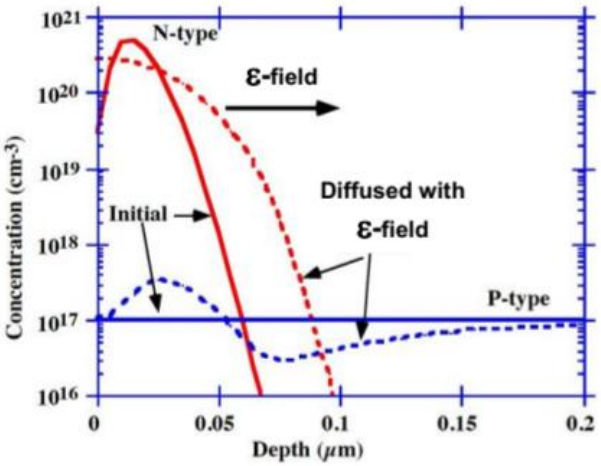


Figure 6: Electric Field Effect on Boron Diffusion.

Figure 6 shows electric field effect on boron diffusion near arsenic-doped region (1000°C). Field generated by high concentration As pulls B atoms into N^+ zone, depleting boron beyond the junction.

2. Hardware

2.1 Laptop

The laptop used in this research is a Lenovo ThinkPad with an Intel Core i5-8250U (8th Gen) processor. It has 4 physical cores and 8 threads, with a base clock speed of 1.70 GHz.

The graphic card used Intel UHD Graphics 620. The system has 16GB DDR-4 2400 RAM. The laptop has 14.1" HD (1366x768) anti-glare display with 220 nits brightness.

2.2 HPC Server

The high-performance computing server used in this project was ideally created to run the large scale simulations and complex numerical solutions. Unlike standard hardware, HPC server is very effective in running large datasets by using its features such as parallel processing in tasks using its high computational power. The HPC server provided by Technological University Dublin is a DELL PowerEdge R750xs Rack Server. It is equipped with two Intel Xeon Silver 4310 processors (each with 12 cores and 24 threads, running at 2.1 GHz with 18 MB cache, Turbo Boost, and Hyper-Threading support), resulting in a total of 24 cores and 48 threads. The system has 128 GB of DDR4-2666 RAM and 12 TB of storage, so it offers large memory and storage capacity. [7]

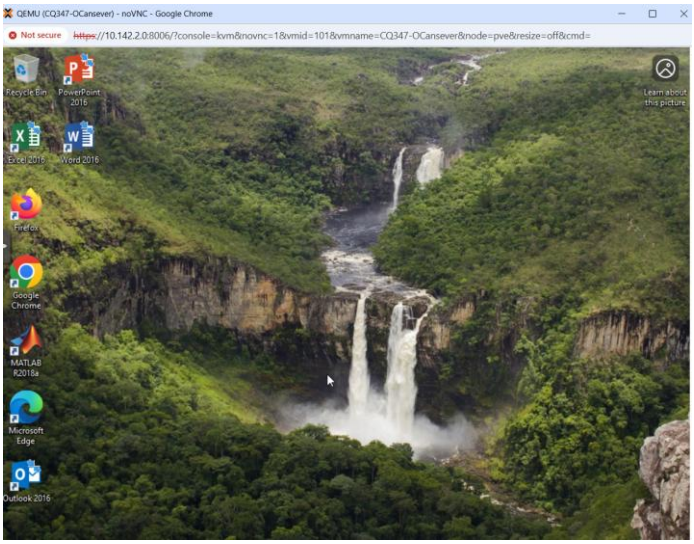


Figure 7: HPC Server Desktop

2.3 MATLAB®

MATLAB is a programming language and numerical computing environment developed by MathWorks. MATLAB allows matrix manipulation, drawing functions and data, implementing algorithms, creating user interfaces, and interfacing with programs written in other languages.

The platform offers users numerous benefits, making it such an effective tool. It:

• Carries out matrix-based calculations quickly.

• Has a lot of self-paced courses which makes it easy to learn

• Has tons of built-in functions

• Is very efficient to plot simulations.

• Allows user to create user-friendly applications. [4]

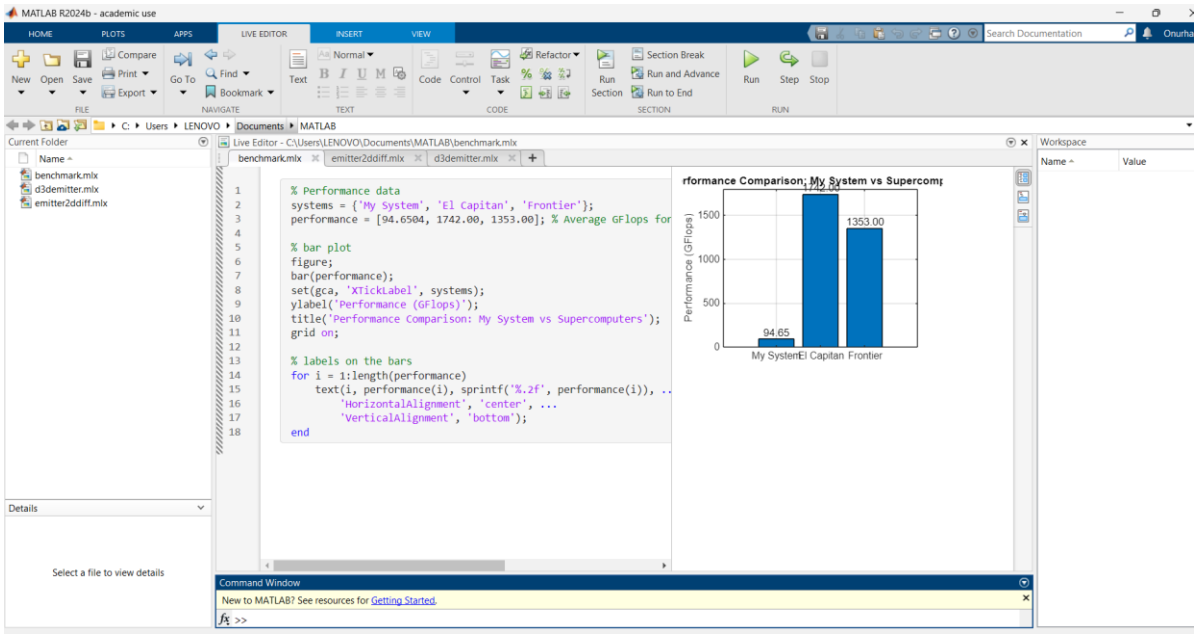


Figure 8: MATLAB User Interface

3. Results: 1D

This section discusses the numerical results of one-dimensional dopant diffusion. The main objective here is to verify the finite difference method by comparing it with known analytical models, and also to examine how the concentration profiles change according to the depth.

3.1 Constant Source Diffusion (Predeposition)

As illustrated in Figure 8, the simulation begins with a predefined concentration profile. This profile represents the state before any diffusion has taken place. Along the depth axis, 100 points are used so that the change in dopant concentration with depth can be observed.

At the starting point, only the first two grid points near the surface are assigned a high concentration, while all remaining sections are kept at zero. In this setup, those two surface points are given a dopant level of about 2×10^{19} cm^{-3} .This condition is called as predeposition stage.

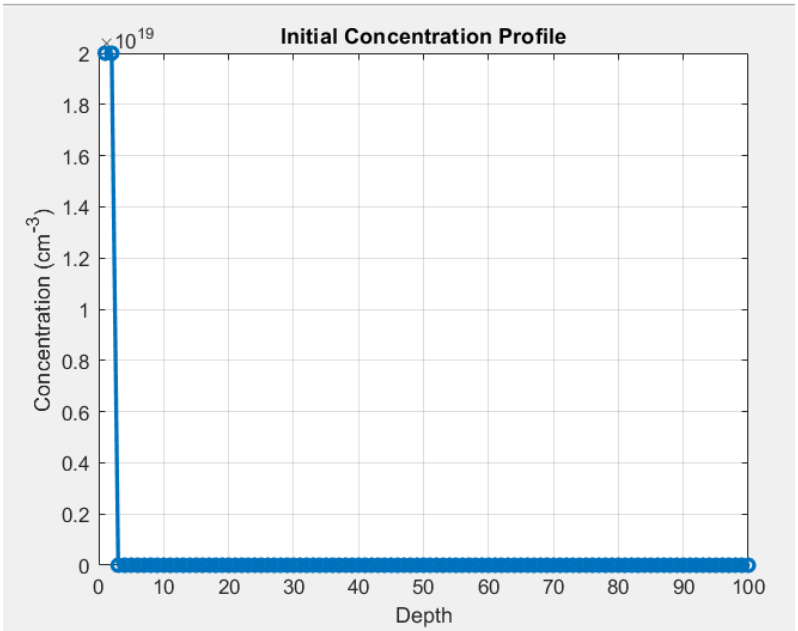


Figure 9: Initial Concentration Profile.

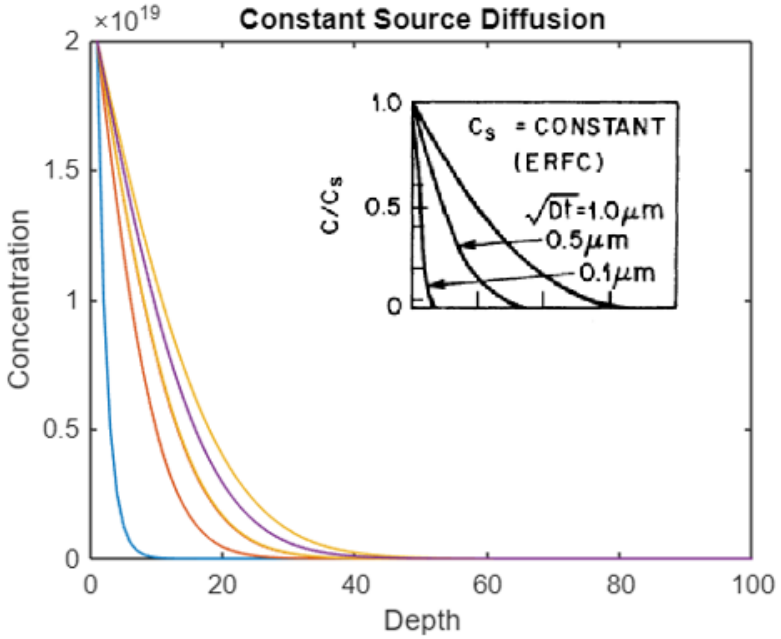


Figure 10: Numerical simulation results of constant source diffusion in 1D.

Figure 10 shows the numerical simulation of constant source diffusion 1D, modeling the predeposition step. The simulation is based on a simplified form of Fick's second law using the finite difference method. The update equation used is:

$$C_i^{t+1} = \frac{1}{2}(C_{i-1}^t + C_{i+1}^t) \quad (10)$$

The boundary condition is defined as:

$$C(1{:}2) = 2 \times 10^{19}\ cm^{-3}$$

the first two grid points near the surface are held at a high constant concentration (constant source). The simulation runs for 100-time steps, and a new profile is plotted every 25 iterations. Each line on the graph represents the dopant concentration profile after every 25-time steps, showing how the dopants diffuse deeper into the material also matches with theoretical profile.

3.2 Drive-In Diffusion

Drive-in diffusion is the process that happens after the initial deposition step (predeposition). At this stage, the surface source of dopants is removed, and the existing dopant profile spreads deeper into the material.

The same equation (10) is used, but the constant concentration condition at the surface is removed.

$$C_i^+ = \frac{1}{2}(C_{i-1} + C_{i+1}) \qquad (10)$$

```
// Update rule

C_new(i) = 0.5 * (C_old(i-1) +
C_old(i+1))

// Difference lies in boundary
condition

if Constant_Source:

    C(1:2) = 2e19   % Fixed boundary

else if Drive_In:

 % C is initialized with C(1:2) = 2e19

  %  C evolves freely, no fixed
boundary
```

Pseudo-code 1. Diffusion Update for Constant Source vs Drive-in

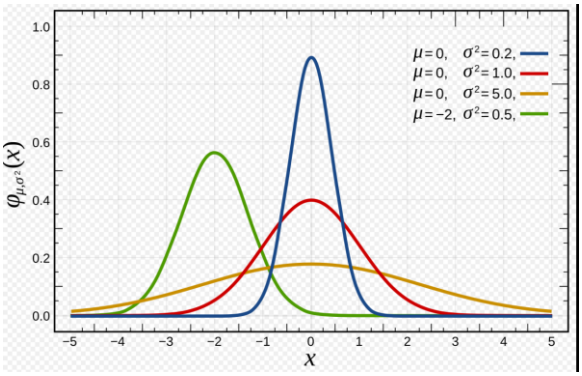


Figure 11: The Gaussian function is an example of a bell-shaped function

The figure below shows how dopant concentration evolves during drive-in diffusion. As outlined in Pseudo-code 1, the key difference from constant source diffusion is the absence of a fixed boundary. This allows leading to a gradual decrease in surface concentration. Over time, the distribution takes on a Gaussian shape, as illustrated in Figure 11 the profile spreading deeper into the material.

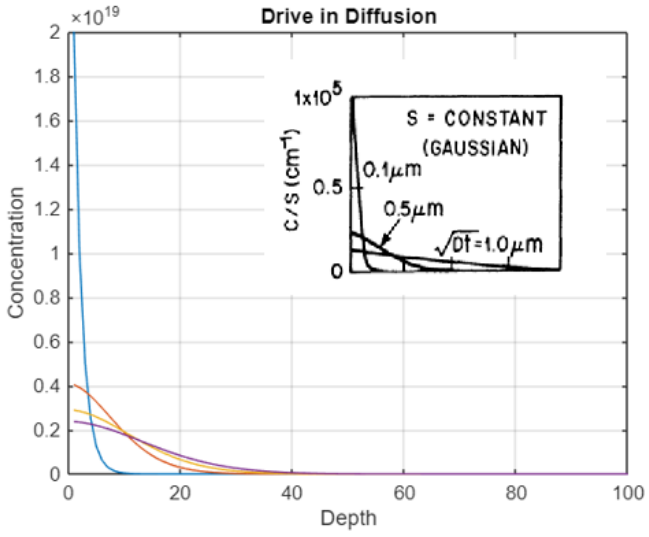


Figure 11: Drive-In Diffusion 1D Result.

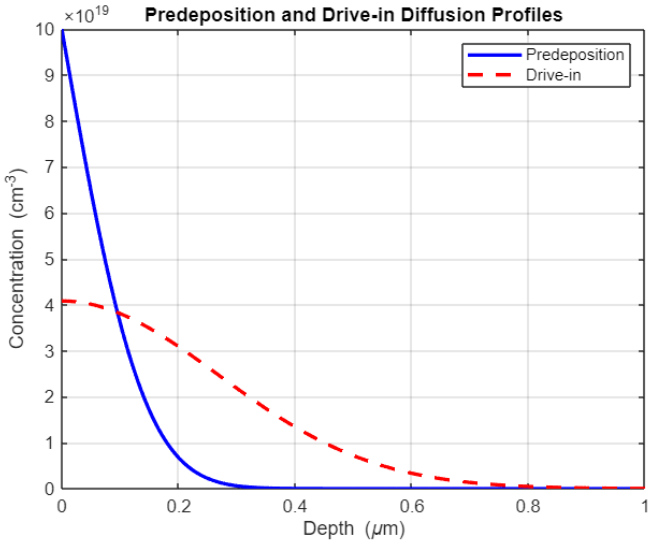


Figure 12: Predeposition and Drive-In Diffusion Profiles.

$$Q = C_s \cdot \sqrt{\pi D t_1} \tag{18}$$

In this simulation, a two-stage diffusion process was applied. The process started with the predeposition step first, then continued with the drive-in stage. During predeposition, the surface concentration is fixed at approximately Cs = 1×10^{20} cm^{-3}, allowing the dopant atoms allowed to diffuse for about 10 minutes. Following this step, a drive-in operation was performed that took 60 minutes for the dopant atoms to penetrate deeper into the material. The total dopant dose after predeposition was estimated using the equation above.

The blue curve corresponds to the predeposition profile while the red dashed curve represents the drive-in step, where dopants diffuse deeper and the profile becomes Gaussian-shaped.

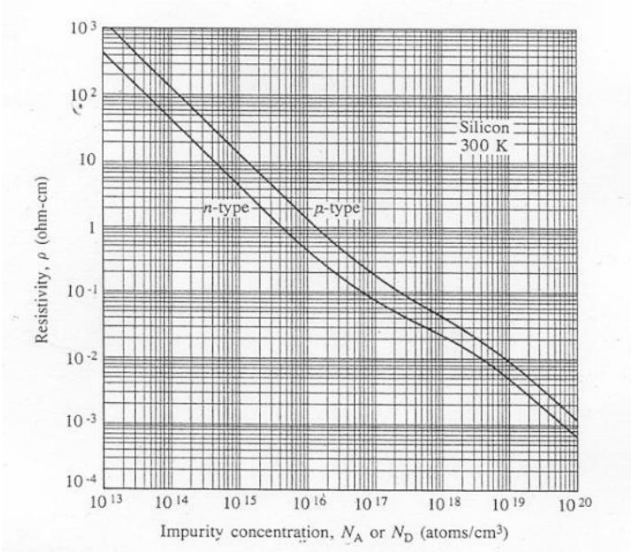
For further investigation of dopant diffusion, a question is solved below with realistic parameters such as surface concentration and temperature.

A uniformly doped n-type silicon wafer of 1 ohm-cm resistivity is subjected to a boron diffusion at a temperature of 1150°C. A constant surface concentration of 2×10^{19} cm^{-3} is maintained throughout the diffusion.

Calculate the background concentration in the wafer.

How long should the diffusion be carried out to obtain a junction depth of 4 microns?

Analytical Solution



From the impurity concentration vs. resistivity graph, for 1 ohm-cm n-type silicon:

$$\mathbf{C_B = 4 \times 10^{15}\ cm^{-3}}$$

We use the analytical solution for constant source diffusion:

$$C = C_s \operatorname{erfc}\left(\frac{x_j}{2\sqrt{Dt}}\right)$$

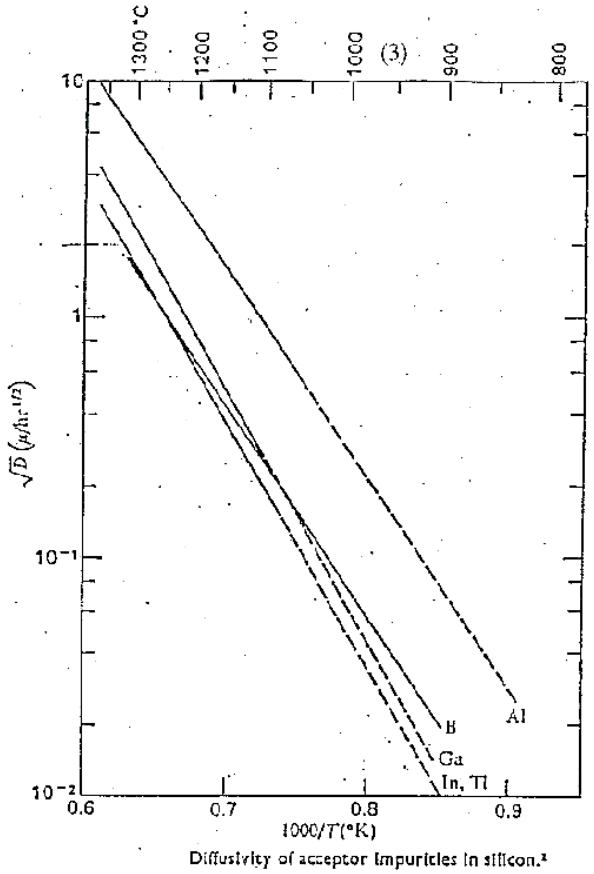
C_B = Background concentration $= 4 \times 10^{15}\ \text{cm}^{-3}$

C_s = Surface concentration $= 2 \times 10^{19}\ \text{cm}^{-3}$

x_j = Junction Depth $= 4\ \mu\text{m}$

From the temperature vs diffusivity graph for boron at 1150°C

$\sqrt{D} = 0.5\ \mu\text{m} \cdot \text{hr}^{-1/2}$



Diffusivity of acceptor impurities in silicon.[2]

$$\frac{C_B}{C_s} = \text{erfc}\left(\frac{x_j}{2\sqrt{Dt}}\right) = \text{erfc}\left(\frac{4\times10^{-4}}{2\sqrt{Dt}}\right)$$

$\frac{4\times10^{15}}{2\times10^{19}} = 2\times10^{-4}$ $\text{erfc}^{-1}(2\times10^{-4}) \approx 2.55$

$2.55 = \frac{4\times10^{-4}}{2\sqrt{Dt}} \Rightarrow \sqrt{Dt} = \frac{4\times10^{-4}}{2.55\times2}$ $Dt = \left(\frac{4\times10^{-4}}{5.1}\right)^2$

$\sqrt{D} = 5\times10^{-5}\ \text{cm}\cdot\text{hr}^{-\frac{1}{2}}$

$t = \frac{(4\times10^{-4})^2}{(2.55\cdot2)^2\cdot D} \Rightarrow t \approx 2.46\ \text{hours}$

To reach a junction depth of **4 μm**, the diffusion should be carried out for approximately **2.46 hours**.

Numerical Solution

The diffusion process continues for 2.46 hours, which corresponds to the time required to achieve a junction depth of 4 microns based on the analytical solution.

The predefined values are:

$\sqrt{D} = 0.5\ \mu m\ \text{hr}^{-1/2}$

$t = 2.46\ \text{hours} = 8856\ \text{seconds}$

Since the total time is 2.46 hours, the time step is selected as:

$\Delta t = \frac{1}{t} = \frac{1}{8856} \approx 0.0001$ seconds

$\Delta x = 0.01\ \mu m$

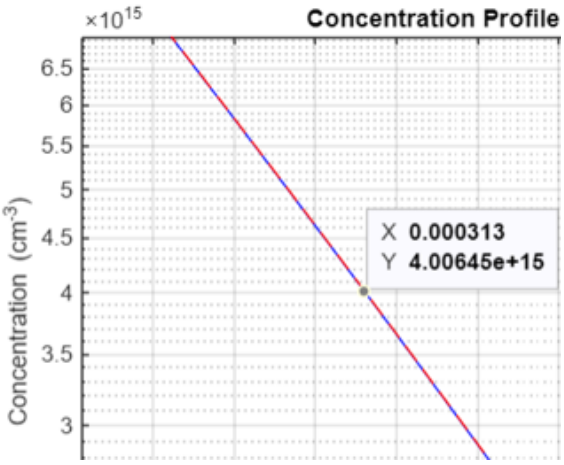


Figure 13: Numerical Solution of Diffusion Question

Our result shows that the junction depth is 3.13 microns, which is very close to the expected value of 4 microns given in the question. This small difference can happen due to rounding or small changes in simulation settings. Overall, our simulation result is accurate and matches the expected behavior of the diffusion process.

3.3 Comparison of Numerical and Analytical Solutions

In this section, dopant diffusion from a constant source was simulated both numerically and analytically to validate the accuracy of the model. The simulation assumes a fixed surface concentration of Cs =$2x10^{19}$ cm^{-3} based on boron diffusion at 1150°C.

The numerical solution was obtained using the finite difference method applied to Fick's second law, while the analytical solution uses the complementary error function:

$$C(x,t) = C_s \operatorname{erfc}\left(\frac{x}{2\sqrt{Dt}}\right) \qquad (6)$$

As shown in Figure 12, both results align closely, confirming that the numerical method accurately models constant source diffusion.

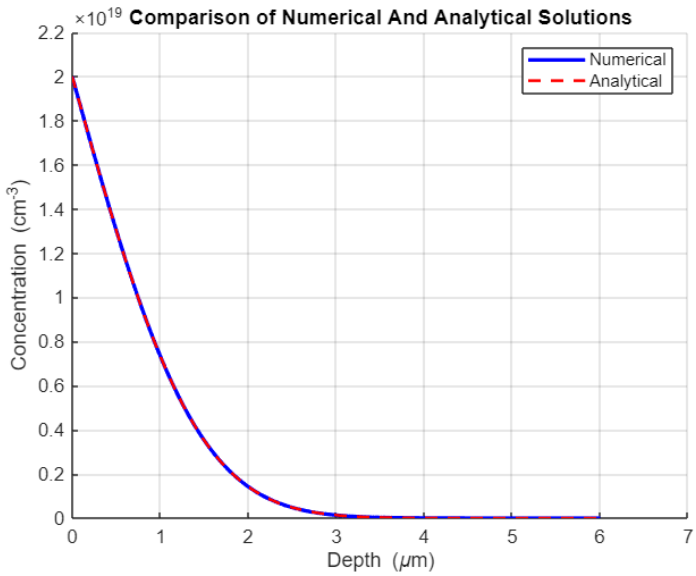


Figure 14: Comparison of Numerical and Analytical Solutions.

3.4 Numerical Instability

Previously, the numerical factor $\frac{D\Delta t}{\Delta x^2}$ used in Equation 10 was consistently set to 0.5.

$$C_i^{t+1} = \frac{1}{2}(C_{i-1}^t + C_{i+1}^t) \quad (10)$$

If this value rises above 0.5, the numerical method begins to become unstable. When the threshold is exceeded, the concentration values fluctuate and reduce the reliability of the diffusion results.

To explore this behavior, the stability parameter $\frac{D\Delta t}{\Delta x^2}$ was raised to 0.6 and in Equation 15, and the resulting effects were examined. [1]

$$C_i^{t+1} = 0.6(C_{i-1}^t + C_{i+1}^t) \quad (19)$$

$$C_i^{t+1} = 0.7(C_{i-1}^t + C_{i+1}^t) \quad (20)$$

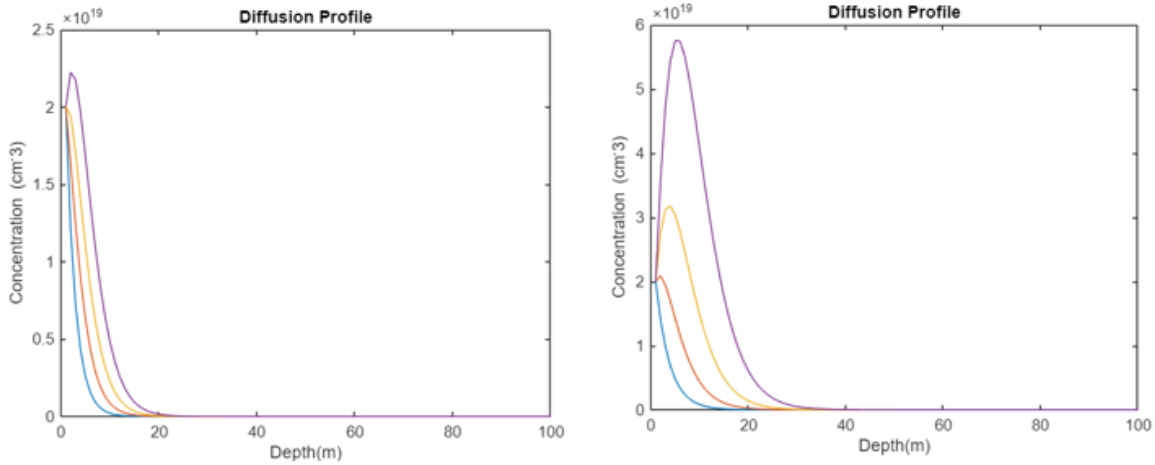


Figure 15: D$\Delta t/(\Delta x$^2) set to 0.6 (left) and 0.7 (right). (Constant Source Diffusion)

The concentration values at some points become higher than the initial surface concentration. Physically, this is not possible because in constant source diffusion, the surface concentration should always be the highest value.

Also in drive-in diffusion, using $\frac{D\Delta t}{\Delta x^2}$ = 0.6 and 0.7 make the solution unstable. As seen in the graphs, concentration profile becomes unrealistic.

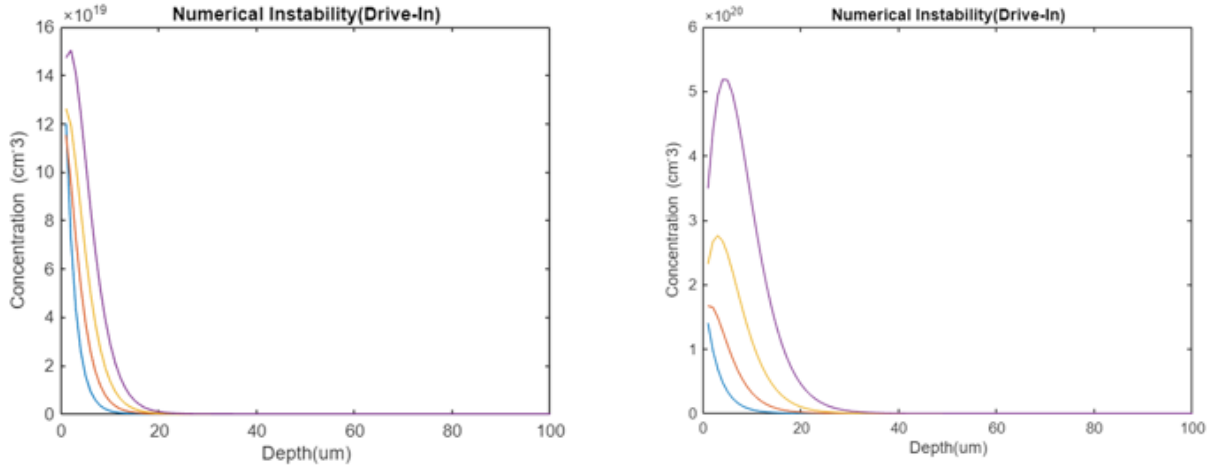


Figure 16: D$\Delta t/(\Delta x$^2) set to 0.6 (left) and 0.7 (right). (Drive-In Diffusion)

In the next simulation of drive-in diffusion, the instability factor exceeds the stability limit. As a result, after around 75-time steps, the concentration values rapidly grow to unrealistic levels (e.g 10^{31}) clearly indicating numerical instability.

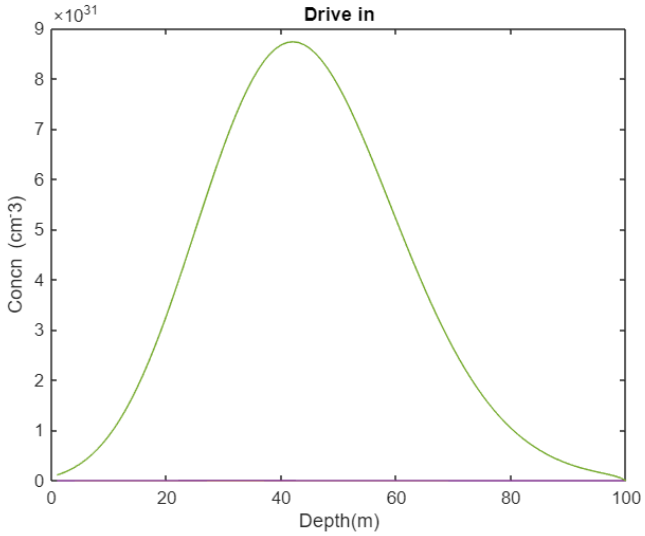


Figure 17: Unstable Condition of Drive-In Diffusion after 75 steps.

3.5 Concentration Dependent Diffusion: 1D Results

To understand better this concept, we will solve a sample problem that involves calculating the effective diffusion coefficient of arsenic in silicon at 1000°C, using two different doping levels.[29]

- **Example:**

Calculate the effective diffusion coefficient of arsenic in silicon at 1000°C for two different box-shaped doping profiles grown by silicon epitaxy:

- One doped at 1×10^{18} cm^{-3}
- The other doped at 1×10^{20} cm^{-3}

Use the concentration-dependent diffusion model for arsenic.

Given

Temperature $T = 1000$°C $= 1273$ K

Intrinsic carrier concentration at 1000°C: $n_i = 7.14 \times 10^{18}$ cm^{-3}

Arsenic diffusion parameters in silicon:

- $D_1 = 0.01\ \text{cm}^2/\text{s}\ \ E_1 = 3.44\ \text{eV}$
- $D_2 = 310\ \text{cm}^2/\text{s}\ E_2 = 4.15\ \text{eV}$

$k = 8.617 \times 10^{-5}\ \text{eV/K}$

Solution

$C = 1 \times 10^{18}\ \text{cm}^{-3}$ This is less than n_i so:

$$D_{\text{As}} = D_1 \cdot e^{-E_1/kT} + D_2 \cdot e^{-E_2/kT}$$

$$= 0.01 \cdot e^{-3.44/(8.617\times10^{-5}\cdot1273)} + 310 \cdot e^{-4.15/(8.617\times10^{-5}\cdot1273)}$$

$$= 2.67 \times 10^{-16} + 1.17 \times 10^{-15} = 1.43 \times 10^{-15}$$

$C = 1 \times 10^{20}\ \text{cm}^{-3}$ This is greater than n_i so the second term is scaled:

$$D_{\text{As}} = D_1 \cdot e^{-E_1/kT} + D_2 \cdot e^{-E_2/kT} \cdot (\frac{C}{n_i})$$

$$= 2.67 \times 10^{-16} + 1.63 \times 10^{-14} = 1.66 \times 10^{-14}$$

We created a simulation (figure 18) to observe the effect of concentration-dependent diffusion, using parameters from a sample problem. The initial state is a sharp, **box-like shape** showing high dopant concentration near the surface. As diffusion progresses, dopants spread deeper. A key aspect of this simulation is the concentration-dependent diffusion coefficient, D(C), for arsenic. As shown in the formula:

$$D_{\text{As}} = D_1 \cdot e^{-\frac{E_1}{kT}} + D_2 \cdot e^{-\frac{E_2}{kT}} \cdot max(1, \frac{C}{n_i})$$

the diffusion rate changes based on the concentration relative to the intrinsic carrier concentration (n_i).

In the High C regime (when C > ni), the $max\left(1, \frac{C}{n_i}\right)$ term scales D(C) proportionally to C, boosting diffusion and causing rapid spreading. In the Low C regime (when C < ni), the $max(1, \frac{C}{n_i})$ term becomes 1.

The next figure compares two different diffusion profiles for arsenic in silicon.
The blue curve shows diffusion that is not dependent on the concentration. In this case, the concentration slowly decreases with depth, creating a **Gaussian-like profile**.
The red curve represents the diffusion due to the concentration. In this case, the surface remains almost flat with high concentration, and the profile drops sharply at a certain depth, shows a **box-like shape.**

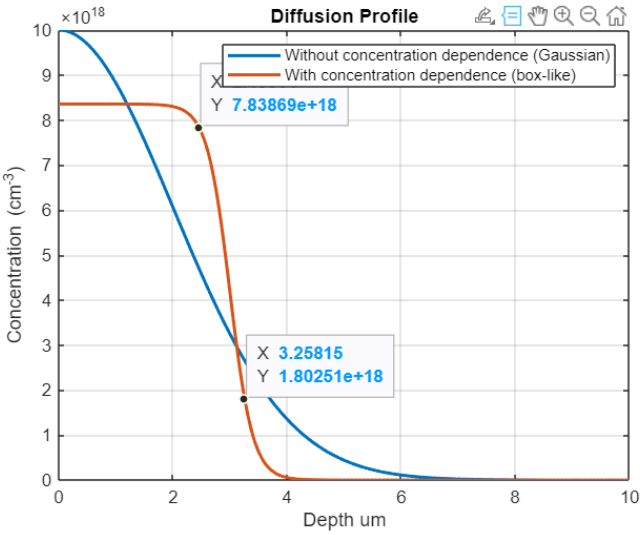


Figure 18: Concentration Dependent Diffusion vs No Concentration Dependent Diffusion

3.6 Electric Field Effects: 1D Results

In this section, the one-dimensional diffusion of the dopant atoms in silicon is examined under the electric field. The electric field effect was accounted for by adding a flux, linked to concentration.

$$h = 1 + \frac{c}{\sqrt{\mathbb{C}^2+4n^2}} \qquad (17)$$

This flux changes the diffusion rate depending on the intensity of the dopant. Thus, the model can show faster diffusion in high-doped areas. The simulation begins with a Gaussian-shaped initial concentration profile and follows how it spreads over time through finite-differences. In addition, reflective boundary conditions are applied at both ends.

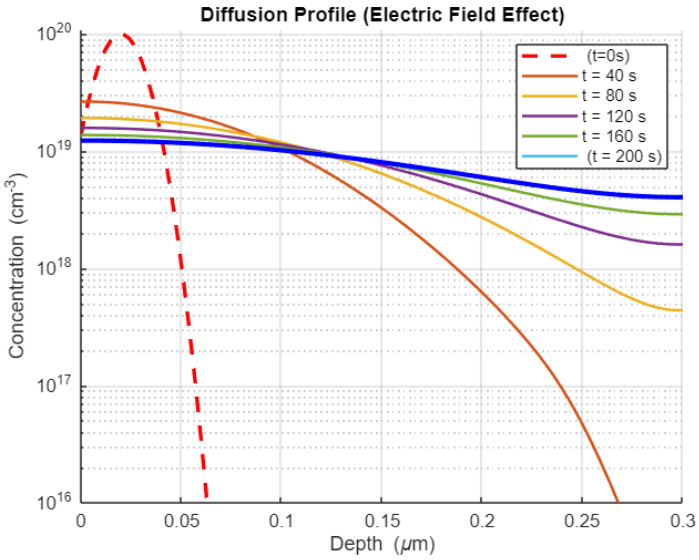


Figure 19: Diffusion Profile (Electric Field Effect)

In the initial profile, there is a peak value of $1x10^{20}$ cm^{-3} at a depth about 0.02 µm.

At t=40 the peak begins to expand. At a depth of 0.1 µm, the concentration is about $3x10^{18}$ cm^{-3}, means dopants are moving deeper.

At t=80 and t=120 the profiles are flattened, especially between 0.05 - 0.2 µm, with concentrations staying between $1x10^{18}$ and $1x10^{19}$ cm^{-3}. This reveals that the electric field increased diffusivity.

4. Results: 2D

Two-dimensional (2D) simulations help extend the diffusion of the dopant atom to more complex shapes. In this way, both vertical and horizontal dopant distribution can be examined in semiconductors. While 1D models only calculate in one direction, 2D models show detailed interactions and more realistically reflect dopant propagation on real devices, such as BJT transistors.

4.1 Emitter Concentration of Bipolar Junction Transistors (BJT)

In order to better understand this part, it is important to first examine the doping profile of a bipolar junction transistor (BJT). The followings show how the dopant concentration in the emitter, base and collector zones of the device has changed.

In a typical NPN BJT, the emitter region is heavily doped, with concentrations close to 10^{20} cm^{-3}. The base region is moderately doped, around 10^{17}–10^{18} cm^{-3}. When the device undergoes thermal treatment at 1000 °C for 30 minutes, diffusion causes significant changes in this profile. The emitter concentration decreases as dopants diffuse deeper.

Such changes directly affect the electrical behavior of the transistor. [27]

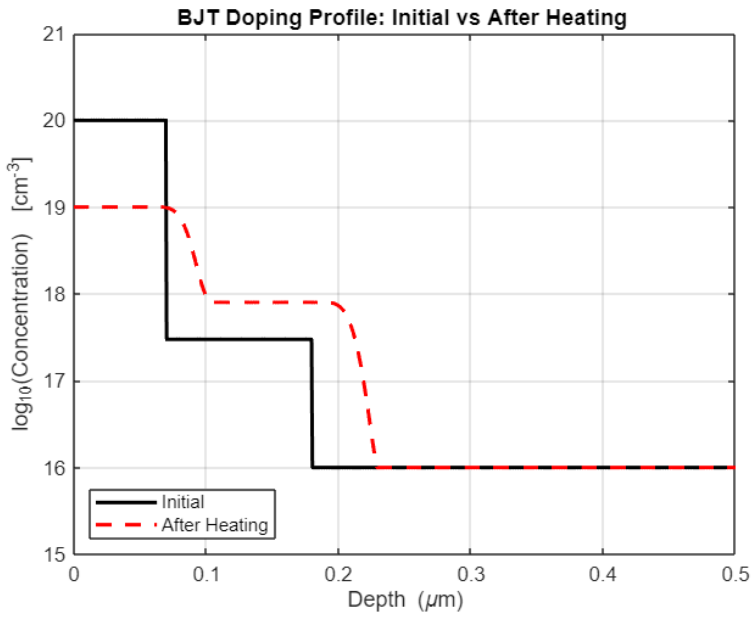


Figure 20: Doping profile for standard silicon transistor (NPN) is imported to MATLAB

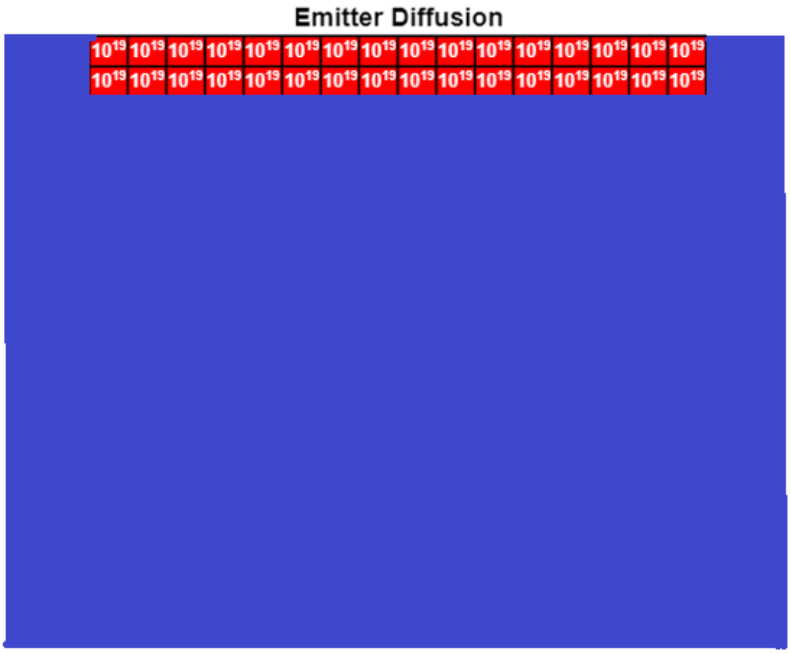


Figure 21: 2D Matrix of an Emitter in BJT's

To better explain how the emitter doping profile in Figure 20 looks in real space, Figure 21 shows a 2D matrix version. While Figure 20 shows how the doping concentration changes with depth, Figure 21 shows where the high doping (about 10^{19}–10^{20} cm^{-3}) is located on the surface of the silicon. This 2D view helps us see the starting condition of high surface concentration, which is important for the first step of emitter diffusion, called predeposition. These two figures together give both a 1D and spatial 2D understanding of how the emitter is formed in a BJT.

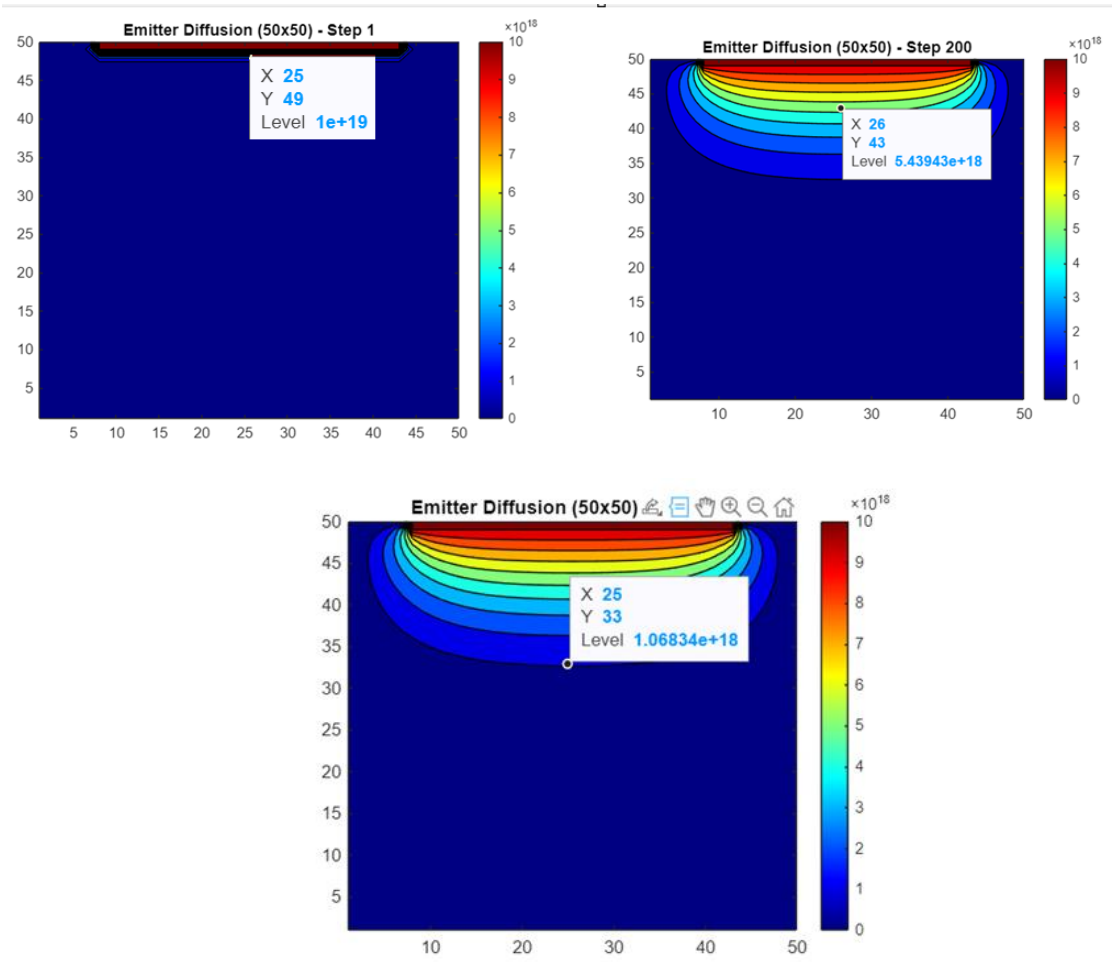


Figure 22: Emitter Diffusion as a function of time.

Figure 22 shows how emitter diffusion progresses over time in the silicon substrate. A two-dimensional section of the material is modeled with a grid of 50x50. Initially (Step 1), the dopant was placed on the top surface and a high concentration of about 1x 10^{19} cm^{-3} was set. The simulation runs for 200 time steps, each of which was 0.1s.

As the diffusion process advances, the dopants begin to move into the substrate. ranging from red (high) to blue (low) indicating how dopants spread downward. At this stage, the concentration at a selected inner point has already decreased to approximately 5.4x10^{19} cm^{-3}, showing significant diffusion into the material.

In the later stages, the profile becomes smoother, and the contributions reach deeper into the substrate. At this point, the concentration drops to about 1.07x10^{18} cm^{-3}.

5. Results: 3D

Drawing diffusion results as a 3D surface offers much more context compared to a 2D color map. In the Z-axis, we see the absolute value of concentration and the continuity of gradients at the same time. It also makes it easier to read the height difference numerically between different points, pick up sections and extract and compare profiles. Therefore, 3D is more efficient than 2D in terms of both analysis and presentation.

The current 2D emitter concentration code has been converted to 3D to use the same grid and concentration matrices. The coordinate points selected as references in the 2D figure were also marked on the 3D surface. The same concentration values were obtained at the same locations in both approaches.

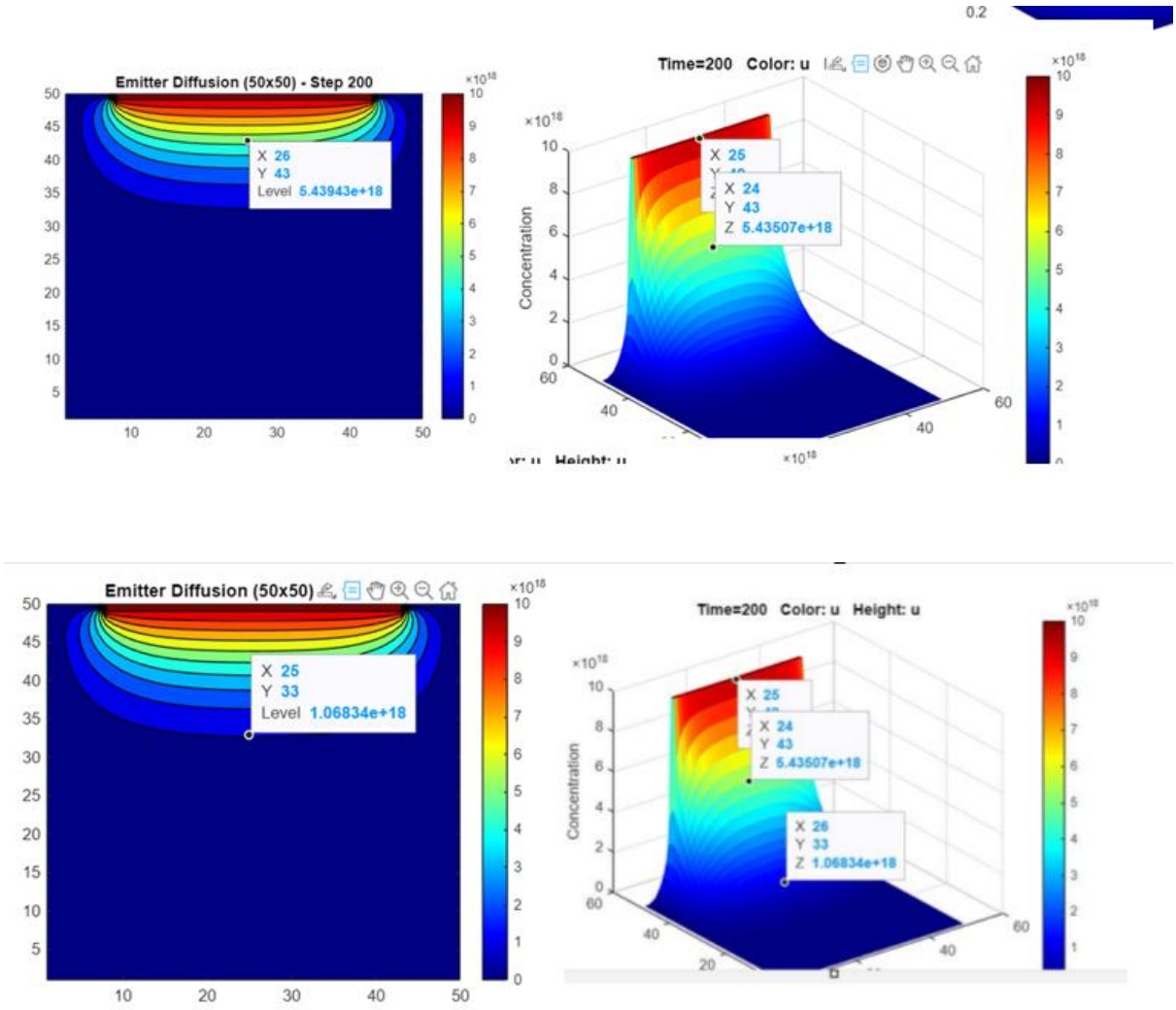


Figure 23: Validation of 3D Emitter Diffusion Results Using Selected Coordinate Points

Another way to create 3D plots is MATLAB pdeModeler tool. This advanced tool creates 3D figures in seconds. To create following 3D graph, the MATLAB PDE Modeler tool, which can simulate 2D diffusion in different ways, was used.

First, the emitter zone is located on the upper surface of a rectangular area, and a high initial doping concentration is assigned to this area. Later, PDE Modeler created a mesh network with smaller elements in regions close to the emitter. As the simulation progressed, the contributions began to spread from the emitter to the silicon. [4]

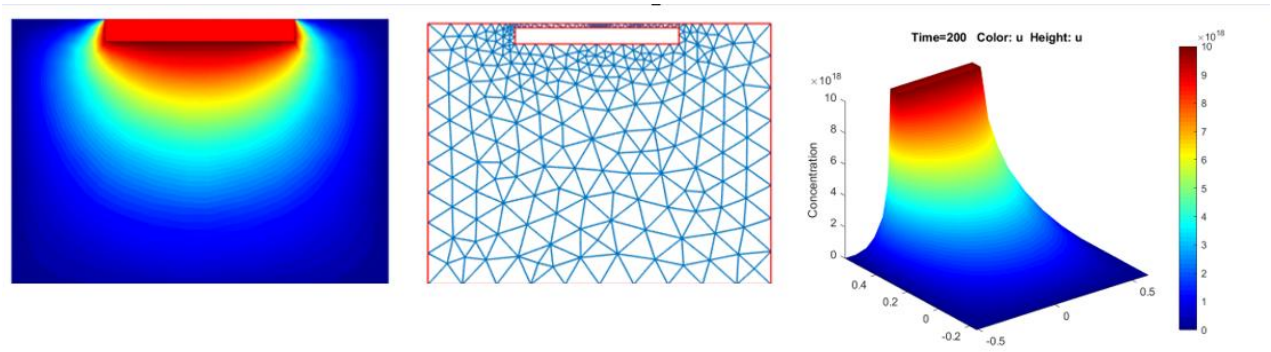


Figure 24: Solving PDE using MATLAB's pdeModeler.

Finally, the results are visualized in 3D (Figure 23). The height and colour in this graph show the concentration. The red peak marks high intensity in the emitter, while the softness of colour transitions indicates that the diffusion spreads both deep and sideways over time. This 3D graphic exposes the emitter diffusion process in a clear and detailed way.

6. Results: Benchmarking

- Laptop Specification

The laptop used in this study has an Intel Core i5 (8th generation) processor, 16 GB RAM and an Intel UHD Graphics 620 graphics card.

```
Parameters are set to:

Number of tests: 1
Number of equations to solve (problem size) : 20000
Leading dimension of array                  : 20000
Number of trials to run                     : 5
Data alignment value (in Kbytes)            : 4

Maximum memory requested that can be used=3200404096, at the size=20000

================== Timing linear equation system solver ==================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
20000  20000  4      54.693     97.5291  2.657681e-010 2.352630e-002   pass
20000  20000  4      64.986     82.0818  2.657681e-010 2.352630e-002   pass
20000  20000  4      57.225     93.2128  2.657681e-010 2.352630e-002   pass
20000  20000  4      52.244     102.0997 2.657681e-010 2.352630e-002   pass
20000  20000  4      54.248     98.3286  2.657681e-010 2.352630e-002   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
20000  20000  4       94.6504  102.0997

Residual checks PASSED

End of tests
```

Figure 24: Benchmark using Linpack Xtreme.

A 3 GB benchmark test was performed with Linpack Xtreme. The test was repeated five times to ensure the results were reliable. The system has generally shown stable performance, with a value above 100 GFlops at best, but there have been slight differences between trials. [5]

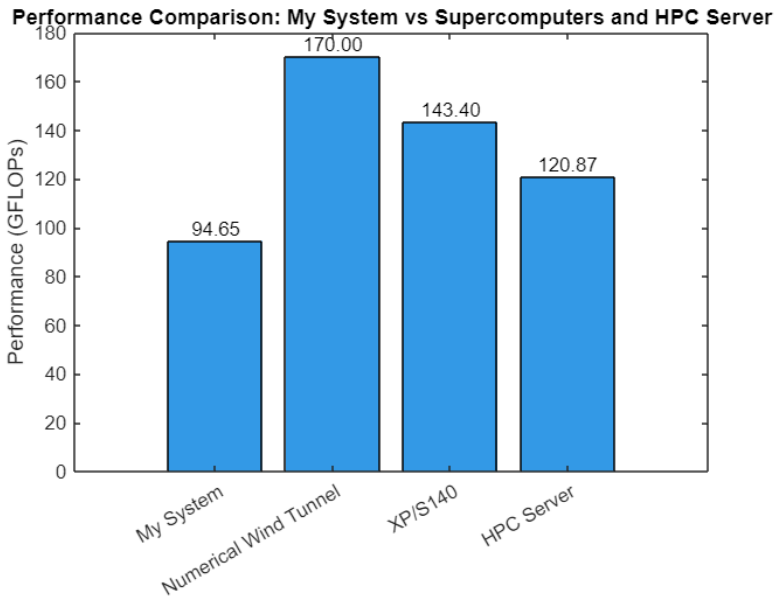


Figure 25: GFLOP comparison

The bar chart above compares the computing power of the four systems in terms of GFLOPs (Giga Floating Point Operations per Second). Our laptop reaches 94.65 GFLOPs, while "HPC Server (TU Dublin)" has achieved 3 GB, 5 repetitions) with the same Linpack Xtreme test 120.87 GFLOPs. For comparison, older supercomputers such as "Numerical Wind Tunnel (1995)" and "XP/S140 (1995)" recorded the values of 170.00 and 143.40 GFLOPS respectively. [5] [6].

6.1 Comparison of HPC Server and Laptop on 2D Results

System	HPC Server (DELL PowerEdge R750xs Rack Server)	Laptop
RAM	128 GB DDR 4 - 2666	16 GB DDR4- 2400
Storage	12 TB	512 GB
Processor	2× Intel Xeon Silver 4310, 24 cores, 2.1 GHz	Intel Core i5 (8th Gen), 8 cores, 1.7 GHz

Table 2: Comparison of the machines used in this project.

The laptop used in this study has an Intel Core i5 (8th generation) processor, 16 GB RAM and Intel UHD Graphics 620 hardware. It is more suitable for daily operations and light applications.

In contrast, the HPC server (DELL PowerEdge R750xs) is designed for heavy workloads. It has two Intel Xeon Silver 4310 processors and offers a total of 24 cores and 48 threads. It also has 128 GB of RAM and 12 TB of storage. Thanks to these features, it has provided much higher processing power, memory capacity and storage, making it suitable for solving large-scale PDE problems.

In the next figure we see, the HPC server completed the 2D emitter simulation in about 450 seconds (7.5 minutes), while the laptop required 1348 seconds (22.4 minutes) for the same process. This result shows that the HPC server is about three times faster. The main reason for the high performance is the greater number of cores and a higher clock speed; therefore, numerical calculations are much more efficient.

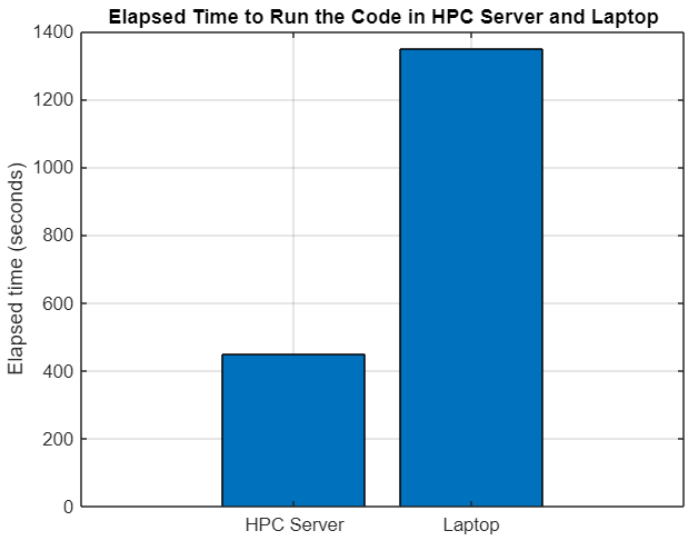


Figure 26: Comparison of Elapsed Times (HPC Server and Laptop)

The bar chart compares the run times of the same MATLAB code on the HPC server and laptop. The HPC server has completed much faster as it has more cores and better parallel processing support.

6.2 Elapsed Time Using Different MATLAB Versions

For further investigation, run times were examined in MATLAB 2018a and MATLAB 2024b versions. Results showed that the code was completed in MATLAB 2024b in 645 seconds and MATLAB 2018 in 670 seconds.

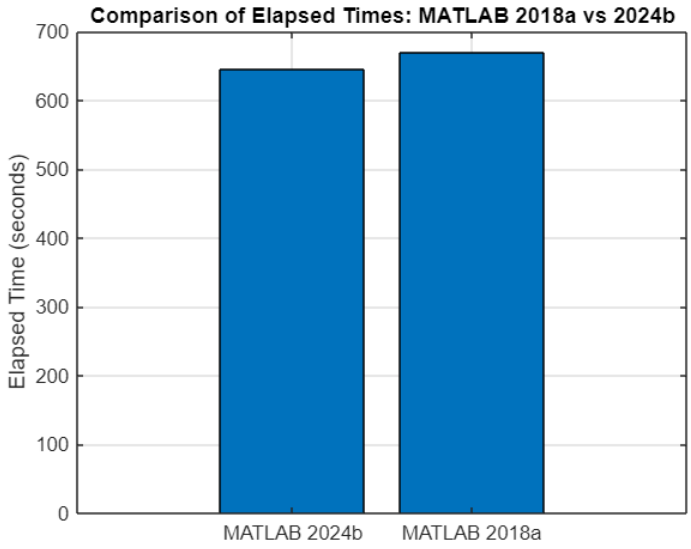


Figure 27: Elapsed Time (MATLAB 2024b vs MATLAB2018a)

The faster results in MATLAB 2024b can be explained by the fact that the new versions are more optimized and the calculations are better done with the software improvements with the new updates. Although the time difference is not very large, it shows that software updates can also improve performance without changing the hardware.

6.3 Benchmarking MATLAB Against C and C++

2D emitter diffusion simulation was written in MATLAB before. To achieve higher speed and compare results, the code has been converted to C and C++ by the MATLAB Coder tool. This tool automatically generated C and C++ source files from the original MATLAB code.

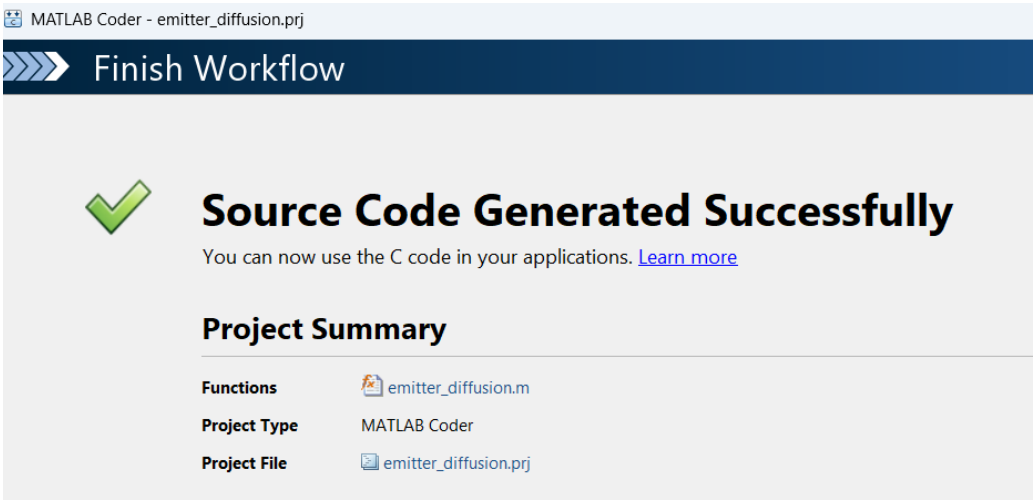


Figure 28: MATLAB Coder Add-on

After the conversion, C and C++ codes were compiled and executed separately. Thus, working times in MATLAB, C and C++ environments were compared. C and C++ simulations are run in Microsoft Visual Studio Code 2022. This environment has been selected to ensure consistency and reliability in the testing process. [10]

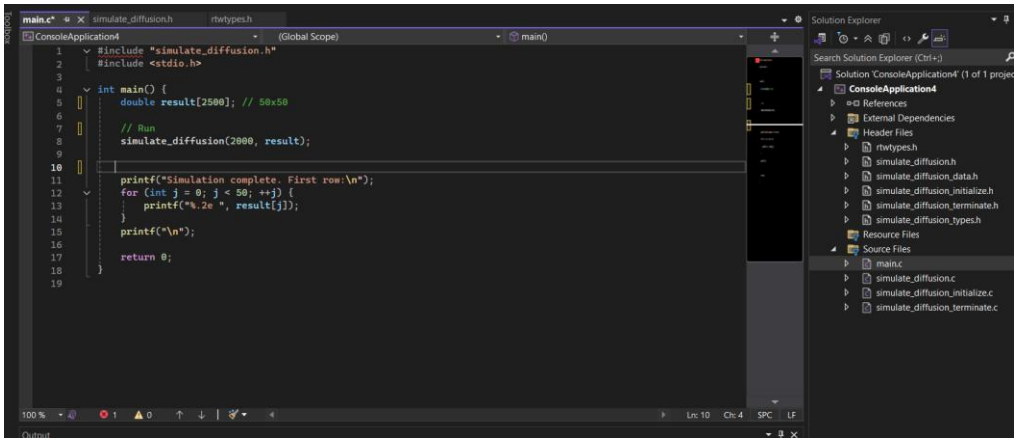


Figure 29: MS Visual Studio Code 2022 Interface

The following bar chart shows the run times of the same 2D emitter diffusion simulation in three different programming environments: MATLAB, C and C++.

According to the results, MATLAB is the fastest and is completed in about 448 seconds. C++ lasted 585 seconds, while C lasted 683 seconds at its slowest. Normally C++ is expected to be faster than MATLAB , in which case MATLAB performed better.

However, since this simulation uses matrix operations heavily, MATLAB performs better because it is optimized for matrix-based calculations.

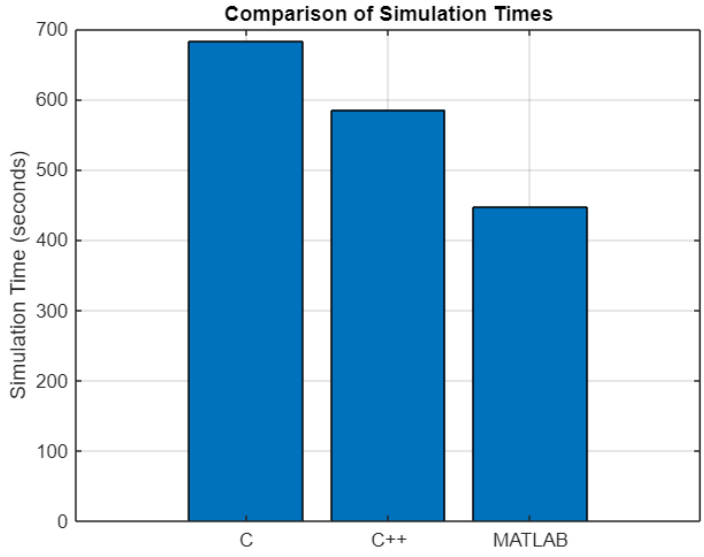


Figure 30: C vs C++ vs MATLAB

7. Machine Learning Approaches for Junction Depth Prediction

Machine learning is a set of methods that allows computers to learn patterns from data and make predictions/decisions without being explicitly programmed. In this part, we compare two methods to predict the junction depth in a silicon wafer:

- a physics-based approach using diffusion equations, and
- a machine learning model trained on pre-calculated data.

We used boron diffusion parameters from experimental graphs and applied the standard analytical formulas for pre-deposition and drive-in steps.
The same input values were also given to a trained regression tree model.

The predicted junction depths from both methods were very close.
This shows that machine learning can accurately estimate junction depth when trained with reliable physics-based data.

To perform a detailed analysis, we proceeded with the example question below.

Calculate the junction depth and the total amount of dopant introduced after a boron predeposition performed at 950 °C for 30 minutes in a neutral ambient.

Assume the substrate is n-type silicon with
$C_B = 1.8 \times 10^{16}\ cm^{-3}$
and the Boron surface concentration is
$Cs = 1.8 \times 10^{20}\ cm^{-3}$.

A drive-in diffusion is now performed at 1050 °C for 60 minutes.
Calculate the new junction depth.

Solution

The sample diffusion question was solved in MATLAB, and the following results were obtained: pre-deposition junction depth, total dopant amount (Q), and drive-in junction depth.

```
--- Diffusion Question 3  ---
Pre-deposition junction depth: 0.1717 µm
Q: 5.745e+14 cm^-2
Drive-in junction depth: 0.3089 µm
```

The MATLAB code was updated to solve the diffusion question at different temperatures and with various C_S and C_B values.

Using this approach, a large dataset was generated showing the calculated junction depths for each combination of inputs.

	A	B	C	D	E	F	G
1	Temperature_C	Time_min	Cs	ND	xj_micron	Q_cm2	
2	943.6350297	114.578574	3.93E+20	3.39E+16	0.29464561	2.35E+15	
3	889.0046601	27.1593972	1.23E+20	4.46E+16	0.09433124	2.6E+14	
4	1000.278753	87.8879836	1.08E+20	4.88E+16	0.36104247	8.89E+14	
5	1058.11066	33.3573022	1.73E+20	1.73E+16	0.36836432	1.31E+15	
6	926.0605607	67.7232075	2.73E+20	2.16E+16	0.20621041	1.14E+15	
7	1002.963224	25.3443247	2.17E+20	2.47E+16	0.217526	9.75E+14	
8	964.0174961	96.3693558	1.80E+20	3.06E+16	0.29997171	1.14E+15	
9	998.1036422	15.1095454	3.43E+20	1.68E+16	0.17044899	1.15E+15	
10	866.2628982	114.377409	4.86E+20	4.23E+16	0.17816292	1.76E+15	
11	926.1534423	20.7439325	3.74E+20	2.76E+16	0.11467914	8.63E+14	
12	880.5095587	64.4694601	1.14E+20	4.64E+16	0.13526573	3.47E+14	
13	914.6949954	82.8774513	2.25E+20	3.08E+16	0.20748482	9.75E+14	
14	986.6775698	30.3339901	4.88E+20	4.10E+16	0.21355619	2.11E+15	
15	1084.874735	108.431009	3.39E+20	4.69E+16	0.78027607	5.54E+15	
16	872.1231255	31.5581149	1.18E+20	2.30E+16	0.09332868	2.36E+14	
17	947.1693224	39.8483935	4.31E+20	2.43E+16	0.18228587	1.56E+15	
18	920.2336274	69.6965691	1.56E+20	4.21E+16	0.18726265	6.41E+14	
19	868.6376609	118.557563	4.09E+20	1.79E+16	0.19273056	1.54E+15	
20	851.3805293	99.7007571	3.83E+20	3.92E+16	0.14360454	1.13E+15	
21	1042.817587	18.1449117	2.43E+20	1.46E+16	0.25250522	1.22E+15	
22	1065.775856	78.562794	2.32E+20	1.25E+16	0.61781408	2.84E+15	
23	927.7455804	45.7701654	3.92E+20	3.55E+16	0.16967383	1.36E+15	

Figure 31: Junction Depth Values Calculated for Different Combinations

To develop a predictive model for junction depth, we used MATLAB's **Regression Learner App**.

This useful tool allows users to train and compare various regression models.

We imported our dataset which included temperature, time, surface concentration, background concentration to find **JunctionDepth** and **Dose** as the response variables.

Multiple models were trained and compared automatically. [13]

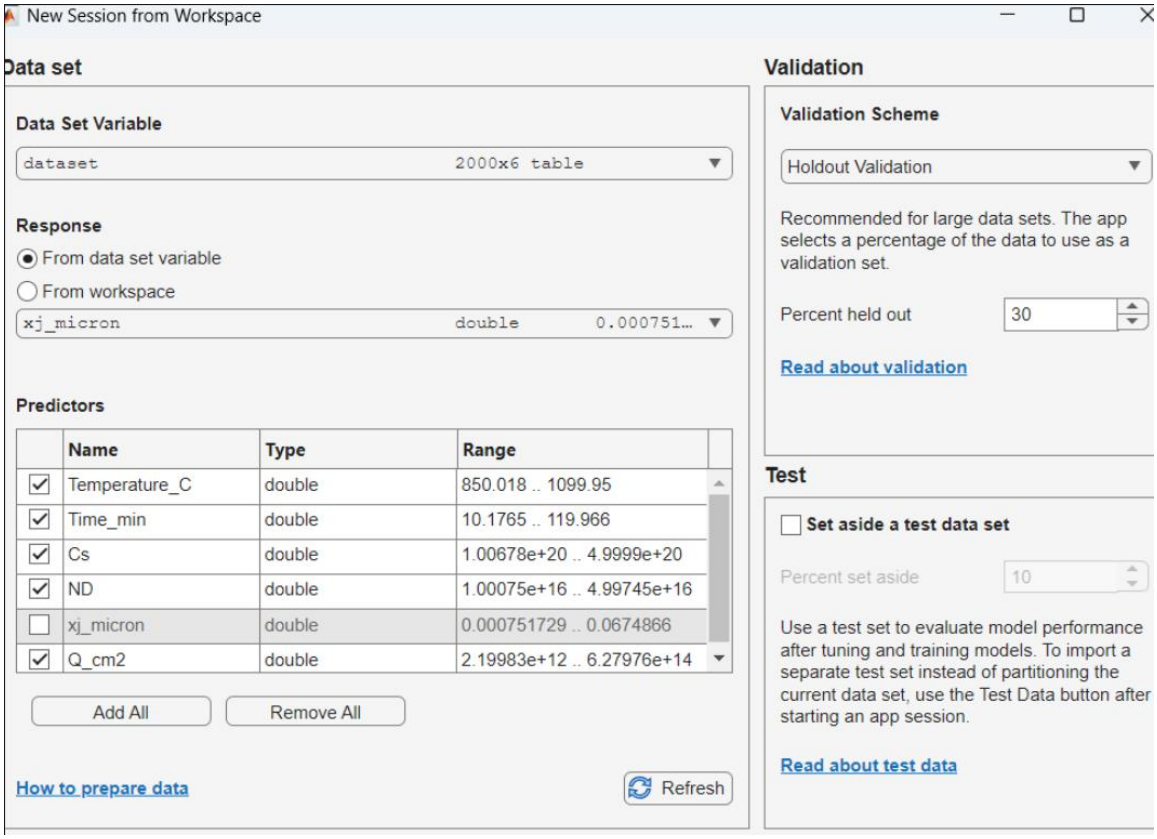


Figure 32: Creating a New Session in Regression Learner App

The figure below shows the validation RMSE (Root Mean Squared Error) values for different regression models trained in MATLAB's Regression Learner App. Each model was evaluated using the same dataset.

Among all models, the **MATERN 5/2 GPR** model achieved the **lowest RMSE (Validation) of 0.012533**, shows the most accurate predictions. The setting Validation Scheme divides the dataset into two using Holdout Validation: the model is trained at 70%, while the 30% section is divided into an 'unseen' set of validations.

"Unseen validation" means that the model is evaluated on data that it never uses while being trained. So, part of the data (e.g., 30%) is separated, the model learns only in the remaining part; then in this reserved (unseen) part, the generalization ability is measured by looking at its performance. It stops being a simple lookup table and demonstrates the true **train-and-learn** principle of machine learning.

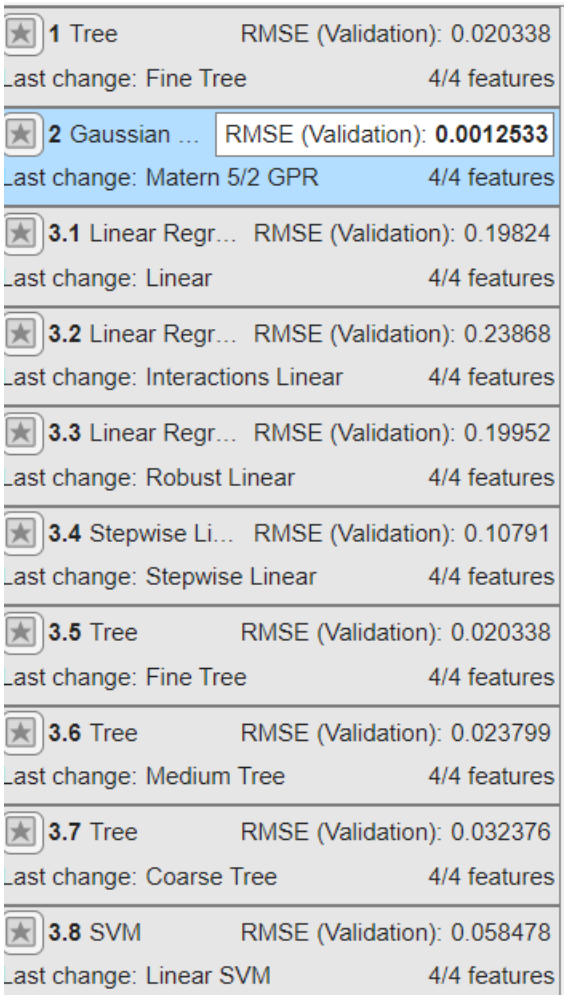


Figure 33: Models.

Figure 34 shows the response plot of the MATERN 5/2 GPR model trained using the MATLAB Regression Learner App. In the graph, the actual depth of junction values (blue dots) and the predicted values (yellow dots) are compared.

The estimated values are pretty close to the actual values in all regions. Only very small differences have been observed, indicating that the GPR model captures the relationship. between input variables and junction depth with high accuracy.

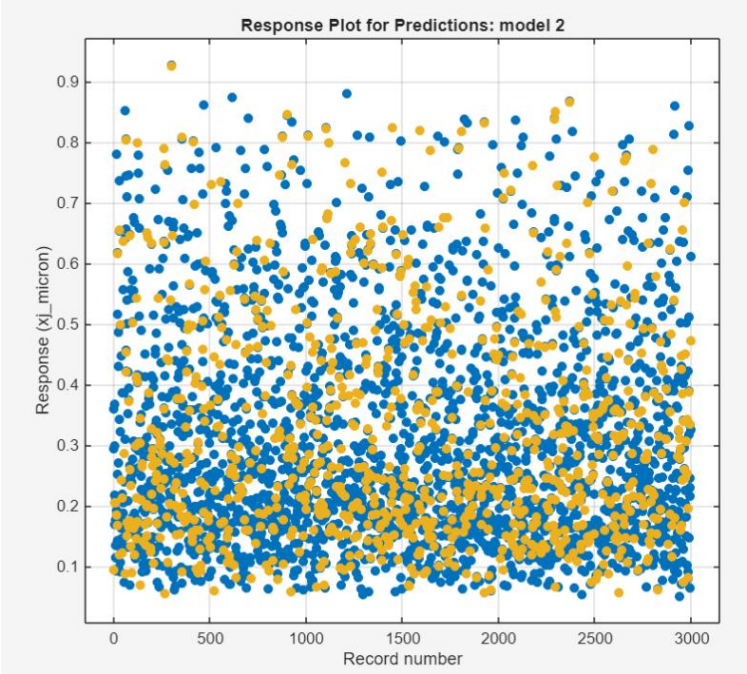


Figure 34: Response plot of the Fine Tree model

After the model was trained, the best performing model (trainedModel) was transferred to the MATLAB workspace. In this way, new data can be predicted without retraining the model. By creating a new input table consisting of temperature, time, Cs and Cb values, the predictFcn function was used and the depth of the junction and dose were estimated.

```
Structure 'trainedModel' exported from Regression Learner.
To make predictions on a new table, T:
  yfit = trainedModel.predictFcn(T)
```

The junction depth obtained from the diffusion equations is compared with the value predicted by the machine learning model in the next step.

```
Case 1 | T=902.7°C, t=78.7 min, Cs=1.12e+20, ND=3.84e+16 | z-dist=0.45
   Physics xj = 0.177025 um | ML xj = 0.175316 um | diff = 0.001709 um (0.97 %)
Case 2 | T=1063.7°C, t=117.2 min, Cs=2.03e+20, ND=1.10e+16 | z-dist=0.39
   Physics xj = 0.743915 um | ML xj = 0.744226 um | diff = 0.000311 um (0.04 %)
Case 3 | T=975.6°C, t=47.3 min, Cs=3.38e+20, ND=2.40e+16 | z-dist=0.30
   Physics xj = 0.245043 um | ML xj = 0.244838 um | diff = 0.000205 um (0.08 %)
Case 4 | T=989.3°C, t=32.8 min, Cs=2.73e+20, ND=1.78e+16 | z-dist=0.31
   Physics xj = 0.230265 um | ML xj = 0.229569 um | diff = 0.000695 um (0.30 %)
Case 5 | T=1061.1°C, t=59.9 min, Cs=3.22e+20, ND=1.17e+16 | z-dist=0.45
   Physics xj = 0.534636 um | ML xj = 0.534002 um | diff = 0.000634 um (0.12 %)
```

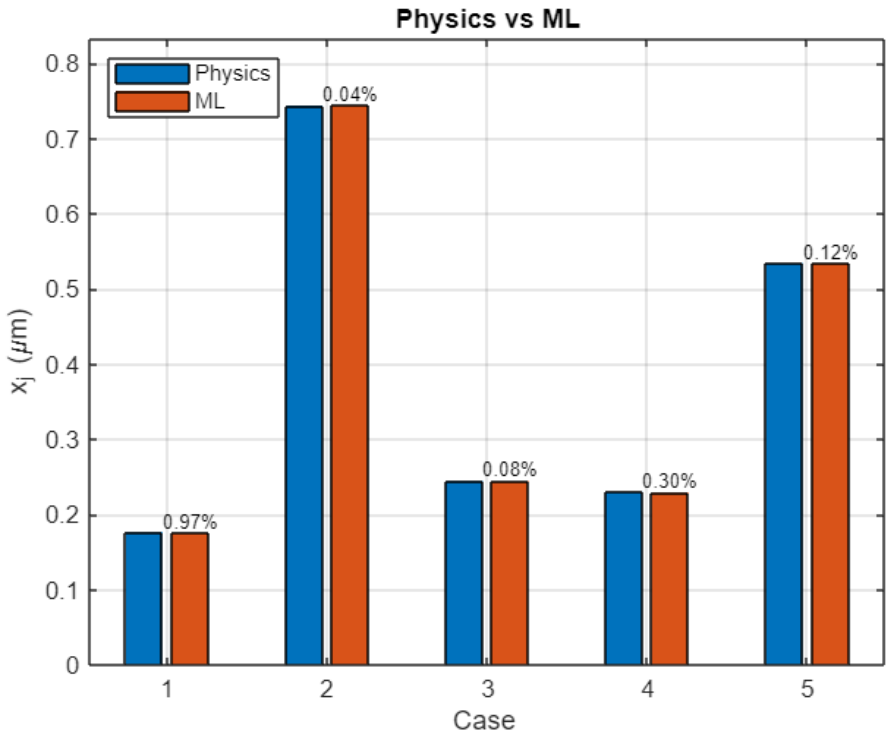


Figure 35 : Physics-based calculation vs Machine Learning Predicted Calculation

The chart shows the ML prediction (red) junction depth values side by side with the physics-solved calculations (blue) for 5 random cases. The fact that the red bars are very close to blue ones is that the model learns and generalizes the physical formula well. The percentages at the top of each bar show difference rates all below %1.

7.1 Interactive Prediction Tool for Dopant Diffusion Using MATLAB App Designer

Since the dataset used for training the models was already prepared in a previous study, we focused on creating an interactive tool to visualize and test predictions. For this purpose, we developed an app using **MATLAB App Designer**, which is a built-in environment for designing graphical user interfaces.

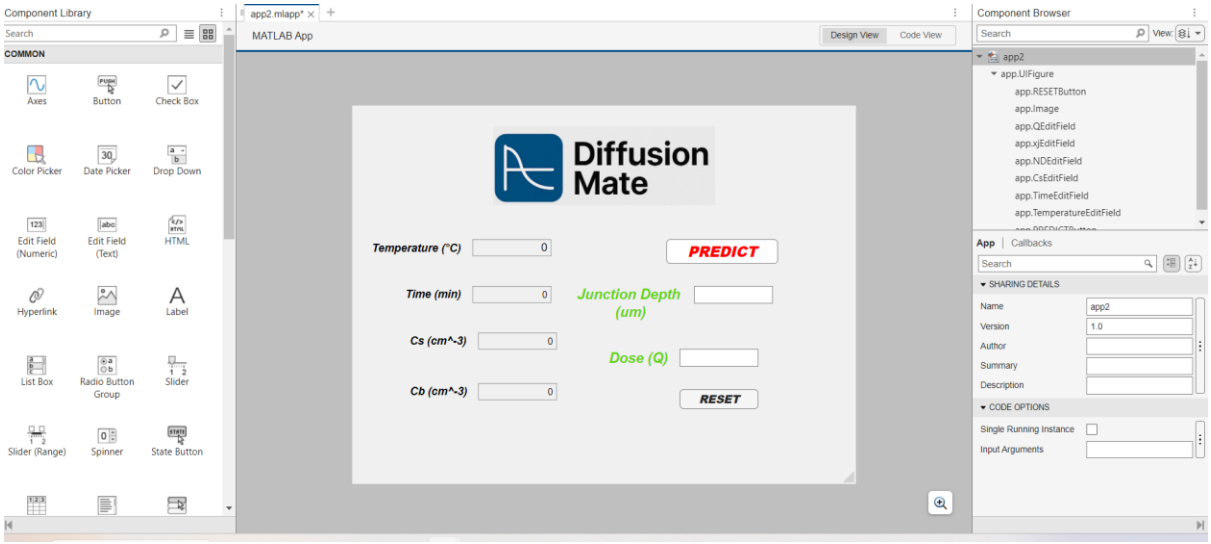


Figure 36: MATLAB App Designer – Interface

The application called **DiffusionMate** allows users to enter temperature, time, surface concentration (Cs) and background concentration (Cb) into the input parameters to estimate the junction depth. The predicted value is shown numerically, giving a different point of view in machine learning environment.

```
%  ML prediction for xj (um)
Xin = table(T, t, Cs, ND, ...
    'VariableNames', {'Temperature_C','Time_min','Cs','ND'});
xj_um = app.Model.predictFcn(Xin);

% dose Q (cm^-2)--
t_hr       = t/60;
sqrtD_um   = app.sqrtD_from_graph(T);        % um/sqrt(hr)
sqrtDt_um  = sqrtD_um * sqrt(t_hr);          % um
Q_cm2      = (2*Cs/sqrt(pi)) * (sqrtDt_um*1e-4);  % 1 um = 1e-4 cm

%  Update outputs
app.xjEditField.Value = sprintf('%.6f', xj_um);
app.QEditField.Value  = sprintf('%.3e', Q_cm2);
```

Figure 37: Prediction Workflow

Before building the application, we trained our machine learning models using a prepared dataset that includes various diffusion scenarios with different temperatures, time, surface concentration (Cs), and background concentration (Nb) values. [12]

Models were trained, validated and recorded using MATLAB's machine learning environment. We continued to create an interface using MATLAB APP Designer after confirming the models were accurate.

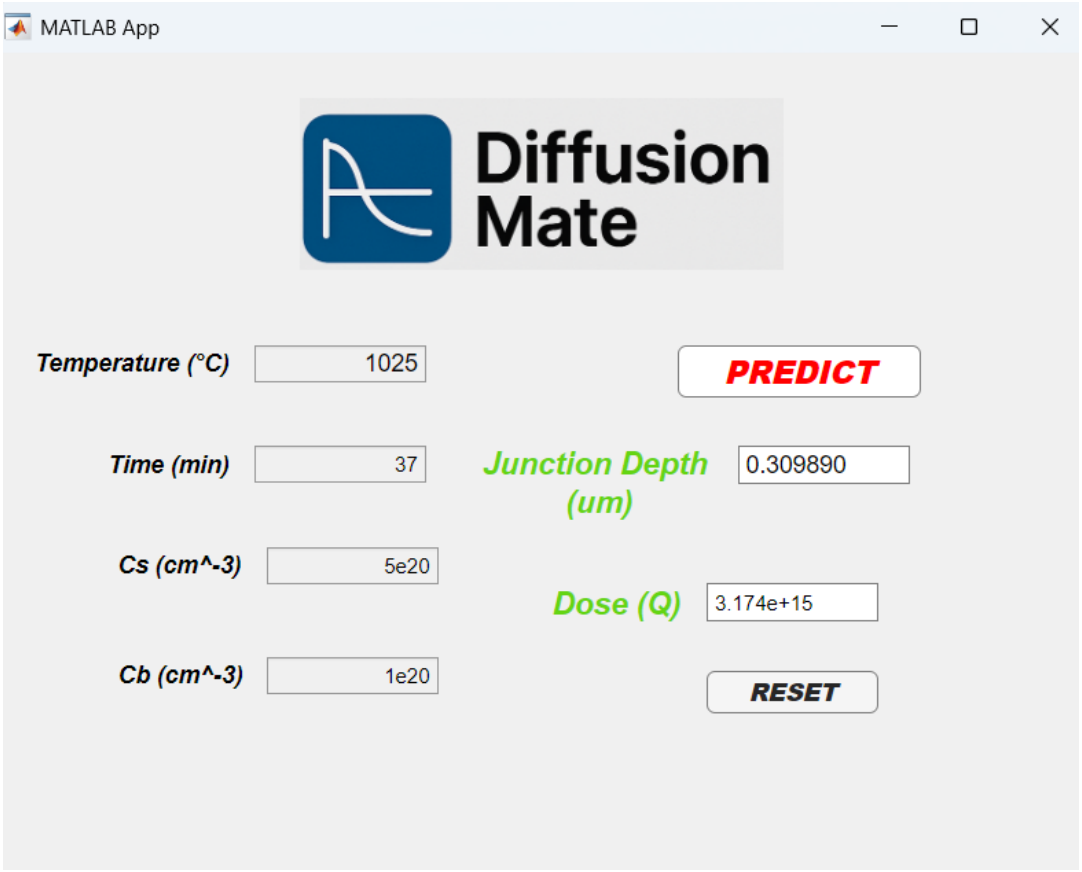


Figure 38: Junction Depth Prediction with DiffusionMate App..

Figure 38 shows the graphical user interface (GUI) of the DiffusionMate MATLAB application. This application was developed using a machine learning model to estimate the junction depth and dose in the predeposition process in semiconductors.

In the example shown in Figure 37, the user entered random values of temperature **1025°C**, **Cs ≈ 5x10^{20} cm^{-3} , Cb ≈ 1x10^{20} cm^{-3}, time = 37 minutes** and the app predicted the junction depth of **0.3 um and total dose of 3.1x10^{15}.** The app also provides a RESET button to return the starting point for different cases quickly.

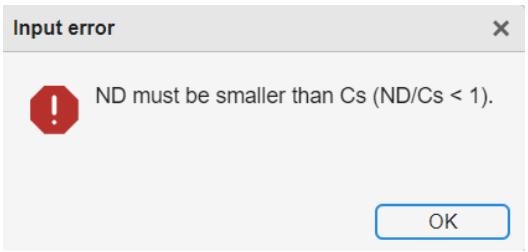


Figure 39: Concentration Validation

Figure 39 illustrates the input validation feature of the DiffusionMate application, which ensures that the surface concentration (Cs) must be greater than the background concentration (Cb).

8. Conclusions

This thesis focuses on developing and analyzing numerical solutions for dopant diffusion in silicon semiconductor structures. The finite difference method was used to solve the second law of Fick's under different conditions of diffusion. These conditions include situations involving constant-source diffusion (predeposition), drive-in diffusion, concentration-dependent diffusion, and electric field effects.

In the execution of these simulations, MATLAB has been used as the main tool and the changes of the doping concentration profiles over time have been visualized. To check the accuracy of the numerical approach, the simulation results are compared with analytical models. These models include error function profiles for constant-source diffusion and Gaussian profiles for drive-in diffusion. The close alignment between numerical and analytical results has confirmed the reliability of the finite difference method.

The performance of semiconductor devices can be enhanced by modeling a proper diffusion. Engineers can design critical device zones more efficiently by predicting how impurities spread and form junction regions. The need for a reliable diffusion model increases as device sizes decrease to the nanometer scale.

Another result of this project is the expansion of diffusion simulations into two-dimensional (2D) and three-dimensional (3D) situations. In this way, as the dopant distribution in the emitter region of BJT's is modeled on a silicon wafer, diffusion in the device structures can be examined more realistically. Multi-dimensional results have provided valuable information on both horizontal and vertical diffusion of the dopant during the formation of the junction.

Because 2D and 3D simulations require much more computational power, High-Performance Computing (HPC) resources were used to manage the workload. Running the code on the HPC server (optimized for C/C++) reduced execution time compared to a standard laptop and allowed for longer simulations that would otherwise be limited.

A machine learning approach has also been used in addition to physics-based simulations. Diffusion data from simulations were used to train the model, which was then able to quickly estimate the junction depth after predeposition step and total dose using the input parameters provided. To make this more accessible, an interactive MATLAB application has been created. Users can input parameters like concentrations and temperature in this application and receive instant results from the trained model. Integrating machine learning is a practical approach to quickly investigate different diffusion scenarios and support semiconductor production planning.

Machine learning approaches can be extended to drive-in process after predeposition. The machine learning approach can also be extended to include drive-in diffusion after pre-

deposition and made more comprehensive by adding different dopants. Methods such as deep learning, which are highly popular today, will also play an important role in modeling the diffusion process. Concentration-dependent diffusion and electric field effects will be included to further develop of the project to obtain more efficient and accurate simulations.

9. REFERENCES

[[1] J. Plummer, P. B. Griffin, and M. D. Deal, *Silicon VLSI Technology: Fundamentals, Practice and Modelling*, Prentice Hall, 2000, pp. 365–367. [Accessed: Jan. 12, 2025]

[2] S. L. Jacques and S. A. Prahl, "Diffusion Theory," Oregon Graduate Institute, 1998. [Online]. Available: https://omlc.org/classroom/ece532/class5/ficks1.html [Accessed: Jan. 12, 2025]

[3] J. Plummer, P. B. Griffin, and M. D. Deal, *Silicon VLSI Technology: Fundamentals, Practice and Modelling*, Prentice Hall, 2000, pp. 368–374. [Accessed: Jan. 12, 2025]

[4] MathWorks, "What is MATLAB?," [Online]. Available: https://uk.mathworks.com/discovery/what-is-matlab.html [Accessed: Jan. 13, 2025]

[5] "LINPACK Benchmark," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/LINPACK [Accessed: Feb. 12, 2025]

[6] "Top 500 Supercomputers List – June 1995," Top500.org. [Online]. Available: https://top500.org/lists/top500/list/1995/06/ [Accessed: Feb. 12, 2025]

[7] IBM, "What is high-performance computing (HPC)?," [Online]. Available: https://www.ibm.com/think/topics/hpc [Accessed: Feb. 12, 2025]

[8] MathWorks, "Executing for-loop iterations in parallel on workers," [Online]. Available: https://uk.mathworks.com/help/parallel-computing/parfor.html [Accessed: April. 9, 2025]

[9] S. W. Jones, "Diffusion in Silicon," IC Knowledge LLC. [Online]. Available: https://www-eng.lbl.gov/~shuman/NEXT/MATERIALS&COMPONENTS/Xe_damage/Diffusionin%20siliconpdf.pdf. [Accessed: April 12, 2025]

[10] MathWorks, "Using MATLAB with C and C++," [Online]. Available: https://uk.mathworks.com/products/matlab/matlab-and-c.html [Accessed: April 12, 2025]

[11] MathWorks, "MATLAB Coder," [Online]. Available: https://uk.mathworks.com/products/matlab-coder.html [Accessed: May 12, 2025]

[12] MathWorks, "Graphical User Interface with MATLAB App Designer," [Online]. Available: https://imperix.com/doc/help/gui-with-matlab-app-designer

[13] J. Crank, The Mathematics of Diffusion, 2nd ed. Oxford, U.K.: Clarendon Press, 1975.

[14] S. A. Campbell, "Fabrication Engineering at the Micro- and Nanoscale," Oxford, pp. 43–36. [Online]. Available: https://www.iqytechnicalcollege.com/BAE%20665Fabrication%20Engineering%20at%20the%20Micro%20and%20Nanoscale.pdf

[15] MathWorks, "MATLAB App Designer," [Online]. Available: https://uk.mathworks.com/products/matlab/app-designer.html [Accessed: July. 16, 2025]

[16] S. Selberherr, Analysis and Simulation of Semiconductor Devices, Springer, 1984. [Accessed: May. 15, 2025]

[17] D. Shaw, "Self- and Impurity Diffusion in Ge and Si," *Phys. Stat. Sol. B*, vol. 72, no. 11, pp. 11–39, 1975.

[18] MathWorks, "Graphical User Interface with MATLAB App Designer," [Online]. Available: https://imperix.com/doc/help/gui-with-matlab-app-designer [Accessed: July. 16, 2025]

[19] H. Q. Le, "Bipolar Junction Transistor," ECE 4339, University of Houston. [Lecture notes, unpublished]. [Accessed: Jan. 12, 2025]

[20] G. Lazaridis, *The BJT Transistor Theory*, 2013. [Accessed: Jan. 12, 2025]

[21] S. Dimitrijev, *Principles of Semiconductor Devices*, 2nd ed., Oxford University Press, 2006. (The Oxford series in electrical and computer engineering). ISBN: 978-0-19-538803-9. [Accessed: Jan. 12, 2025]

[22] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, 2nd ed., Society for Industrial and Applied Mathematics, 2004. [Accessed: Jan. 12, 2025]

[23] A. Kloeckner, "Numerical Methods for Partial Differential Equations," CS555 / MATH552 / CSE510, Spring 2022, p. 51. [Accessed: Jan. 12, 2025]

[24]. Jim Plummer, Peter B griffin and Michael D. Deal, Silicon VLSI technology Fundamentals, Practice and Modelling, Prentice Hall Electronics and VLSI series- Charles Sodini, Series editor, 2000, pp.386-389. [Accessed: Dec. 28, 2024]

[25]. Jim Plummer, Peter B griffin and Michael D. Deal, Silicon VLSI technology Fundamentals, Practice and Modelling, PrenPrentice Hallctronics and VLSI series- Charles Sodini, Series editor, 2000, pp.407-412. [Accessed: Dec. 28, 2024]

[26] S. Wolf and R. N. Tauber, Silicon Processing for the VLSI Era, Vol. 3: The Submicron MOSFET. page 19- 20 [Accessed: Aug. 28, 2025]

[27] C. Hu, Bipolar Transistor, Chapter 8, University of California, Berkeley, Lecture Notes, 2001. [Online]. Available: https://www.chu.berkeley.edu/uploads/2020/01/Chemning-Hu_ch8-2.pdf [Accessed: Aug. 28, 2025]

[28] SVCS Process Innovation, "Horizontal Diffusion Furnace," *SVCS Process Innovation*, [Online]. Available: https://svcs.com/products/horizontal-diffusion-furnace/. [Accessed: Aug. 28, 2025].

[29] Dr Kevin Berwick, Nanofabrication 1 Lecture Notes: Question 27, Diffusion', School of electrical and Electronic Engineering, Technological University Dublin.

APPENDICES

Figure 10

```
C=zeros(1,100);
C(1:2)=2e19;
points=100
for runs=1:1:points

 for n=2:1:99
     C(1)=2e19;

 C(n)=0.5*(C(n-1)+C(n+1));

 end
 if mod(runs,25)==1
 plot(C)
 hold on
 title('Constant Source Diffusion');
 xlabel('Depth');
 ylabel('Concentration');
 drawnow;
 end
end
```

Figure 11

```
C=zeros(1,100);
C(1:2)=1e19;
iterations=100;
for runs=1:iterations
 for i=1:1:99
 if i==1
     C(i)=0.5*(C(i+1)+C(i));
 else
 C(i)=0.5*(C(i-1)+C(i+1));
 end
 end
 if mod(runs,25)==1
 plot(C);
 hold on;
 title('Drive in Profile');
 xlabel('Depth(m)');
 ylabel('Concentration (cm^-3)');
 drawnow;
 end
end
```

Figure 12

```
% Parameters
D = 1e-13;
t1 = 600;            % Predeposition  (s)
t2 = 3600;           % Drive-in (s)
Cs = 1e20;           % Surface c (cm^-3)
x = linspace(0, 1e-4, 200);  % depth (cm)

% dopant
Q = Cs * sqrt(pi * D * t1);  % cm^-2


C_pre = Cs * erfc(x / (2 * sqrt(D * t1)));         % Predeposition
C_drive = (Q ./ sqrt(pi * D * t2)) .* exp(-x.^2 / (4 * D * t2));  %

figure;
plot(x*1e4, C_pre, 'b-', 'LineWidth', 2); hold on;
plot(x*1e4, C_drive, 'r--', 'LineWidth', 2);
xlabel('Depth (\mum)');
ylabel('Concentration (cm^{-3})');
title('Predeposition and Drive-in Diffusion Profiles');
legend('Predeposition', 'Drive-in');
grid on;
```

Figure 15

```
    C(i)=0.6*(C(i+1)+C(i));
 else
 C(i)=0.6*(C(i-1)+C(i+1));
 end
 end
 if mod(runs,25)==1
 plot(C);
 hold on;
 title('Drive in Profile');
 xlabel('Depth(m)');
 ylabel('Concentration (cm^-3)');
 drawnow;
 end
end
```

Figure 16

```
  if i==1
      C(i)=0.7*(C(i+1)+C(i));
  else
  C(i)=0.7*(C(i-1)+C(i+1));
  end
  end
  if mod(runs,25)==1
  plot(C);
  hold on;
  title('Drive in Profile');
  xlabel('Depth(m)');
  ylabel('Concentration (cm^-3)');
  drawnow;
  end
 end
```

Figure 18

```
% Depth axis
x = linspace(0,10,400);

%  (Gaussian)
D0 = 2;
t  = 1;
Cs = 1e19;                % surface concentration
Q  = Cs*sqrt(pi*D0*t);       %
C_gauss = (Q./sqrt(pi*D0*t)) .* exp( -x.^2 ./ (4*D0*t) );

% With concentration dependence
xp = 3;                      % where profile starts to fall (controls
w  = 0.2;                    % transition width (smaller -> sharper ec
C_box = Cs ./ (1 + exp((x - xp)/w));   % box-like


figure;
plot(x, C_gauss, 'LineWidth', 1.8); hold on;
plot(x, C_box,   'LineWidth', 1.8);
grid on;
xlabel('Depth um');
ylabel('Concentration (cm^{-3})');
title('Diffusion Profile');
legend('Without concentration dependence (Gaussian)', ...
       'With concentration dependence (box-like)', 'Location','northea
```

Figure 22

```
% Create a 50x50 matrix
matrix = zeros(50, 50);

% Set the middle 16 cells in the LAST 2 rows to 10^11 (top of the matrix)
matrix(end-1:end, 8:43) = 1e19; % Last 2 rows fixed at the top

figure;
h = contourf(matrix);
colormap(jet);
colorbar;
title('Emitter Diffusion ');
axis off;

% Simulate
num_steps = 200; % Number of time steps
for t = 1:num_steps
    % Create a new matrix
    new_matrix = zeros(50, 50);

    % Apply diffusion in both horizontal and vertical directions
    for i = 2:49
        for j = 2:49
            new_matrix(i, j) = 0.25 * (matrix(i-1, j) + matrix(i+1, j) + matrix(i, j-1) + matrix(i, j+1));
        end
    end
```

```
    %  boundary conditions
    new_matrix(end-1:end, 8:43) = 1e19; % Last 2 rows fixed at the top
    new_matrix(:, [1, 50]) = 0; % Edges set to zero

    % Update the matrix
    matrix = new_matrix;

    % Update the visualization
    contourf(matrix);
    colorbar;
    title(sprintf('Emitter Diffusion (50x50) - Step %d', t)); %  step number
    drawnow; % Update the animation
    pause(0.1);
end
```

Figure 24

```
% Create a 50x50 grid
matrix = zeros(50, 50);


matrix(end-1:end, 8:43) = 1e19; % Emitter region


[X, Y] = meshgrid(1:50, 1:50);

% Simulate
num_steps = 200;
for t = 1:num_steps
    % New matrix for updated values
    new_matrix = zeros(50, 50);

    % Apply  diffusion
    for i = 2:49
        for j = 2:49
            new_matrix(i, j) = 0.25 * (matrix(i-1, j) + matrix(i+1, j) + matrix(i, j-1) + matrix(i, j+1));
        end
    end

    %  boundary conditions
    new_matrix(end-1:end, 8:43) = 1e19;
    new_matrix(:, [1, 50]) = 0;

    % Update the matrix
    matrix = new_matrix;
```

```
    % 3D Surface Plot
    surf(X, Y, matrix, 'EdgeColor', 'none');


    title(sprintf('Time=%d   Color: u   Height: u', t));
    xlabel('X'); ylabel('Y'); zlabel('Concentration');
    colorbar;
    colormap(jet);

    clim([0 1e19]); % colorbar scaling
    zlim([0 1e19]);  %  set z-axis limit
    drawnow;
end
```

Figure 26

```
systems = {'HPC Server','Laptop'};
times = [450,1348];

figure;
bar(times);
ylabel('Elapsed time (seconds)');
title ('Elapsed Time to Run the Code in HPC Server and Laptop');
set(gca , 'xticklabel' , systems);
grid on
```

Figure 30

```
% execution times
times = [448.618496, 683.943763, 585.258]; % [MATLAB, C, C++]

% Language
methods = {'MATLAB', 'C', 'C++'};

% bar chart
figure;
bar(times);

% Set x axis
set(gca, 'xticklabel', methods);

title('Comparison of Execution Times');
ylabel('Elapsed Time (seconds)');
xlabel('Programming Language');

grid on;
```

Figure 35

```
% given
Cb = 1.8e16;
Cs = 1.5e20;
tp_hr = 0.5; td_hr = 1;
tp = tp_hr * 3600;
td = td_hr * 3600;

% sqrt(D) from graph at 1100°C
sqrtD = 0.14;  % μm/sqrthr
D = sqrtD^2 * (1/3600) * (1e-4)^2; % cm^2/s

Dp = D;
Dd = D;

%  Pre-deposition
xj_pre = 2 * sqrt(Dp * tp * log(Cs / Cb));
xj_pre_um = xj_pre * 1e4;


Q = 2 * Cs * sqrt(Dp * tp / pi);
```

```
%  Drive-in
log_concentration_ratio = log(Q / (Cb * sqrt(pi * Dd * td)));
xj_drive = sqrt((4 * Dd * td / pi) * log_concentration_ratio);
xj_drive_um = xj_drive * 1e4;

new_input = table(1100, 1.5e20, 1.8e16, ...
    'VariableNames', {'Temperature', 'Cs', 'Cb'});
yfit = trainedModel.predictFcn(new_input);

figure;
bar([xj_drive_um, yfit], 0.4)
set(gca, 'XTickLabel', {'Solved', 'ML-Predicted'})
ylabel('Junction Depth (μm)')
title('Comparison of Junction Depth at 1100°C')
grid on;
```

Figure 36

```
% Inputs
    T_input  = app.tempInput.Value;
    Cs_input = app.csInput.Value;
    Cb_input = app.cbInput.Value;

    % Value control
    if any([T_input, Cs_input, Cb_input] <= 0)
        app.StatusLabel.Text = 'Please enter only positive values.';
        app.StatusLabel.FontColor = [1 0 0];  % Red
        return;
    end

    % Cs < Cb check
    if Cs_input < Cb_input
        app.StatusLabel.Text = 'Surface concentration must be higher than background!';
        app.StatusLabel.FontColor = [1 0 0];  % Red
        return;
    end


    model_xj = loadLearnerForCoder('model_JunctionDepth');

    % Predict
    input_data = [T_input, Cs_input, Cb_input];
    xj_pred = predict(model_xj, input_data);
```