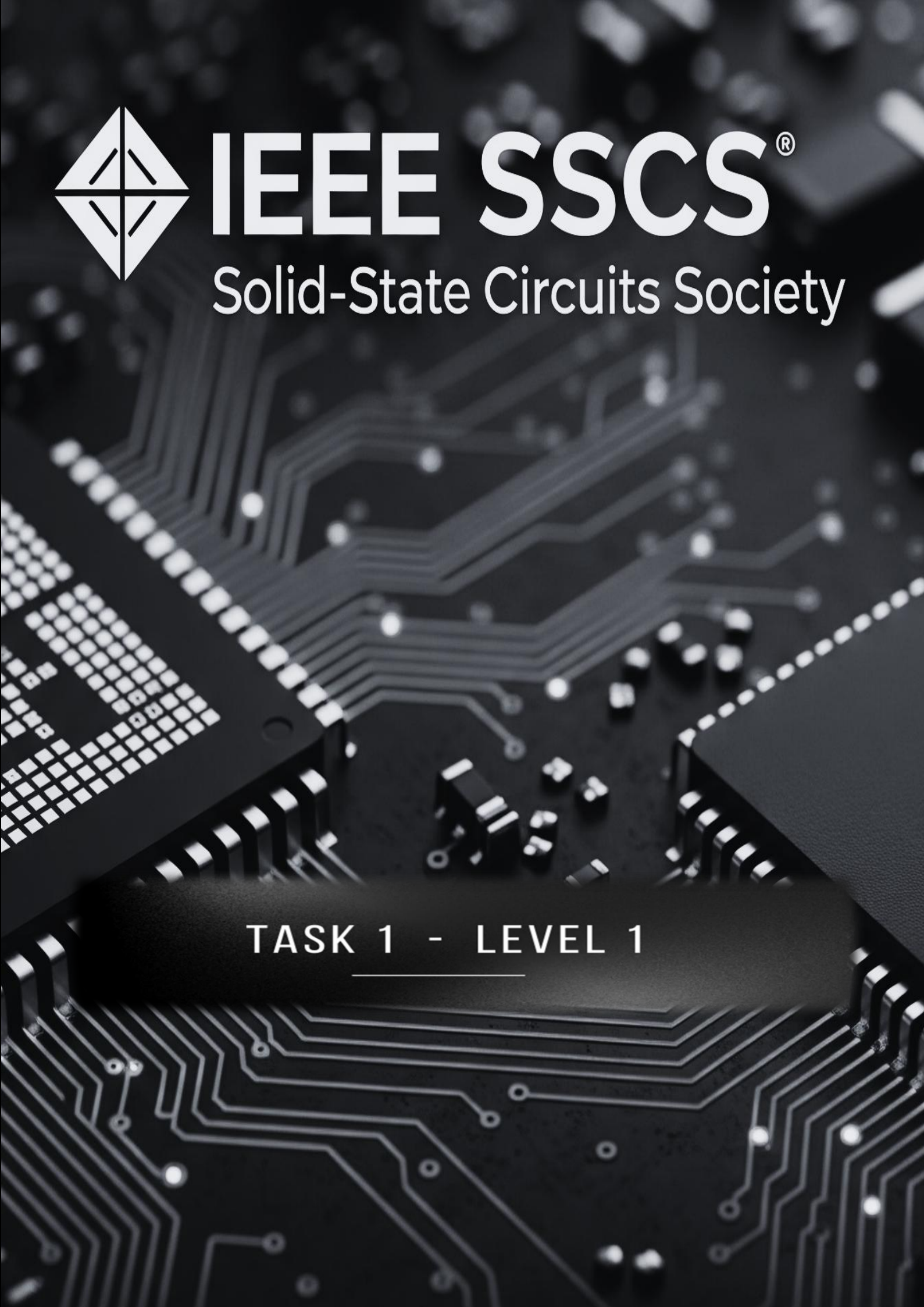
IEEE SSCS®
Solid-State Circuits Society
TASK 1 - LEVEL 1

Level 1 Task

NumPy Assignment

Part 1 – Core NumPy Operations

You are given the following dataset representing students' grades in 4 subjects:

grades =
[[85, 78, 92, 88],
[70, 76, 80, 65],
[90, 88, 94, 91],
[60, 65, 58, 62],
[100, 95, 98, 97]]

Tasks:

- Convert the data into a **NumPy array.**
- **Print** the **shape** of the array.
- **Compute**:
  - The **mean** grade of **each student**.
  - The **mean** grade of **each subject**.
- **Extract** the students whose **average grade** is **greater than 85.**

- Add a bonus of 5 marks to all grades using **broadcasting**.
- **Normalize** the grades using **Min-Max normalization.**
- **Flatten** the array into a **single vector**.

Do not use loops.

Part 2 – Linear Regression from Scratch (Normal Equation)

We want to predict house prices based on house size (in square meters).

X (house size):
[50, 60, 80, 100, 120]

y (price in thousands):
[150, 180, 240, 300, 330]

Tasks:

- **Convert** X and y into **NumPy arrays**.
- **Reshape X** into a **column vector.**
- **Add a bias column** of **ones** to X.
- Compute the parameter vector θ using the Normal Equation:

$$\theta = (X^T X)^{-1} X^T y$$

You must implement the equation step by step using NumPy matrix operations only.

Do not use sklearn or any ML library.

- **Use the computed θ to predict the price of a 90 m² house.**

Part 3 – Research

Answer the following questions clearly and concisely

Question 1:

If the dataset becomes very large (for example, hundreds of thousands or millions of samples), why would a NumPy-based implementation be significantly faster and more efficient than using pure Python loops?

Question 2:

If some features are highly correlated or duplicated (multicollinearity), what mathematical problem can occur in the Normal Equation?

Why does this happen?

How can it be solved without simply removing features or losing useful information?