SMART MATERIAL IDENTIFIER

(HALL EFFECT & MAGNETORESISTANCE)

CREATED BY-

Anmol Bhatnagar

TABLE OF CONTENTS

INTRODUCTION

1.1 BREIF INTRODUCTION

In the study of solid-state physics and materials science, it is essential to accurately determine the type and nature of materials based on their electrical properties. Two important phenomena that help characterize materials are the Hall Effect and Magnetoresistance.

This project presents a Material Identifier Simulator that uses measurements from the Hall Effect and Magnetoresistance experiments to classify unknown materials as metals, semiconductors, or insulators. It also determines whether the semiconductor is n-type or p-type based on the sign of the Hall coefficient.

The program allows users to:

1. Input experimental data (Hall voltage, magnetic field, current, resistances, and sample thickness).
2. Calculate critical parameters such as Hall coefficient, carrier density, and magnetoresistance ratio.
3. Predict the material type and describe its properties.
4. Confirm the prediction by allowing a second experiment for verification.
5. Learn the underlying theory of the Hall Effect and Magnetoresistance.

1.2 RELEVANCE & IMPORTANCE

Hall Effect and Magnetoresistance are two critical phenomena used to identify the properties of materials at a microscopic level.

They provide insights into:

1) The type of charge carriers (electrons or holes),
2) Carrier concentration,
3) Electrical conductivity,
4) Material classification (metal, semiconductor, or insulator).

This knowledge is crucial for the development of electronic devices, sensors, and materials engineering.

The Material Identifier Simulator helps bridge theoretical knowledge and practical understanding by:

- Allowing students and researchers to simulate experiments
- Enhancing the ability to interpret experimental data
- Training users to predict material behavior without expensive laboratory setups
- Supporting education

In today's rapidly advancing technological world, where materials innovation is key to progress in fields like semiconductors and renewable energy identification of materials is the very basic and the first step to projects in these fields.

FEATURES

- Takes inputs in user-friendly units (mV, Gauss, mA, mm)
- Converts units and performs calculations internally
- Determines:
    - Hall Coefficient
    - Charge Carrier Density
    - Carrier Type (n-type or p-type)
    - Magnetoresistance
- Classifies material as:
    - Metal
    - Semiconductor (n-type / p-type)
    - Insulator
- Displays scientific theory for user reference

FORMULAE USED

The key formulas used in the code:

1. **Hall Coefficient (R_h) Calculation**:

   Hall Coefficient = (Hall Voltage * thickness) / (Magnetic Field * current)

2. **Carrier Density (n) Calculation**:

   Carrier Density = 1.0 / (|Hall Coefficient| * ELEMENTARY_CHARGE)

   Where ELEMENTARY_CHARGE = $1.602*10^{-19}$ Coulombs.

3. **Magnetoresistance (ΔR/R) Calculation**:

   Magnetoresistance = $(R_b - R_0) / R_0$

   Where R_b is resistance with magnetic field and R_0 is resistance without magnetic field.

4. **Unit Conversion Formulas**:

- GAUSS_TO_TESLA = 1.0 / 10000.0
- MV_TO_V = 1.0 / 1000.0
- MA_TO_A = 1.0 / 1000.0
- MM_TO_M = 1.0 / 1000.0

These are used throughout the code to convert between different units of measurement.

5. **Material Classification Thresholds**:
   - **For Hall effect:** Carrier Density < 1e19 (insulator), Carrier Density < 1e26 (semiconductor), otherwise metal
   - **For Magnetoresistance:** |magnetoresistance|< 0.005 (metal), |magnetoresistance| > 0.05(semiconductor)

CODE

```
#include <iostream>
#include <cmath>
#include <string>
#include <limits>
#include <vector>
using namespace std;

// Constants
const double ELEMENTARY_CHARGE = 1.602e-19; // Coulombs
const double GAUSS_TO_TESLA = 1.0 / 10000.0; // Conversion factor
const double MV_TO_V = 1.0 / 1000.0; // Conversion factor
const double MA_TO_A = 1.0 / 1000.0; // Conversion factor
const double MM_TO_M = 1.0 / 1000.0; // Conversion factor

// Class for experimental measurements
class ExperimentalData {
private:
    double hallVoltage;      // in volts
    double magneticField;    // in Tesla
    double current;          // in Amperes
    double resistanceNoField; // in Ohms
    double resistanceWithField; // in Ohms
```

```
    double thickness;        // in meters
    double hallCoefficient;  // in m³/C
    string sampleName;       // name or identifier for the sample

public:
    // Default constructor
    ExperimentalData() {
        hallVoltage = 0.0;
        magneticField = 0.0;
        current = 0.0;
        resistanceNoField = 0.0;
        resistanceWithField = 0.0;
        thickness = 0.0;
        hallCoefficient = 0.0;
        sampleName = "Unknown";
    }

    // Parameterized constructor
    ExperimentalData(double hv, double mf, double i, double r0, double rb, double t, string name) {
        hallVoltage = hv;
        magneticField = mf;
        current = i;
        resistanceNoField = r0;
        resistanceWithField = rb;
```

```
    thickness = t;
    hallCoefficient = 0.0;
    sampleName = name;
}

// Getters
double getHallVoltage() const { return hallVoltage; }
double getMagneticField() const { return magneticField; }
double getCurrent() const { return current; }
double getResistanceNoField() const { return resistanceNoField; }
double getResistanceWithField() const { return resistanceWithField; }
double getThickness() const { return thickness; }
double getHallCoefficient() const { return hallCoefficient; }
string getSampleName() const { return sampleName; }

// Setters
void setHallVoltage(double hv) { hallVoltage = hv; }
void setMagneticField(double mf) { magneticField = mf; }
void setCurrent(double i) { current = i; }
void setResistanceNoField(double r0) { resistanceNoField = r0; }
void setResistanceWithField(double rb) { resistanceWithField = rb; }
void setThickness(double t) { thickness = t; }
void setHallCoefficient(double hc) { hallCoefficient = hc; }
void setSampleName(string name) { sampleName = name; }
```

```
// Function to calculate Hall coefficient
void calculateHallCoefficient() {
    if (magneticField != 0.0 && current != 0.0) {
        hallCoefficient = (hallVoltage * thickness) / (magneticField * current);
    } else {
        hallCoefficient = 0.0;
        cout << "Error: Magnetic field or current cannot be zero." << endl;
    }
}

// Function to input Hall effect data from the user
void inputHallData() {
    cout << "\nEnter Sample Name or ID: ";
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    getline(cin, sampleName);

    cout << "Enter Hall Voltage (mV): ";
    cin >> hallVoltage;
    hallVoltage *= MV_TO_V; // Convert to volts

    cout << "Enter Magnetic Field (Gauss): ";
    cin >> magneticField;
    magneticField *= GAUSS_TO_TESLA; // Convert to Tesla
    cout << "Enter Current (mA): ";
```

```
    cin >> current;
    current *= MA_TO_A; // Convert to amperes

    cout << "Enter Sample thickness (mm): ";
    cin >> thickness;
    thickness *= MM_TO_M; // Convert to meters

    // Calculate Hall coefficient automatically
    calculateHallCoefficient();
}

// Function to input magnetoresistance data
void inputMagnetoresistanceData() {
    cout << "\nEnter Resistance without magnetic field (Ohm): ";
    cin >> resistanceNoField;

    cout << "Enter Resistance with magnetic field (Ohm): ";
    cin >> resistanceWithField;
}

// Function to display the Hall effect input data
void displayHallData() const {
    cout << "\n====== HALL EFFECT DATA FOR " << sampleName << " ======" << endl;
    cout << "Hall Voltage: " << (hallVoltage / MV_TO_V) << " mV (" << hallVoltage << " V)" << endl;
```

```
        cout << "Magnetic Field: " << (magneticField / GAUSS_TO_TESLA) << " Gauss (" << magneticField << " T)"
<< endl;
        cout << "Current: " << (current / MA_TO_A) << " mA (" << current << " A)" << endl;
        cout << "Sample Thickness: " << (thickness / MM_TO_M) << " mm (" << thickness << " m)" << endl;
        cout << "Hall Coefficient: " << hallCoefficient << " m³/C" << endl;
    }

    // Function to display the magnetoresistance input data
    void displayMagnetoresistanceData() const {
        cout << "\n====== MAGNETORESISTANCE DATA FOR " << sampleName << " ======" << endl;
        cout << "Resistance without field: " << resistanceNoField << " Ohm" << endl;
        cout << "Resistance with field: " << resistanceWithField << " Ohm" << endl;
    }
};

// Class for multiple observations
class MultipleObservations {
private:
    vector<ExperimentalData> observations;
    string sampleName;
    double thickness;
    double avgHallCoefficient;
    double avgHallVoltage;
    double avgMagneticField;
```

```
    double avgCurrent;
    double avgResistanceNoField;
    double avgResistanceWithField;

public:
    // Constructor
    MultipleObservations() {
        sampleName = "Unknown";
        thickness = 0.0;
        avgHallCoefficient = 0.0;
        avgHallVoltage = 0.0;
        avgMagneticField = 0.0;
        avgCurrent = 0.0;
        avgResistanceNoField = 0.0;
        avgResistanceWithField = 0.0;
    }

    // Function to collect multiple Hall effect observations
    void collectHallObservations() {
        observations.clear();

        cout << "\nEnter Sample Name or ID: ";
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        getline(cin, sampleName);
```

```
cout << "Enter Sample thickness (mm): ";
cin >> thickness;
thickness *= MM_TO_M; // Convert to meters

int numObservations;
cout << "How many observations would you like to make? ";
cin >> numObservations;

for (int i = 0; i < numObservations; i++) {
    cout << "\n--- Observation " << (i+1) << " of " << numObservations << " ---";

    ExperimentalData observation;

    // Set sample name and thickness for all observations
    observation.setSampleName(sampleName);
    observation.setThickness(thickness);

    // Get Hall voltage, magnetic field and current
    cout << "\nEnter Hall Voltage (mV): ";
    double voltage;
    cin >> voltage;
    observation.setHallVoltage(voltage * MV_TO_V);

    cout << "Enter Magnetic Field (Gauss): ";
```

```
        double field;
        cin >> field;
        observation.setMagneticField(field * GAUSS_TO_TESLA);

        cout << "Enter Current (mA): ";
        double current;
        cin >> current;
        observation.setCurrent(current * MA_TO_A);

        // Calculate Hall coefficient for this observation
        observation.calculateHallCoefficient();

        observations.push_back(observation);

        // Display the Hall coefficient for this observation
        cout << "Hall Coefficient for this observation: " << observation.getHallCoefficient() << " m³/C" << endl;
    }

    // Calculate averages
    calculateAverages();

    // Display the observation table
    displayObservationTable();
}
```

```
// Function to collect multiple magnetoresistance observations
void collectMagnetoresistanceObservations() {
    int numObservations;
    cout << "\nHow many magnetoresistance observations would you like to make? ";
    cin >> numObservations;

    double sumR0 = 0.0;
    double sumRB = 0.0;

    cout << "\n--- Magnetoresistance Observations ---" << endl;
    cout << "No.  R without field (Ω)  R with field (Ω)" << endl;

    for (int i = 0; i < numObservations; i++) {
        double r0, rb;
        cout << "\nObservation " << (i+1) << ":" << endl;
        cout << "Enter Resistance without magnetic field (Ohm): ";
        cin >> r0;

        cout << "Enter Resistance with magnetic field (Ohm): ";
        cin >> rb;

        cout << (i+1) << "    " << r0 << "               " << rb << endl;
        sumR0 += r0;
        sumRB += rb;
```

```
    }

    // Calculate average magnetoresistance values
    avgResistanceNoField = sumR0 / numObservations;
    avgResistanceWithField = sumRB / numObservations;

    // Display averages
    cout << "\n====== AVERAGE MAGNETORESISTANCE VALUES ======" << endl;
    cout << "Average Resistance without field: " << avgResistanceNoField << " Ohm" << endl;
    cout << "Average Resistance with field: " << avgResistanceWithField << " Ohm" << endl;
}
// Calculate averages from all observations
void calculateAverages() {
    if (observations.empty()) return;

    double sumHV = 0.0;
    double sumMF = 0.0;
    double sumI = 0.0;
    double sumHC = 0.0;  // Sum of Hall coefficients

    for (const auto& obs : observations) {
        sumHV += obs.getHallVoltage();
        sumMF += obs.getMagneticField();
        sumI += obs.getCurrent();
```

```
        sumHC += obs.getHallCoefficient();
    }
    avgHallVoltage = sumHV / observations.size();
    avgMagneticField = sumMF / observations.size();
    avgCurrent = sumI / observations.size();
    avgHallCoefficient = sumHC / observations.size();

    cout << "\n====== AVERAGE HALL COEFFICIENT ======" << endl;
    cout << "Average Hall Coefficient: " << avgHallCoefficient << " m³/C" << endl;
}

// Display observation table
void displayObservationTable() const {
    cout << "\n====== HALL EFFECT OBSERVATIONS TABLE ======" << endl;
    cout << "Sample: " << sampleName << endl;
    cout << "Thickness: " << (thickness / MM_TO_M) << " mm" << endl;
    cout << "\nNo.  Hall V (mV)  Mag Field (Gauss)  Current (mA)  Hall Coef (m³/C)" << endl;
    cout << "-----------------------------------------------------------------" << endl;

    for (size_t i = 0; i < observations.size(); i++) {
        cout << (i+1) << "    "
            << (observations[i].getHallVoltage() / MV_TO_V) << "        "
            << (observations[i].getMagneticField() / GAUSS_TO_TESLA) << "            "
            << (observations[i].getCurrent() / MA_TO_A) << "        "
```

```
            << observations[i].getHallCoefficient() << endl;
    }
    cout << "-----------------------------------------------------------------" << endl;
    cout << "AVG  "
        << (avgHallVoltage / MV_TO_V) << "        "
        << (avgMagneticField / GAUSS_TO_TESLA) << "            "
        << (avgCurrent / MA_TO_A) << "        "
        << avgHallCoefficient << endl;
}

// Create an ExperimentalData object with average values
ExperimentalData getAverageData() const {
    ExperimentalData avgData(
        avgHallVoltage,
        avgMagneticField,
        avgCurrent,
        avgResistanceNoField,
        avgResistanceWithField,
        thickness,
        sampleName + " (Average)"
    );
    // Set the average Hall coefficient directly
    avgData.setHallCoefficient(avgHallCoefficient);
    return avgData;
```

```
    }
    // Get average hall coefficient
    double getAvgHallCoefficient() const {
        return avgHallCoefficient;
    }
    // Get average resistance values
    double getAvgResistanceNoField() const {
        return avgResistanceNoField;
    }
    double getAvgResistanceWithField() const {
        return avgResistanceWithField;
    }
};
// Class for calculating material properties
class Material Analyzer {
private:
    double hallCoefficient;   // in m³/C
    double Carrier Density;    // in m⁻³
    double magnetoresistance; // dimensionless ratio
    bool isValidHallData;     // flag for Hall effect data validity
    bool isValidMRData;       // flag for magnetoresistance data validity
    char carrierType;         // 'n' for negative, 'p' for positive

public:
```

```
// Default constructor
Material Analyzer() {
    hallCoefficient = 0.0;
    Carrier Density = 0.0;
    magnetoresistance = 0.0;
    isValidHallData = false;
    isValidMRData = false;
    carrierType = '?';
}
// Function to analyze the Hall effect data using provided Hall coefficient
void analyzeHallData(double avgHallCoefficient) {
    // Use the provided average Hall coefficient
    hallCoefficient = avgHallCoefficient;
    if (hallCoefficient == 0.0) {
        isValidHallData = false;
        cout << "Error: Hall coefficient cannot be zero." << endl;
        return;
    }
    // Determine carrier type based on the sign of Hall coefficient
    if (hallCoefficient < 0) {
        carrierType = 'n'; // negative carriers (electrons)
    } else {
        carrierType = 'p'; // positive carriers (holes)
    }
```

```
        // Calculate carrier density
        Carrier Density = 1.0 / (fabs(hallCoefficient) * ELEMENTARY_CHARGE);
        isValidHallData = true;
    }
    // Function to analyze magnetoresistance data
    void analyzeMagnetoresistanceData(double r0, double rb) {
        // Check for division by zero
        if (r0 == 0.0) {
            isValidMRData = false;
            cout << "Error: Resistance without field cannot be zero." << endl;
            return;
}       // Calculate magnetoresistance (ΔR/R)
        magnetoresistance = (rb - r0) / r0;
        isValidMRData = true;
    }
    // Function to display the Hall effect results
    void displayHallResults() const {
        if (!isValidHallData) {
            cout << "No valid Hall effect analysis available." << endl;
            return;
        }
        cout << "\n====== HALL EFFECT ANALYSIS RESULTS ======" << endl;
        cout << "Hall Coefficient: " << hallCoefficient << " m^3/C" << endl;
        cout << "Carrier Type: " << (carrierType == 'n' ? "n-type (electrons)" : "p-type (holes)") << endl;
```

```
    cout << "Charge Carrier Density: " << Carrier Density << " carriers/m^3" << endl;
}
// Function to display the magnetoresistance results
void displayMagnetoresistanceResults() const {
    if (!isValidMRData) {
        cout << "No valid magnetoresistance analysis available." << endl;
        return;
    }

    cout << "\n====== MAGNETORESISTANCE ANALYSIS RESULTS ======" << endl;
    cout << "Magnetoresistance (ΔR/R): " << magnetoresistance << endl;
}
// Function to identify material type based on Hall effect
string identifyMaterialFromHall() const {
    if (!isValidHallData) {
        return "Analysis not available";
    }
    if (Carrier Density < 1e19) {
        return "INSULATOR";
    }
    else if (Carrier Density < 1e26) {
        return carrierType == 'n' ? "n-type SEMICONDUCTOR" : "p-type SEMICONDUCTOR";
    }
    else {
```

```
        return "METAL";
    }
}
// Function to identify material type based on magnetoresistance
string identifyMaterialFromMR() const {
    if (!isValidMRData) {
        return "Analysis not available";
    }
    if (fabs(magnetoresistance) < 0.005) {
        return "METAL";
    }
    else if (fabs(magnetoresistance) > 0.05) {
        return "SEMICONDUCTOR";
    }
    else {
        return "HEAVILY DOPED SEMICONDUCTOR OR POOR METAL";
    }
}
// Function to provide comprehensive material identification
string identifyMaterial() const {
    if (!isValidHallData && !isValidMRData) {
        return "Analysis not available";
    }
    string hallResult = isValidHallData ? identifyMaterialFromHall() : "";
```

```
    string mrResult = isValidMRData ? identifyMaterialFromMR() : "";
    if (isValidHallData && isValidMRData) {
        // Compare and combine results
        if (hallResult == "INSULATOR") {
            return "INSULATOR";
        }
        else if (hallResult==("SEMICONDUCTOR") &&
                mrResult==("SEMICONDUCTOR") ) {
            return hallResult; // Return with carrier type info
        }
        else if (hallResult == "METAL" && mrResult == "METAL") {
            return "METAL";
        }
        else {
            return "HEAVILY DOPED SEMICONDUCTOR OR POOR METAL";
        }
    }
    else if (isValidHallData) {
        return hallResult;
    }
    else {
        return mrResult;
    }
}
```

```
// Function to provide detailed material description
void describeMaterial(const string& sampleName) const {
    if (!isValidHallData && !isValidMRData) {
        cout << "No valid analysis available." << endl;
        return;
    }
    string materialType = identifyMaterial();
    cout << "\n====== MATERIAL IDENTIFICATION ======" << endl;
    cout << "Sample: " << sampleName << endl;
    cout << "Material Type: " << materialType << endl;
    if (materialType==("INSULATOR")) {
        cout << "Properties:" << endl;
        cout << "- Very high electrical resistance" << endl;
        cout << "- Minimal Hall effect response" << endl;
        cout << "- Negligible free charge carriers" << endl;
        cout << "Examples: Glass, Rubber, Plastic, Ceramics" << endl;
    }
    else if (materialType==("SEMICONDUCTOR") ) {
        cout << "Properties:" << endl;
        cout << "- Moderate carrier density (" << Carrier Density << " carriers/m³)" << endl;
        if (isValidMRData) {
            cout << "- Significant magnetoresistance effect (" << magnetoresistance << ")" << endl;
        }
        if (isValidHallData) {
```

```
            cout << "- " << (carrierType == 'n' ? "Electrons are majority carriers" : "Holes are majority carriers")
<< endl;
        }
        cout << "Examples: Silicon, Germanium, Gallium Arsenide" << endl;
    }
    else if (materialType==("METAL")) {
        cout << "Properties:" << endl;
        if (isValidHallData) {
            cout << "- High carrier density (" << Carrier Density << " carriers/m³)" << endl;
        }
        if (isValidMRData) {
            cout << "- Low magnetoresistance effect (" << magnetoresistance << ")" << endl;
        }
        cout << "- Excellent electrical conductivity" << endl;
        cout << "Examples: Copper, Aluminum, Gold, Silver" << endl;
    }
    else if (materialType==("HEAVILY DOPED SEMICONDUCTOR OR POOR METAL")) {
        cout << "Properties:" << endl;
        if(Carrier Density==0){
        if (isValidMRData) {
            cout << "- Less magnetoresistance effect (" << magnetoresistance << ")" << endl;
        }}
        else{
            cout << "- Less magnetoresistance effect (" << magnetoresistance << ")" << endl;
```

```
            cout << "- Moderate carrier density (" << Carrier Density << " carriers/m³)" << endl;
        }
        if (isValidHallData) {
            cout << "- " << (carrierType == 'n' ? "Electrons are majority carriers" : "Holes are majority carriers")
<< endl;
        }
        cout << "Result is INCONCLUSIVE as Hall Effect shows result of a semiconductor but
Magnetoresistance shows results of highly dopes semiconductor and poor metal." <<endl;
        cout << "Examples: P-type Silicon (Si), P-type Gallium Arsenide (GaAs), P-type Germanium (Ge)" <<
endl;
      }
      else {
        cout << "The measured values don't clearly match typical material patterns." << endl;
        cout << "Consider repeating the experiment with higher precision." << endl;
      }
   }
};
// Class for the simulator application
class MaterialSimulator {
private:
   Material Analyzer analyzer;
   MultipleObservations multipleObs;
public:
   // Constructor
   MaterialSimulator() {}
```

```
// Function to display the main menu
void displayMenu() const {
    cout << "\n==========================================" << endl;
    cout << "   HALL EFFECT & MAGNETORESISTANCE ANALYZER  " << endl;
    cout << "==========================================" << endl;
    cout << "\nMENU:" << endl;
    cout << "1. Run Simulator" << endl;
    cout << "2. Show Apparatus" << endl;
    cout << "3. Show Theory" << endl;
    cout << "4. Show Procedure" << endl;
    cout << "5. Exit" << endl;
    cout << "\nEnter your choice: ";
}

// Function to display the theory
void displayTheory() const {
    cout << "\n==========================================" << endl;
    cout << "            THEORETICAL BACKGROUND         " << endl;
    cout << "==========================================" << endl;

    // Hall Effect Theory
    cout << "\n====== HALL EFFECT THEORY ======" << endl;
    cout << "When a magnetic field is applied perpendicular to a current-carrying" << endl;
    cout << "conductor, charge carriers experience a force perpendicular to both" << endl;
```

```
cout << "the current and magnetic field. This creates a voltage across the conductor." << endl;
cout << "\nKey equation: Hall Voltage (V_H) = (I*B)/(n*e*d)" << endl;
cout << "Where:" << endl;
cout << "- I = current through the sample" << endl;
cout << "- B = magnetic field perpendicular to the current" << endl;
cout << "- n = carrier density" << endl;
cout << "- e = elementary charge (1.602 × 10^-19 C)" << endl;
cout << "- d = thickness of the sample" << endl;
cout << "\nHall Coefficient (R_H) = (V_H * d) / (I * B)" << endl;
cout << "The sign of the Hall coefficient indicates the type of charge carriers:" << endl;
cout << "- Negative R_H: n-type (electrons are majority carriers)" << endl;
cout << "- Positive R_H: p-type (holes are majority carriers)" << endl;
cout << "\nCarrier density can be calculated as: n = 1/(|R_H| * e)" << endl;

// Magnetoresistance Theory
cout << "\n====== MAGNETORESISTANCE THEORY ======" << endl;
cout << "Magnetoresistance is the property of a material to change its" << endl;
cout << "electrical resistance when an external magnetic field is applied." << endl;
cout << "\nThe magnetoresistivity is defined as:" << endl;
cout << "Magnetoresistance (ΔR/R) = [R(B) - R(0)]/R(0)" << endl;
cout << "Where:" << endl;
cout << "- R(B) = resistance with magnetic field" << endl;
cout << "- R(0) = resistance without magnetic field" << endl;
cout << "\nIn semiconductors, the magnetoresistance typically follows:" << endl;
```

```
    cout << "ΔR/R ∝ (μ·B)²" << endl;
    cout << "Where μ is the carrier mobility." << endl;
    cout << "\nCharacteristics by material type:" << endl;
    cout << "- SEMICONDUCTORS: Significant magnetoresistance due to intermediate mobility" << endl;
    cout << "- METALS: Small magnetoresistance due to high carrier density" << endl;
    cout << "- INSULATORS: Minimal effect due to very few free carriers" << endl;

    cout << "\nPress Enter to continue...";
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cin.get();
}
// Function to display the apparatus information
void displayApparatus() const {
    cout << "\n========================================" << endl;
    cout << "          EXPERIMENTAL APPARATUS       " << endl;
    cout << "========================================" << endl;

    cout << "\n====== HALL EFFECT APPARATUS ======" << endl;
    cout << "1. Sample holder with four probes (van der Pauw configuration)" << endl;
    cout << "2. Electromagnet (capable of 0-10,000 Gauss)" << endl;
    cout << "3. Current source (0-100 mA DC)" << endl;
    cout << "4. Digital voltmeter (μV resolution)" << endl;
    cout << "5. Gauss meter for magnetic field measurement" << endl;
    cout << "6. Micrometer for sample thickness measurement" << endl;
```

```
    cout << "\n====== MAGNETORESISTANCE APPARATUS ======" << endl;
    cout << "1. Four-point probe resistance measurement setup" << endl;
    cout << "2. Electromagnet (same as for Hall effect)" << endl;
    cout << "3. Constant current source" << endl;
    cout << "4. Digital ohmmeter" << endl;
    cout << "5. Temperature control system (optional)" << endl;

    cout << "\nPress Enter to continue...";
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cin.get();
}
// Function to display the experimental procedure
void displayProcedure() const {
    cout << "\n============================================" << endl;
    cout << "          EXPERIMENTAL PROCEDURE          " << endl;
    cout << "============================================" << endl;

    cout << "\n====== HALL EFFECT PROCEDURE ======" << endl;
    cout << "1. Mount the sample in the holder with four contacts" << endl;
    cout << "2. Measure the sample thickness with micrometer" << endl;
    cout << "3. Apply constant current through opposite contacts" << endl;
    cout << "4. Apply magnetic field perpendicular to sample surface" << endl;
    cout << "5. Measure Hall voltage across remaining contacts" << endl;
```

```
    cout << "6. Reverse current and magnetic field to eliminate offset errors" << endl;
    cout << "7. Calculate Hall coefficient and carrier density" << endl;

    cout << "\n====== MAGNETORESISTANCE PROCEDURE ======" << endl;
    cout << "1. Mount the sample in four-point probe configuration" << endl;
    cout << "2. Measure resistance without magnetic field" << endl;
    cout << "3. Apply magnetic field perpendicular to current flow" << endl;
    cout << "4. Measure resistance with magnetic field" << endl;
    cout << "5. Calculate magnetoresistance ratio" << endl;
    cout << "6. Compare with theoretical expectations for different materials" << endl;

    cout << "\nPress Enter to continue...";
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cin.get();
}
// Function to run the simulator
void runSimulator() {
    int choice;
    cout << "\n====== SIMULATION OPTIONS ======" << endl;
    cout << "1. Hall Effect" << endl;
    cout << "2. Magnetoresistance" << endl;
    cout << "3. Combined Hall Effect and Magnetoresistance Analysis" << endl;
    cout << "4. Return to Main Menu" << endl;
    cin >> choice;
```

```
switch (choice) {
    case 1: {
        // Multiple Hall Effect measurements
        multipleObs.collectHallObservations();
        // Analyze using average values
        analyzer.analyzeHallData(multipleObs.getAvgHallCoefficient());
        analyzer.displayHallResults();
        analyzer.describeMaterial("Sample (Average of Multiple Measurements)");
        break;
    }
    case 2: {
        // Magnetoresistance measurement
        multipleObs.collectMagnetoresistanceObservations();

        // Analyze using average values
        analyzer.analyzeMagnetoresistanceData(
            multipleObs.getAvgResistanceNoField(),
            multipleObs.getAvgResistanceWithField()
        );
        analyzer.displayMagnetoresistanceResults();
        analyzer.describeMaterial("Sample (Magnetoresistance Only)");
        break;
    }
    case 3: {
```

```
            // Combined analysis
            cout << "\n====== COMBINED HALL EFFECT AND MAGNETORESISTANCE ANALYSIS ======" <<
endl;
            cout << "First, let's collect Hall effect data:" << endl;
            multipleObs.collectHallObservations();

            cout << "\nNow, let's collect magnetoresistance data:" << endl;
            multipleObs.collectMagnetoresistanceObservations();

            // Analyze using both data sets
            analyzer.analyzeHallData(multipleObs.getAvgHallCoefficient());
            analyzer.analyzeMagnetoresistanceData(
               multipleObs.getAvgResistanceNoField(),
               multipleObs.getAvgResistanceWithField()
            );
            // Display combined results
            analyzer.displayHallResults();
            analyzer.displayMagnetoresistanceResults();
            analyzer.describeMaterial("Sample (Combined Analysis)");
            break;
         }
         case 4:
         // Return to main menu
         return;
```

```
        default:
            cout << "Invalid choice. Please try again." << endl;
            break;
    }
    cout << "\nPress Enter to continue...";
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cin.get();
}
// Main function to run the simulator application
void run() {
    int choice;
    do {
        displayMenu();
        cin >> choice;
        switch (choice) {
            case 1:
                runSimulator();
                break;
            case 2:
                displayApparatus();
                break;
            case 3:
                displayTheory();
                break;
```

```
            case 4:
                displayProcedure();
                break;
            case 5:
                cout << "\nExiting the application. Goodbye!" << endl;
                break;
            default:
                cout << "Invalid choice. Please try again." << endl;
                break;
        }
    } while (choice != 5);
}
};
// Main function
int main() {
MaterialSimulator simulator;
simulator.run();
return 0;
}
```

OUTPUT

```
==========================================
   HALL EFFECT & MAGNETORESISTANCE ANALYZER
==========================================

MENU:
1. Run Simulator
2. Show Apparatus
3. Show Theory
4. Show Procedure
5. Exit

Enter your choice: 1

====== SIMULATION OPTIONS ======
1. Hall Effect
2. Magnetoresistance
3. Combined Hall Effect and Magnetoresistance Analysis
4. Return to Main Menu
3

====== COMBINED HALL EFFECT AND MAGNETORESISTANCE ANALYSIS ======
First, let's collect Hall effect data:

Enter Sample Name or ID: A1
Enter Sample thickness (mm): 0.5
How many observations would you like to make? 3

--- Observation 1 of 3 ---
Enter Hall Voltage (mV): 16
Enter Magnetic Field (Gauss): 1000
Enter Current (mA): 1.47
Hall Coefficient for this observation: 0.0544218 m╥│/C

--- Observation 2 of 3 ---
Enter Hall Voltage (mV): 22.5
Enter Magnetic Field (Gauss): 1000
Enter Current (mA): 2.01
Hall Coefficient for this observation: 0.0559701 m╥│/C
```

```
--- Observation 3 of 3 ---
Enter Hall Voltage (mV): 29.8
Enter Magnetic Field (Gauss): 1000
Enter Current (mA): 2.50
Hall Coefficient for this observation: 0.0596 m╥│/C

====== AVERAGE HALL COEFFICIENT ======
Average Hall Coefficient: 0.056664 m╥│/C

====== HALL EFFECT OBSERVATIONS TABLE ======
Sample: A1
Thickness: 0.5 mm

No.  Hall V (mV)  Mag Field (Gauss)  Current (mA)  Hall Coef (m╥│/C)
-----------------------------------------------------------------------
1    16           1000               1.47          0.0544218
2    22.5         1000               2.01          0.0559701
3    29.8         1000               2.5           0.0596
-----------------------------------------------------------------------
AVG  22.7667      1000               1.99333       0.056664

Now, let's collect magnetoresistance data:

How many magnetoresistance observations would you like to make? 3

--- Magnetoresistance Observations ---
No.  R without field (╬®)  R with field (╬®)

Observation 1:
Enter Resistance without magnetic field (Ohm): 55.1
Enter Resistance with magnetic field (Ohm): 55.15
1    55.1                 55.15

Observation 2:
Enter Resistance without magnetic field (Ohm): 55.1
Enter Resistance with magnetic field (Ohm): 55.45
2    55.1                 55.45

Observation 3:
Enter Resistance without magnetic field (Ohm): 55.1
```

```
Observation 3:
Enter Resistance without magnetic field (Ohm): 55.1
Enter Resistance with magnetic field (Ohm): 55.8
3    55.1                 55.8

====== AVERAGE MAGNETORESISTANCE VALUES ======
Average Resistance without field: 55.1 Ohm
Average Resistance with field: 55.4667 Ohm

====== HALL EFFECT ANALYSIS RESULTS ======
Hall Coefficient: 0.056664 m^3/C
Carrier Type: p-type (holes)
Charge Carrier Density: 1.10162e+020 carriers/m^3

====== MAGNETORESISTANCE ANALYSIS RESULTS ======
Magnetoresistance (╬öR/R): 0.00665457

====== MATERIAL IDENTIFICATION ======
Sample: Sample (Combined Analysis)
Material Type: HEAVILY DOPED SEMICONDUCTOR OR POOR METAL
Properties:
- Less magnetoresistance effect (0.00665457)
- Moderate carrier density (1.10162e+020 carriers/m╥│)
- Holes are majority carriers
Result is INCONCLUSIVE as Hall Effect shows result of a semiconductor but Magnetoresistance shows results of highly dopes semiconductor and poor metal.
Examples: P-type Silicon (Si), P-type Gallium Arsenide (GaAs), P-type Germanium (Ge)

Press Enter to continue...
```

CONCLUSION

Summary

The Material Identifier Simulator offers the ability to study and identify material of a sample through Hall Effect and Magnetoresistance. It enables the identification of materials based on their behavior under magnetic fields, helping users determine carrier type, concentration, and conductivity. By providing a practical, accessible simulation environment we can easily test the type of a sample with less efforts and calculations.

Future Work

- Add graphical visualization of Hall voltage vs magnetic field.
- Extend support for temperature-dependent studies.
- Add file handling for saving/loading experiment data.

REFERENCES

- https://en.wikipedia.org/wiki/Hall_effect
- https://en.wikipedia.org/wiki/Magnetoresistance
- http://scienceworld.wolfram.com/physics/HallCoefficient.html
- http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/hall.html
- http://www.eeel.nist.gov/812/hall.html