

Модель провідності металів

Поведінку металів під дією зовнішніх полів (електричного, магнітного, теплового) можна досить просто пояснити без залучення квантових властивостей, на основі класичної електронної теорії, створеної Друде і Лоренцем на початку нашого століття.

Основні положення класичної електронної теорії дуже прозорі і легко піддаються алгоритмізації:

1) Метал складається з іонного каркасу і вільних електронів, які не зв'язані з конкретними іонами і можуть розглядатися як ідеальний газ. Іони лише забезпечують нейтральність системи і не дають електронам розлітатись.

2) Рух електронів у металі - це поступальний рух вільних частинок між зіткненнями. Під час зіткнення електрон «втрачає пам'ять», тобто забуває свою передісторію так, що незалежно від його попереднього руху зразу після зіткнення середня векторна швидкість $\bar{v} = 0$, а середній квадрат швидкості визначається умовою термодинамічної рівноваги $\overline{v^2} = \frac{3k_B T}{m}$, де T - температура в місці зіткнення.

3) Імовірність електрону зазнати зіткнення на протязі малого часу dt пропорційна цьому часу і рівна $p(dt) = \frac{dt}{\tau}$, де τ - час релаксації, а формула справедлива, коли $dt \ll \tau$ (для більшості металів $\tau \approx 10^{-14} \text{ c}$).

Зіткнення електронів призводять до появи ефективної сили опору, яка пропорційна середній швидкості дрейфу електронів і направлена проти неї (в'язкий опір). В строгій теорії показується, що ця сила опору рівна $-\frac{m\bar{v}}{\tau}$. Як і будь-який в'язкий опір, ця сила призводить до постійної швидкості дрейфу під дією постійної зовнішньої сили. Якщо ж зовнішня сила змінна, то наявність опору призведе до відставання по фазі середньої швидкості від сили.

Як відомо, рух електронного газу в металі під дією зовнішнього електричного поля, на перший погляд, повертає нас від Галілея і Ньютона до Аристотеля: силі пропорційне не прискорення, а середня швидкість. Це обумовлено зіткненнями електронів з тепловими і структурними дефектами, що і потрібно коректно врахувати в відповідних моделях. Розглянемо один із шляхів перевірки закону Ома в комп'ютерному експерименті.

Найпростіший спосіб – модель одновимірного блукання електронів в електричному полі з випадковими зіткненнями. Основним параметром цієї моделі є середній час між зіткненнями τ . Електрон рухається під дією поля рівноприскорено до зіткнення. Час руху без зіткнень задається генератором випадкових чисел за правилом

$$t = 2\tau \cdot \text{random.}$$

У даному найпростішому варіанті моделі вважається, що в момент зіткнення електрон повністю втрачає швидкість і починає новий цикл прискорення з нуля. На дисплеї будується графік залежності середньої швидкості від часу. Він має осцилюючий характер і через кілька сотень зіткнень виходить на асимптотичне значення, яке фіксується в пам'яті одночасно з значенням поля. Ця процедура повторюється при різних значеннях поля E , а потім будується графік $\bar{v}(E)$. Точки цього графіка наскільки добре лягають на пряму, що навіть не потребують обробки методом найменших квадратів. Густина електричного струму пропорційна середній швидкості руху електронів, яка в свою чергу пропорційна напруженості поля, що і доводить справедливості закону Ома.

Перевірка закону Джоуля-Ленца

В основу моделі покладене використання генератора випадкових чисел для часу між зіткненнями електронів з решіткою. Вважається, що під час зіткнення електрон втрачає всю набуту від електричного поля енергію. Після усереднення по багатьом зіткненням, знаходиться середня кінетична енергія, яка передається електроном решітці за одиницю часу. Після множення на концентрацію електронів ми отримуємо потужність теплових втрат в одиниці об'єму.

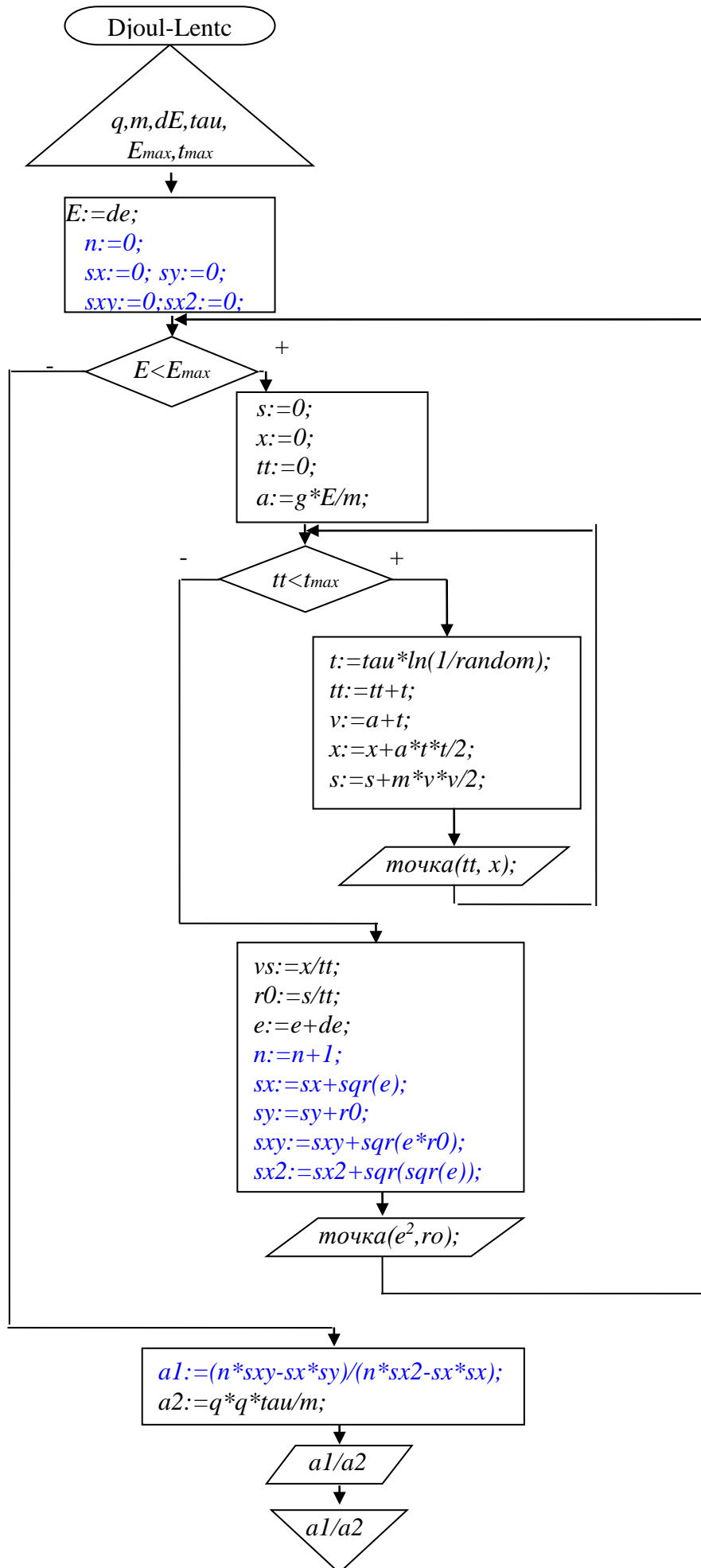
Алгоритм

1. Встановити параметри: заряд електрона, маса електрона, час вільного пробігу електрона, початкова напруженість електричного поля E .
2. Поки $n < 50$ виконувати:
 - а) визначити початкові значення (обнулити): координати електрона, загальний часу руху електрона та сумарну кінетичну енергію;
 - б) поки $k < 200$ виконувати:
 - i) обрахувати прискорення електрона полем;
 - ii) використовуючи генератор випадкових чисел обрахувати час руху електрона до зіткнення з іоном решітки та загальний час руху електрона;
 - iii) визначити миттєву швидкість електрона перед зіткненням з іоном решітки;
 - iv) перемістити електрон в точку зіткнення;
 - v) обрахувати сумарну кінетичну енергію електрона;
 - в) за пройденим шляхом визначити середню швидкість електрона;
 - г) визначити середню кінетичну енергію електрона;
 - д) змінити напруженість електричного поля.

Тестовий приклад. $E = 0$, $\Delta E = 100$, $\tau = 10^{-14}$ - довжина вільного пробігу.

Завдання. Перевірити квадратичну залежність густини потужності теплових втрат від напруженості поля. В результаті комп'ютерного експерименту впевнитися, що коефіцієнт пропорційності в цій залежності рівний $\frac{n e^2 \tau}{m}$, тобто співпадає з електропровідністю металу.

Коментар: В алгоритмі моделі Джоуля-Ленца для апроксимації (аналітичного опису) отриманої залежності середньої кінетичної енергії від напруженості поля використано [метод найменших квадратів MMS](#) (його фрагменти в алгоритмі виділено синім кольором, а нижче описано ідею методу).



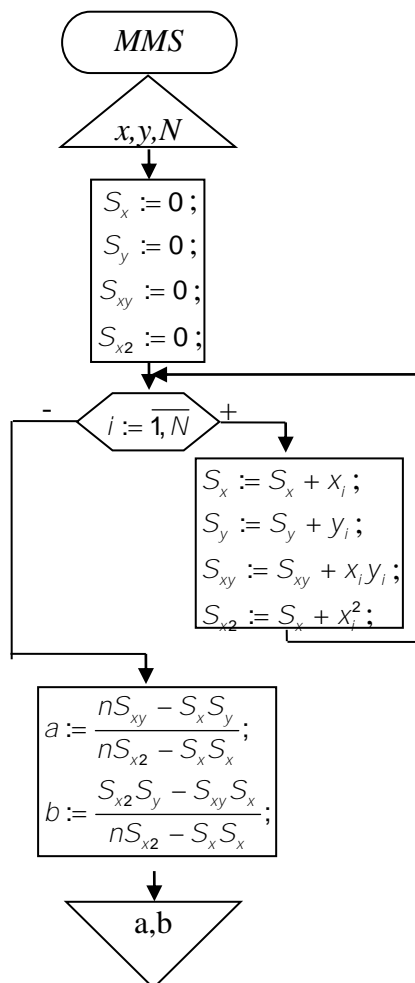
Метод найменших квадратів

Досить часто фізичні величини, отримані у результаті комп'ютерного експерименту, виявляють певні математичні закономірності (лінійну, степеневу, експоненційну, логарифмічну). Існують спеціальні методики для апроксимації (наближення) отриманих числових послідовностей кривими певного виду. Для прикладу розглянемо метод найменших квадратів для апроксимації N експериментальних точок (x_i, y_i) прямою $y = ax + b$. Для визначення коефіцієнтів a і b використаємо умову мінімальності суми квадратів відхилень ординати кожної точки y_i від ординати точки з абсцисою x_i , що належить прямій :

$$\min f(a,b) = \min \sum_i (ax_i + b - y_i)^2 \Rightarrow \begin{cases} \frac{\partial f}{\partial a} = 2 \sum_i x_i (ax_i + b - y_i) = 0 \\ \frac{\partial f}{\partial b} = 2 \sum_i (ax_i + b - y_i) = 0 \end{cases}$$

$$a = \frac{n \sum_i x_i y_i - \sum_i x_i \sum_i y_i}{n \sum_i x_i^2 - \sum_i x_i \sum_i x_i}; b = \frac{\sum_i x_i^2 \sum_i y_i - \sum_i x_i y_i \sum_i x_i}{n \sum_i x_i^2 - \sum_i x_i \sum_i x_i}$$

Отже, для визначення a і b достатньо знайти суми абсцис S_x , ординат S_y , їх добутків S_{xy} та суму квадратів абсцис S_{x2} . При цьому потрібно не забути спочатку суматори обнулити, а потім у циклі пройти по всіх точках. Оскільки відома кількість точок, то можна скористатися циклом *for*. Нижче пропонуємо блок-схему для методу лінійної апроксимації, щоб познайомити читачів з принципом роботи з блок-схемами.



type ar: array [1..100] of real;

procedure MMS(x,y:ar; N:integer;

var a,b:real);

var Sx,Sy,Sxy,Sx2:real;

begin

$S_x := 0;$

$S_y := 0;$

$S_{xy} := 0;$

$S_{x2} := 0;$

for i:=1 to N **do**

begin

$S_x := S_x + x_i;$

$S_y := S_y + y_i;$

$S_{xy} := S_{xy} + x_i y_i;$




$S_{x2} := S_x + x_i^2;$

end;

$a = \frac{n S_{xy} - S_x S_y}{n S_{x2} - S_x S_x};$

$b = \frac{S_{x2} S_y - S_{xy} S_x}{n S_{x2} - S_x S_x};$

end;

1. Підпрограма має назву *MMS* і є процедурою, оскільки автофігура результатів  має суцільну лінію.
2. Процедура має аргументи x, y, N (автофігура ). З контексту умови: x та y множини точок, N – їх кількість. Тому тип аргументів: $x, y: ar; N: integer$; (ar описано через тип, оскільки не можна безпосередньо задати тип масиву при описі аргументів та результатів процедури).
3. Процедура має результати a, b (автофігура ). З контексту умови: a, b дійсні числа. Тому тип результатів: $a, b: real$;
4. У описовому блоці процедури потрібно описати локальні змінні, які будуть використані у самій процедурі, але відсутні у описі заголовку процедури (змінні можуть мати ідентифікатори, що відмінні від використаних у блок-схемі, оскільки не кожна мова програмування дозволяє використовувати підрядкові, нарядкові індекси і т.д.). Увага! Ця частина відсутня у блок-схемі. Але не турбуйтеся, якщо Ви відразу не відстежили всі локальні змінні – їх завжди можна дописати у процесі відлагодження програми
5. Далі формуємо виконуваний блок. У даній схемі досить прості оператори присвоювання. Виняток складає оператор циклу *for*: запис $i := \overline{1, N}$ означає, що змінна i пробігає від 1 до N з кроком одиниця.
6. Після останньої закриваючої операторної дужки *end* ставимо крапку з комою, оскільки описуємо підпрограму.

ДЛЯ ТЕСТУВАННЯ (ПЕРЕВІРКИ КОРЕКТНОЇ РОБОТИ) ПІДПРОГРАМИ *MMS*
 вхідні дані для процедури потрібно певним чином задати. Також треба, щоб програма виводила результат.

Пропонується два способи **введення даних**.

1. Описати типізований масив (тобто такий, для якого одразу задається набір значень) .

Нижче подається приклад опису такого масиву:

var $x: array [1..5] \text{ of } integer = (1, 2, 3, 4, 5)$;

Фрагмент, що є новим порівняно з описом звичайного масиву, підкреслено.

2. Масив x заповнити в циклі значеннями, що збільшуються з постійним кроком, а масив y – значеннями, що близькі до ординат точок прямої $y = a_1 * x + b_1$, але відрізняються від них на невеликі випадково задані значення. Формула для елемента масиву може бути наступною:

$$y_i = a_1 * x_i + b_1 + a_1 * (2 * random - 1) .$$

Суцільною лінією підкреслено випадковий доданок.

Масиви x та y заповнюються в циклі (наприклад, *for*), a_1, b_1 – задані постійні параметри. Для того, щоб випадковий доданок за кожного наступного запуску програми був новим, перед циклом для заповнення масивів x та y потрібно написати randomize ;

Зразок фрагменту коду для введення даних в циклі:

```
const
a1=1;
b1=2;
.....
randomize;
for i:=1 to N do
begin
    x[i]:=i;
    y[i]:=a1*i+b1+a1*(2*random-1);
end;
```

Виведення даних може бути здійснене так. На елементі *chart* вивести два графіка (*series*): перший – точки початкового набору (x_i, y_i) , другий – пряма $y = ax + b$. Значення a, b можна також вивести за допомогою *Label'a*, щоб їх можна було порівняти з a_1, b_1 (для другого способу введення початкових даних) або зі значеннями тестового прикладу (для першого способу введення).

Тестовий приклад:

```
N=10;
x=(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
y=(1.1, 1.9, 2.8, 4.3, 5, 5.9, 7.1, 8.2, 8.8, 10);
a=0.99818;
b=0.02.
```