Аннотация

Работа посвящена разработке численного решателя задачи поиска ёмкости эквивалентного конденсатора для участка электрической цепи. Представлено определение закона сохранения заряда, который является одной из базовых закономерностей в области расчёта электрических цепей. Вместе с тем разобран пример действия закона сохранения заряда для конденсаторов, соединенных между собой. Если для последовательного и параллельного соединения конденсаторов выведены формулы для расчёта ёмкости эквивалентного конденсатора, то для участка цепи произвольной конфигурации вывести общие формулы затруднительно. Так как применение закона сохранения заряда для данной задачи приводит к формированию системы линейных уравнений, в работе приведены базовые определения и понятия, связанные со СЛАУ. Также разобран метод Гаусса как один из наиболее известных численных методов решения СЛАУ. С учетом всего вышесказанного сформирован алгоритм численного решателя задачи поиска ёмкости эквивалентного конденсатора, а также предложен удобный формат файла с входными данными. Программное решение было успешно апробировано на примерах цепей различной конфигурации и может быть использовано учениками в ходе изучения разделов курса физики, связанных с конденсаторами, в том числе для самопроверки.

Содержание

Введение

Цель проекта: разработать алгоритм нахождения ёмкости эквивалентного конденсатора для участка электрической цепи, состоящего из конденсаторов, и реализовать его в виде программы.

Задачи проекта:

- изучить методы расчета ёмкости эквивалентного конденсатора для участка цепи в статическом режиме;
- изучить методы решения систем линейных алгебраических уравнений;
- разработать алгоритм решения поставленной задачи;
- предложить структуру файла входных данных;
- разработать программную реализацию алгоритма, указанного выше.

Закон сохранения заряда

Закон сохранения электрического заряда гласит, что алгебраическая сумма электрических зарядов всех частиц изолированной системы не меняется при происходящих в ней процессах. Электрический заряд любой частицы или системы частиц является целым кратным *элементарному электрическому заряду* (равному по величине заряду электрона) или нулевым.

Одним из подтверждений закона сохранения электрического заряда служит строгое равенство (по абсолютной величине) электрических зарядов электрона и протона. Изучение движения атомов (молекул) и микроскопических тел в электрических полях подтверждает электронейтральность вещества и, соответственно, равенство зарядов электрона и протона (и электронейтральность нейтрона) с точностью до 10^{-21}.

Полный электрический заряд сохраняется и в том случае, если первоначальные заряды тел были отличны от нуля. Если обозначить

первоначальные заряды тел как q_1 и q_2, а заряд тех же тел после их взаимодействия как q'_1 и q'_2. то можно записать:

$$q'_1 + q'_2 = q_1 + q_2.$$

При любых взаимодействиях тел их полный электрический заряд остается неизменным. В этом заключается фундаментальный закон природы — закон сохранения электрического заряда.

Закон сохранения заряда был установлен в 1750 г. американским ученым и видным политическим деятелем Бенджамином Франклином. Он же ввел понятие о положительных и отрицательных зарядах, обозначив их знаками «+» и «-». Закон сохранения заряда имеет глубокий смысл. Он очевиден, когда число элементарных частиц не меняется. Однако элементарные частицы могут возникать (рождаться) и исчезать, т. е. претерпевать различные превращения. Возникают и исчезают элементарные частицы всегда парами (с противоположными зарядами). Многочисленные наблюдения превращений элементарных частиц подтверждают закон сохранения заряда. Этот закон выражает одно из фундаментальных свойств электрического заряда.

Посмотрим, как может быть применен закон сохранения заряда на участке цепи постоянного тока, состоящего из конденсаторов. Конденсатор представляет собой два проводника (называемых *обкладками* конденсатора), разделённые слоем диэлектрика, толщина которого мала по сравнению с размерами проводников. Заряды на обкладках конденсатора равны по модулю и противоположны по знаку, а общий заряд конденсатора равен абсолютному значению заряда одной из обкладок. Если мы соединим несколько заряженных конденсаторов последовательно, то общий заряд системы перераспределится между обкладками конденсаторов с учетом знаков зарядов. Из этого можно сделать важный вывод: заряд между соседними обкладками 2 и более конденсаторов равен нулю. Если мы соединим несколько конденсаторов, то в соединяющем узле это правило будет выполняться.

Рассмотрим соединение конденсаторов, представленное на рисунке 1.

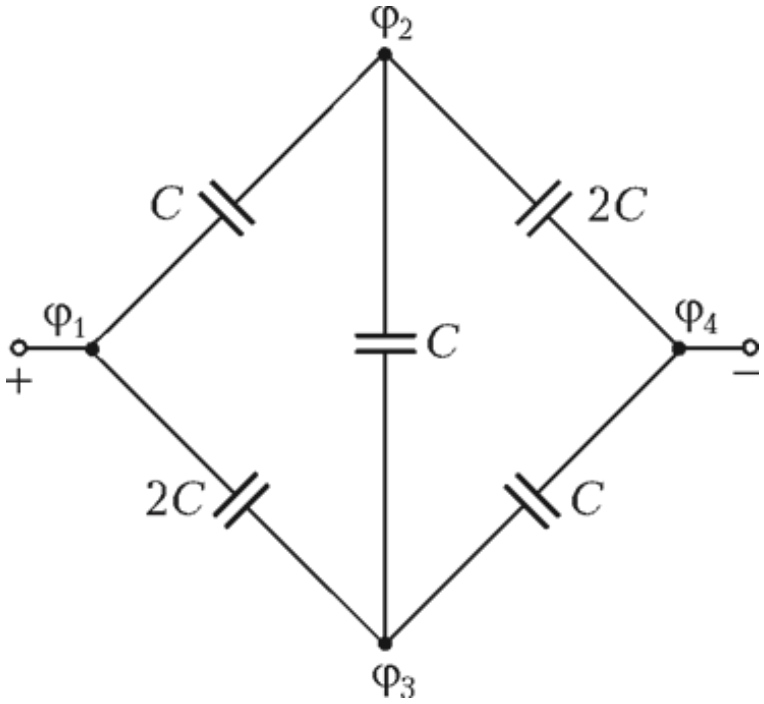

Рис. 1. Пример участка электрической цепи, состоящего из конденсаторов

Суммарную электроемкость можно вычислить по формуле:

$$C_{\text{общ}} = \frac{q_{\text{общ}}}{U},$$

где $q_{\text{общ}}$ – полный заряд и $U = \varphi_1 - \varphi_2$ – напряжение между клеммами.

Проанализируем перераспределение заряда между конденсаторами и найдем области, где заряд между пластинами соседних конденсаторов равен нулю. На рисунке 2 области нулевого заряда между соседними конденсаторами обозначены голубым цветом.

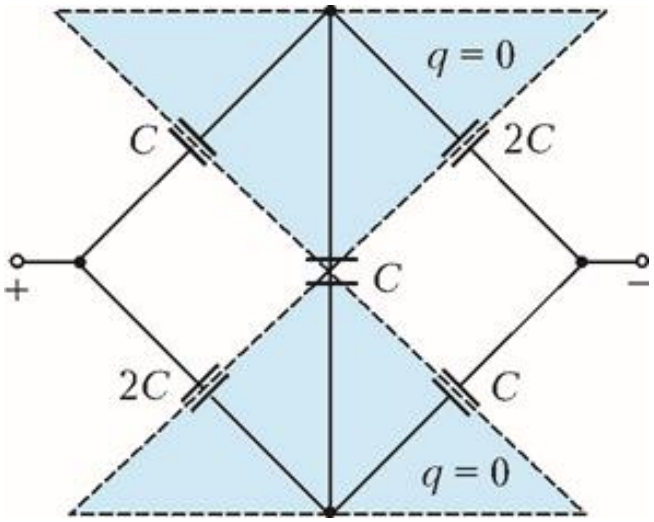

Рис. 2. Разделение цепи на области нулевого заряда

Полный заряд этой системы равняется полному заряду только двух конденсаторов, емкостями C и 2C, которые являются самыми близкими к положительной клемме. Заряд прибывает на пластины только этих конденсаторов, а на остальных он перераспределяется, и никакие новые заряды не создаются в системе.

Чтобы продолжить решение, мы в произвольном порядке приписываем заряды всем конденсаторам, как показано на рисунке 3, и получаем следующую систему уравнений:

$$1.\ -q_1 - q_3 + q_5 = 0$$
$$2.\ q_2 + q_4 - q_5 = 0$$
$$3.\ \varphi_1 - \varphi_2 = \frac{q_1}{C}$$
$$4.\ \varphi_3 - \varphi_1 = \frac{q_2}{2C}$$
$$5.\ \varphi_2 - \varphi_3 = \frac{q_5}{C}$$
$$6.\ \varphi_4 - \varphi_2 = \frac{q_3}{2C}$$
$$7.\ \varphi_3 - \varphi_4 = \frac{q_4}{C}$$

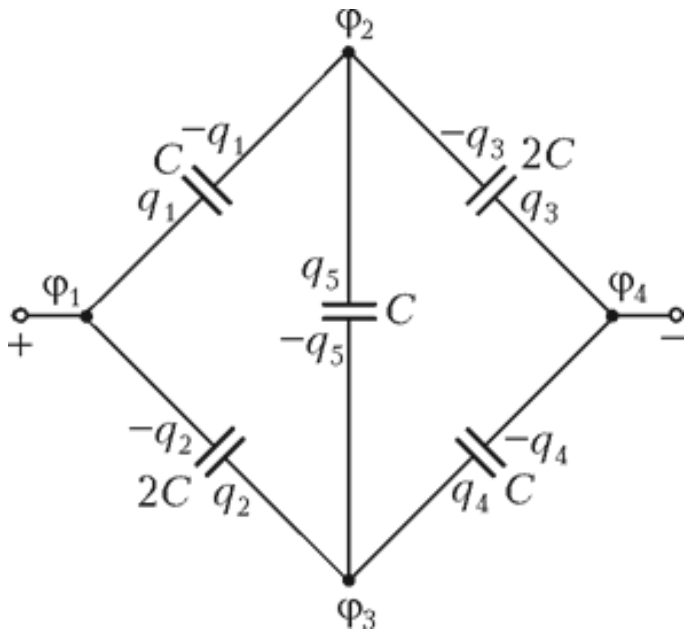

Рис. 3. Расставление зарядов на конденсаторах

Теперь, зная, что $\varphi_1 - \varphi_4 = U$, необходимо решить данную систему и найти q_1 и q_2. Складываем уравнения 3, 4, 5 и получаем:

$$0 = \frac{q_2}{2C} + \frac{q_1}{C} + \frac{q_5}{C}, \text{или } q_2 + 2q_1 + 2q_5 = 0.$$

Затем складываем уравнения 5, 6, 7 и получаем:

$$0 = \frac{q_4}{C} + \frac{q_3}{2C} + \frac{q_5}{C}, \text{ или } 2q_5 + q_3 + 2q_4 = 0.$$

Из уравнений 3 и 6, учитывая, что разность потенциалов φ_1 и φ_4 равна U, находим

$$U = \frac{q_1}{C} - \frac{q_3}{2C}.$$

Теперь объединим получившиеся три уравнения с первыми двумя уравнения нашей исходной системы, выразим из них заряды q_3, q_4 и q_5 через заряды q_1 и q_2 и получим:

$$q_1 = \frac{3CU}{5}, \; q_2 = -\frac{4CU}{5}$$

Подставив данные выражения в формулу для общей емкости, найдем:

$$C_{\text{общ}} = \frac{|q_1| + |q_2|}{U} = \frac{7C}{5}.$$

Данный метод нахождения общей емкости системы конденсаторов универсален. У него есть дополнительное преимущество: нам не нужно знать истинные знаки зарядов заранее; если мы предположили знак неправильно, решение просто даст нам отрицательный ответ.

В ходе решения указанной выше задачи мы формируем математическую модель, которая представляет собой систему линейных алгебраических уравнений.

Системы линейных алгебраических уравнений

Система линейных алгебраических уравнений (СЛАУ) — система уравнений, каждое уравнение в которой является линейным — алгебраическим уравнением первой степени. В классическом варианте коэффициенты при переменных, свободные члены и неизвестные считаются вещественными числами, но все методы и результаты сохраняются (либо естественным образом обобщаются) на случай любых полей, например, комплексных чисел.

Общий вид системы линейных алгебраических уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

Здесь m — количество уравнений, а n — количество переменных, $x_1, x_2, \ldots, x_n$ — неизвестные, которые надо определить, коэффициенты $a_{11}, a_{12}, \ldots, a_{mn}$ и свободные члены $b_1, b_2, \ldots, b_m$ предполагаются известными. Индексы коэффициентов в системах линейных уравнений (a_{ij}) формируются по

следующему соглашению: первый индекс (i) обозначает номер уравнения, второй (j) — номер переменной, при которой стоит этот коэффициент.

Система называется *однородной*, если все её свободные члены равны нулю ($b_1=b_2=...b_m=0$), иначе — *неоднородной*.

Квадратная система линейных уравнений — система, у которой количество уравнений совпадает с числом неизвестных ($m=n$). Система, у которой число неизвестных больше числа уравнений является недоопределённой, такие системы линейных алгебраических уравнений также называются *прямоугольными*. Если уравнений больше, чем неизвестных, то система является переопределённой.

Решение системы линейных алгебраических уравнений — совокупность n чисел $c_1, c_2, \ldots, c_n$, таких что их соответствующая подстановка вместо $x_1, x_2, \ldots, x_n$ в систему обращает все её уравнения в тождества.

Система называется совместной, если она имеет хотя бы одно решение, и несовместной, если у неё нет ни одного решения. Решения считаются различными, если хотя бы одно из значений переменных не совпадает.

В нашем случае система уравнений составляется таким образом, чтобы она получилась определенной (имела единственное решение).

Матричная форма

Система линейных алгебраических уравнений может быть представлена в матричной форме:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Сокращённая запись матричной формы СЛАУ выглядит так: $Ax=b$. Здесь A — это матрица системы, x — столбец неизвестных, а b — столбец свободных членов. Если к матрице A приписать справа столбец свободных членов, то получившаяся матрица называется расширенной.

Методы решения СЛАУ. Метод Гаусса

Численные методы решения СЛАУ делятся на две группы:

- прямые (точные) методы;
- итерационные (приближенные) методы.

К прямым методам относятся такие методы, которые в предположении, что вычисления ведутся без округлений, позволяют получить точные значения неизвестных. Они просты, универсальны и используются для широкого класса систем. Однако они не применимы к системам больших порядков ($n < 200$) и к плохо обусловленным системам из-за возникновения больших погрешностей. К ним можно отнести: правило Крамера, методы обратных матриц, Гаусса, прогонки, квадратного корня и др.

К приближенным относятся методы, которые даже в предположении, что вычисления ведутся без округлений, позволяют получить решение системы лишь с заданной точностью. Это итерационные методы, т. е. методы последовательных приближений. К ним относятся метод простой итерации, метод Зейделя и другие.

Метод Гаусса (или метод последовательного исключения неизвестных) применим для решения систем линейных уравнений, в которых число неизвестных может быть либо равно числу уравнений, либо отлично от него. Система m линейных уравнений с n неизвестными имеет следующий вид:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_n \end{cases},$$

где x_1, x_2, …, x_n – неизвестные, a_{ij} - коэффициенты при неизвестных, b_i - свободные члены (или правые части).

К элементарным преобразованиям системы отнесем следующее:

- перемена местами двух любых уравнений;
- умножение обеих частей любого из уравнений на произвольное число, отличное от нуля;
- прибавление к обеим частям одного из уравнений системы соответствующих частей другого уравнения, умноженных на любое действительное число.

Элементарные преобразования переводят систему уравнений в равносильную ей. Эти преобразования в сущности являются основой метода Гаусса.

Рассмотрим применение метода Гаусса для СЛАУ с тремя неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \end{cases} \quad (1)$$

На первом шаге исключим неизвестное x_1 из всех уравнений системы (1), кроме первого. Пусть коэффициент $a_{11} \neq 0$. Назовем его ведущим элементом. Разделим первое уравнение системы на a_{11}. Получим уравнение:

$$x_1 + a_{12}^{(1)}x_2 + a_{13}^{(1)}x_3 = b_1^{(1)} \quad (2)$$

где $a_{1j}^{(1)} = \frac{a_{1j}}{a_{11}}, j = 1,2,3, b_1^{(1)} = \frac{b_1}{a_{11}}$.

Исключим x_1 из второго и третьего уравнений системы (1). Для этого вычтем из них уравнение (2), умноженное на коэффициент при x_1 (соответственно a_{21} и a_{31}). Система примет следующий вид:

$$\begin{cases} x_1 + a_{12}^{(1)} x_2 + a_{13}^{(1)} x_3 = b_1^{(1)} \\ a_{22}^{(1)} x_2 + a_{23}^{(1)} x_3 = b_2^{(1)} \\ a_{32}^{(1)} x_2 + a_{33}^{(1)} x_3 = b_3^{(1)} \end{cases}. \quad (3)$$

На втором шаге исключим неизвестное x_2 из третьего уравнения системы (3). Пусть коэффициент $a_{22}^{(1)} \neq 0$. Выберем его за ведущий элемент и разделим на него второе уравнение системы (3), получим уравнение:

$$x_2 + a_{23}^{(2)} x_3 = b_2^{(2)}, \quad (4)$$

где $a_{23}^{(2)} = \frac{a_{23}^{(1)}}{a_{22}^{(1)}}, b_2^{(2)} = \frac{b_2^{(1)}}{a_{22}^{(1)}}$.

Из третьего уравнения системы (3) вычтем уравнение (4), умноженное на $a_{33}^{(1)}$. Получим уравнение:

$$a_{33}^{(2)} \bullet x_3 = b_3^{(2)}$$

Предполагая, что $a_{33}^{(2)} \neq 0$, находим x_3:

$$x_3 = \frac{b_3^{(2)}}{a_{33}^{(2)}} = b_3^{3}$$

В результате преобразований система приняла вид:

$$\begin{cases} x_1 + a_{12}^{(1)} x_2 + a_{13}^{(1)} x_3 = b_1^{(1)} \\ x_2 + a_{23}^{(2)} x_3 = b_2^{(2)} \\ x_3 = b_3^{(3)} \end{cases}. \quad (5)$$

Система (5) называется *треугольной***.** Процесс приведения системы (1) к треугольному виду (5) называют *прямым ходом метода Гаусса*. Нахождение неизвестных из треугольной системы называют *обратным ходом метода*

Гаусса. Для этого найденное значение x_3 подставляют во второе уравнение системы (5) и находят x_2. Затем x_2 и x_3 подставляют в первое уравнение и находят x_1.

В общем случае для системы m линейных уравнений с n неизвестными проводятся аналогичные преобразования. На каждом шаге исключается одно из неизвестных из всех уравнений, расположенных ниже ведущего уравнения. Поэтому метод Гаусса имеет другое название – метод последовательного исключения неизвестных.

Если в ходе преобразований системы получается противоречивое уравнение вида $0 = b$, где $b \neq 0$, то это означает, что система несовместна и решений не имеет.

Формат файла входных данных

Программная реализация алгоритма должна позволять запускать алгоритм на различных входных данных наиболее удобным образом (в первую очередь – без необходимости вносить исправления в программный код). Поэтому было принято решение разработать свой формат файла с входными данными для программы, который впоследствии было бы легко обработать перед решением задачи.

Разработанный формат файла имеет следующие детали и особенности:

- файл должен быть текстовым и называться «input.txt»;
- каждая строка файла описывает ветвь между двумя узлами участка цепи, на которой расположен конденсатор;
- узлы нумеруются слева направо и сверху вниз в соответствии со схемой, нумерация начинается с 0;
- строка файла имеет следующий вид: C_i N_j N_k, где C_i – емкость i-го конденсатора (единица измерения в данном случае [мкФ]), а N_j и N_k –

номера узлов, между которыми расположен i-й конденсатор (обычно $N_j < N_k$).

Например, у нас есть участок цепи с конденсаторами, изображенный на рис. 5.

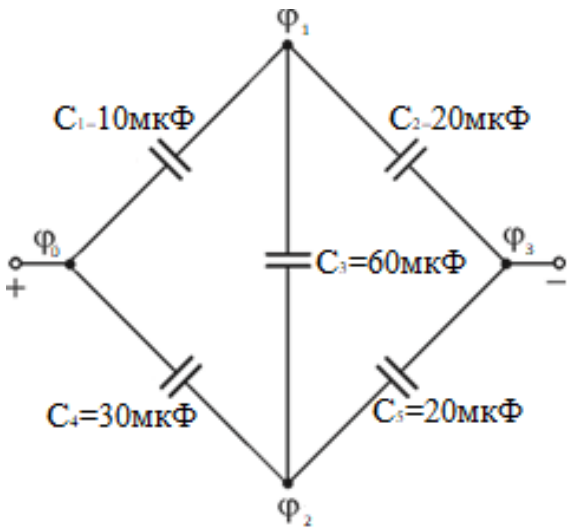

Рис. 5. Пример участка цепи с указанными ёмкостями конденсаторов

Набор входных данных для такого участка цепи будет выглядеть следующим образом:

10 0 1

20 1 3

60 1 2

30 0 2

20 2 3

Алгоритм нахождения ёмкости эквивалентного конденсатора с учетом формата входных данных

Напряжение на участке цепи принимается равным 1 В, причем стартовый потенциал принимается равным 1 В, а конечный потенциал – равным 0 В.

Разработанный алгоритм имеет следующий вид:

1. На основе файла с входными данными заполняется массив с информацией о конденсаторах.
2. Определяется количество неизвестных потенциалов на основе входных данных (находим наибольший номер потенциала и вычитаем из него единицу, так как нумерация потенциалов идет с 0, а значения входного и выходного потенциалов известны).
3. Формируется 3 изначально пустых массива для СЛАУ: матрица коэффициентов при искомых потенциалах, вектор свободных членов и вектор неизвестных потенциалов.
4. Построчно заполняется матрица в соответствии с законом сохранения зарядов и с учетом входных данных:
    4.1. При заполнении элемента на основной диагонали матрицы: если в ходе проверки массива входных данных встречается ситуация, когда номер первого потенциала равен номеру строки плюс 1 (считая от 0), тогда [i, i]-элемент массива увеличивается на емкость подходящего конденсатора.
    4.2. При заполнении остальных элементов: если в ходе проверки массива входных данных встречается ситуация, когда номер первого и второго потенциалов равны номеру строки плюс 1 (считая от 0), тогда [i, j]-элемент массива уменьшается на емкость подходящего конденсатора.

4.3. При заполнении элемента вектора свободных членов: если в ходе проверки массива входных данных встречается ситуация, когда номер первого потенциала равен нулю, а второй номеру строки плюс 1 (считая от 0), тогда [i]-элемент массива увеличивается на емкость подходящего конденсатора.

5. Матрица коэффициентов и вектор свободных членов преобразуются при помощи алгоритма прямого хода метода Гаусса.
6. При помощи обратного хода метода Гаусса находятся неизвестные потенциалы.
7. Находятся заряды на конденсаторах, подключенных непосредственно ко входу схемы.
8. Находится общая ёмкость конденсаторов цепи по формуле $C_{общ} = q_{общ}/U_{общ}$.

Программная реализация алгоритма

Программная реализация разработана на языке программирования C++. Этот язык содержит средства разработки программ для широкого спектра задач: от низкоуровневых утилит и драйверов до весьма сложных программных комплексов. Особенности языка C++:

- вычислительная производительность: язык спроектирован так, чтобы дать программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы; имеется возможность работы с памятью на низком уровне;
- поддержка различных стилей программирования: традиционное императивное программирование (структурное, объектно-ориентированное), обобщённое программирование, функциональное программирование, порождающее мета-программирование;

- доступность: для C++ существует огромное количество учебной литературы, переведённой на всевозможные языки; несмотря на то, что язык имеет высокий порог вхождения, среди всех языков такого рода C++ обладает наиболее широкими возможностями;
- удобные инструменты для работы с файлами: в C++ для этих целей используются специальные типы данных, называемые потоками; поток ifstream служит для работы с файлами в режиме чтения, а ofstream в режиме записи.

На выбор языка кроме указанных выше преимуществ языка также повлиял тот факт, что C++ сейчас активно изучается в школах (в том числе и в нашей школе).

Программа выполнена в среде разработки Microsoft Visual Studio (Community version), объём кода составил 125 строк. Листинг кода программной реализации представлен в приложении.

Выводы

Все поставленные цели и задачи в рамках проекта были мной выполнены. В процессе работы над проектом я подробнее рассмотрел способы решения задач расчета электрических цепей с конденсаторами, ознакомился с матричным представлением систем линейных алгебраических уравнений и численными методами решения таких систем. Мной был разработан алгоритм нахождения ёмкости эквивалентного конденсатора для участка цепи, который лег в основу численного решателя поставленной задачи. Решатель был мной написан на языке программирования C++, и в ходе работы над ним я улучшил свои навыки программирования на этом языке. Для дальнейшего развития проекта я планирую исследовать возможность построения подобного решателя не только для статического режима, но и с учетом переходных процессов.

Список источников

1. Мякишев Г. Я. Физика. 10 класс: учеб. для общеобразоват. организаций с прил. на электрон. носителе : базовый уровень / Г. Я. Мякишев, Б. Б. Буховцев, Н. Н. Сотский; под ред. Н. А. Парфентьевой. – М. : Просвещение, 2014. – 416 с. : ил.
2. В. Зварич, В. Ляховец. Расчет электроемкости конденсаторов // Научно-популярный физико-математический журнал «Квант», №5-6, 2011. URL: http://kvant.mccme.ru/pdf/2011/056/Zvarich.pdf (дата обращения: 28.02.2020)
3. Поляков К.Ю. Информатика. Углублённый уровень: учебник для 11 класса. В 2-х частях / К.Ю. Поляков, Еремин Е.А. – М.: БИНОМ. Лаборатория знаний, 2013. – 240 и 304 с.
4. Статьи, посвященные закону сохранения заряда, СЛАУ и методам решения СЛАУ, размещенные на сайте https://ru.wikipedia.org/

Приложение А

Листинг основного файла программной реализации

```
#include <iostream>
#include <vector>
#include <fstream>
#include <cmath>

using namespace std;

typedef struct {
	int c, f1, f2;
}Tin;

int main() {
	ifstream Fin;
	Fin.open("input.txt");
	int i = 0, kolC, kolF = 0, j, k, U = 1;
	float del, a = 0, Q = 0, Ce = 0;
	Tin C;
	vector <Tin>  Input;

	while (!Fin.eof()) {
		Fin >> C.c;
		Fin >> C.f1 >> C.f2;
		Input.push_back(C);
	}
	kolC = Input.size();

	for (i = 0; i < kolC; i++) {
		if (Input[i].f1 > kolF) kolF = Input[i].f1;
		if (Input[i].f2 > kolF) kolF = Input[i].f2;
	}
	kolF--;

	if (kolF == 0) {
		for (i = 0; i < kolC; i++)
```

```
                Ce += Input[i].c;
            cout << Ce << endl;
      }
      else {

            float** matrix;
            matrix = new float* [kolF];
            for (i = 0; i < kolF; i++)
                  matrix[i] = new float[kolF];
            float* Vec;
            Vec = new float[kolF];
            float* Fi;
            Fi = new float[kolF];

            for (i = 0; i < kolF; i++) {
                  for (j = 0; j < kolF; j++)
                        matrix[i][j] = 0;
                  Vec[i] = 0;
                  Fi[i] = 0;
            }

            //формирование матрицы
            for (i = 0; i < kolF; i++) {
                  for (j = 0; j < kolF; j++) {
                        if (i == j) {
                              for (k = 0; k < kolC; k++) {
                                    if (Input[k].f1 == (j + 1) || Input[k].f2
== (j + 1)) matrix[i][j] += Input[k].c;
                              }
                        }
                        else {
                              for (k = 0; k < kolC; k++) {
                                    if   ((Input[k].f1   ==   (i   +   1))   &&
(Input[k].f2 == (j + 1)) || (Input[k].f1 == (j + 1)) && (Input[k].f2 == (i +
1))) matrix[i][j] -= Input[k].c;
                              }
                        }
                  }
            }
```

```
            for (i = 0; i < kolF; i++) {
                  for (k = 0; k < kolC; k++) {
                        if ((Input[k].f1 == 0) && (Input[k].f2 == (i + 1)))
Vec[i] += U * Input[k].c;
                  }
            }

            // после формирования матрицы и вектора свободных членов
            for (i = 0; i < kolF; i++) {
                  for (j = 0; j < kolF; j++) cout << matrix[i][j] << " ";
                  cout << Vec[i] << endl;
            }
            cout << endl;

            // прямой ход метода Гаусса
            for (k = 1; k < kolF; k++) {
                  for (i = k; i < kolF; i++) {
                        if (matrix[i][i - 1] != 0) {
                              del = matrix[i][i - 1];
                              for (j = i - 1; j < kolF; j++) matrix[i][j] =
((matrix[i][j] / del) * matrix[i - 1][i - 1]) - matrix[i - 1][j];
                              Vec[i] = ((Vec[i] / del) * matrix[i - 1][i - 1])
- Vec[i - 1];
                        }
                  }
            }

            // после прямого хода Гаусса
            for (i = 0; i < kolF; i++) {
                  for (j = 0; j < kolF; j++) cout << matrix[i][j] << " ";
                  cout << Vec[i];
                  cout << endl;
            }

            // обратный ход метода √аусса
            for (i = (kolF - 1); i > (-1); i--) {
                  a = 0;
                  for (j = 0; j < kolF; j++) {
```

```
                        if (j != i) a += (matrix[i][j] * Fi[j]);
                  }
                  Fi[i] = (Vec[i] - a) / matrix[i][i];
            }

            for (k = 0; k < kolC; k++) {
                  if (Input[k].f1 == 0) {
                        if (Input[k].f2 == (kolF + 1)) Q += abs(U * Input[k].c);
                        else Q += abs((U - Fi[(Input[k].f2 - 1)]) * Input[k].c);
                  }
            }

            Ce = Q / U;
            cout << endl << Ce << endl;
      }

      system("pause");
      return 0;
}
```