Simple Circuit simulation

Репозиторий: ссылка

Разработчик: Роман Лещук

Источники: ссылка, ссылка, Электричество и магнетизм (Демидчик А.В.) 2020г.

В данном фреймворке для симуляций используется пошаговый расчет электрической цепи с использованием метода штамповки (stamping) и решения системы линейных уравнений.

Метод штамповки - один из способов расчета электрических цепей, который применяется для нахождения токов в сложных разветвленных схемах. Он основан на использовании законов Кирхгофа и позволяет свести систему уравнений к удобной матричной форме.

Суть метода:

1) **Составление уравнений**: Записываются уравнения по первому и второму законам Кирхгофа для узлов и контуров схемы.
2) **Формирование матрицы проводимостей**: Используется матричное представление схемы, где узловые потенциалы связаны с токами через матрицу проводимостей (матрицу Гесса).
3) **"Штамповка" матрицы**: Добавление элементов схемы (резисторов, источников напряжения и тока) в общую матрицу. Процесс называется "штамповкой", потому что каждый элемент вносит свой вклад в строго определенные позиции матрицы.
4) **Решение системы уравнений**: После заполнения матрицы система решается численными методами, например методом Гаусса или LU-разложением.

Класс CircuitComponent

Для начала был создан абстрактный класс CircuitComponent, который реализует общее поведение компонентов цепи, а так же содержит номера контуров, к которым подключен компонент. Для условности, контур с номером -1 обозначает землю.

В данном классе содержится метод Stamp для штамповки компонента в матрицу системы уравнений. Происходит это следующим образом:

$$A * X = -B$$

где **A** – матрица коэффициентов, **B** – вектор свободных членов, а **X** – вектор неизвестных (контурные токи). Остальная содержится в summary класса/методов.

Класс Resistor

Резистор описывается законом Ома:

$$U = I * R$$

При штамповке в уравнениях метода контурного анализа добавляется вклад сопротивления. Если резистор соединяет два свободных контура, то он вносит вклад **R** к диагональным элементам и **-R** к перекрёстным (сцепляющим) элементам матрицы **A**. Если один из контуров задан (форсирован), его вклад переносится в вектор B. Остальная информация содержится в summary класса/методов.

Класс Capacitor

Конденсатор определяется соотношением:

$$I = C * dU / dt \Rightarrow dU / dt = I / C$$

Для обновления напряжения на конденсаторе используется явная схема (метод Эйлера):

$$\mathrm{U_{new}} = \mathrm{U_{old}} + \frac{I}{C}\mathrm{dt}$$

В данном случае разность токов (между двумя контурами) влияет на изменение напряжения. При штамповке в систему уравнений конденсатор не вносит вклад в матрицу A, а только добавляет постоянный член (текущую разность потенциалов) в B. Остальная информация содержится в summary класса/методов.

Класс Inductor

Индуктивность описывается уравнением:

$$U = L * \frac{dI}{dt}$$

Для численной симуляции используется схема обратного Эйлера:

$$\mathrm{dI/dt} \approx (\mathrm{I_{new}} - \mathrm{I_{old}}) / \mathrm{dt}$$

откуда получаем:

$$\mathrm{U} \approx \frac{\mathrm{L}}{\mathrm{dt}} * (\mathrm{I_{new}} - \mathrm{I_{old}})$$

При штамповке в матрицу A добавляется коэффициент *L/dt*, а вклад из прошлой итерации ***(L/dt * I_old)*** переносится в свободный член B. Остальная информация по содержится в summary класса/методов.

Класс CurrentSource

Класс источника тока. Источник задаёт ток по функции времени: I(t) = f(t). Если один из выводов подключён к земле, то источник принудительно задаёт значение тока в данном контуре. Если же источник подключён между двумя свободными контурами, его влияние в этой модели обрабатывается отдельно (например, в методе Step класса Circuit). В своем конструкторе принимает функцию I(t) = f(t), которая позволяет настроить источник тока на переменное или постоянное напряжение.

Остальная информация по содержится в summary класса/методов.

Класс Circuit

Основной класс цепи. Здесь собираются компоненты, формируется система уравнений, выполняется шаг симуляции. Метод Step реализует численное решение системы методом конечных разностей.

Работа метода Steep():

1. Сохраняем предыдущее состояние (для расчёта производных).
2. Применяем источники тока, подключённые к земле, которые форсируют значение контура.
3. Определяем «свободные» контуры – те, значение которых не задано напрямую.
4. Формируем систему уравнений методом штамповки (Stamp) для всех компонентов, кроме источников тока.
5. Решаем полученную систему уравнений (методом Гаусса).
6. Обновляем значения токов в свободных контурах.
7. Обновляем напряжения на конденсаторах (интегрируя ток по времени).
8. Увеличиваем время симуляции на dt.

Остальная информация содержится в summary класса/методов.

Различные примеры

Проект уже содержит различные проекты, которые можно запустить из метода Main класса Program.

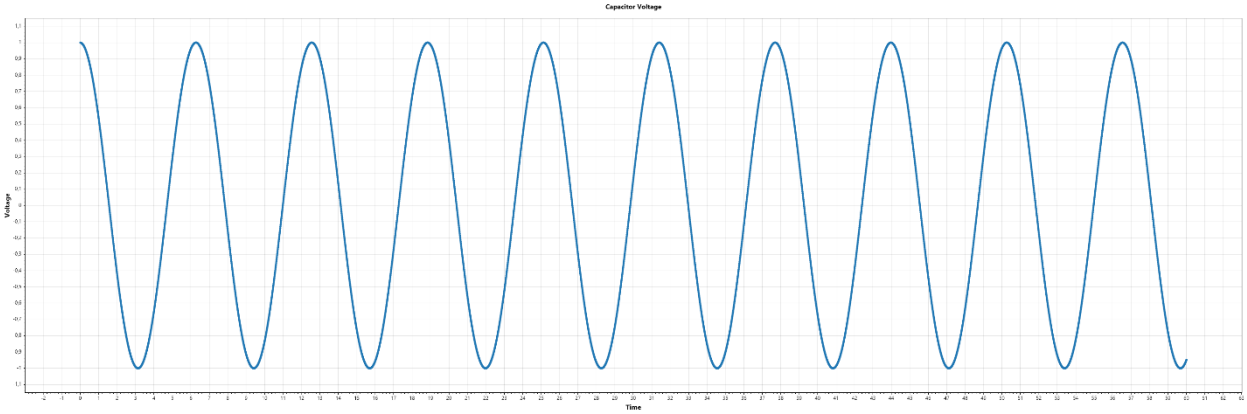
Так же содержатся различные инструменты для экспорта данных в картинку, лист Excel, а так же формат .xy для Origin.

Для примера получим график напряжения для линейного гармонического осциллятора. Для этого напишем такой код:

```
Shershnyaga * More...
public abstract class Program
{
    Shershnyaga *
    public static void Main(string[] args)
    {
        var circuit = new Circuit(timeStep: 0.01f); // 0.01f - шаг симуляции.

        // Для корректной работы метода штамповки необходимо указать источник напряжения,
        // который не будет выдавать ток, однако позволит указать направление токов для начала симуляции.
        circuit.AddComponent(new CurrentSource(mesh1: 1, mesh2: -1, _ => 0.0));

        circuit.AddComponent(new Inductor(mesh1: 0, mesh2: 1, inductance: 1));
        var cap = new Capacitor(mesh1: 0, mesh2: 1, capacitance: 1)
        {
            Voltage = 1.0 // Задаём начальное напряжение 1 В, что запускает колебания.
        };
        circuit.AddComponent(cap);

        // Инициализируем состояние симуляции.
        circuit.InitializeState();

        // Запуск 60-ти секундной симуляции.
        var voltages = new Dictionary<double, double>(); // Словарь с зависимостью напряжения от времени.
        while (circuit.State.Time < 60)
        {
            circuit.Step();

            voltages.Add(circuit.State.Time, cap.Voltage); // Записываем текущее значение в словарь.
        }
```

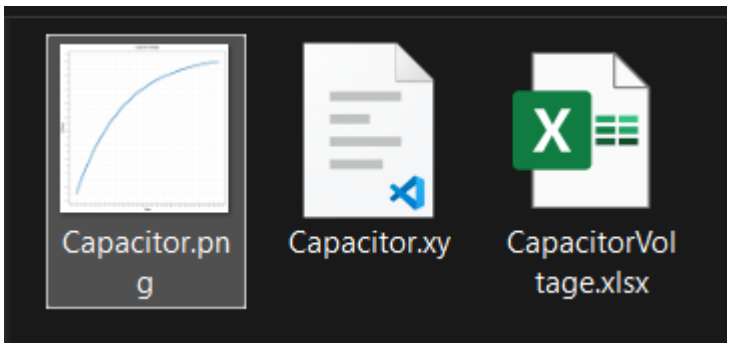
```
        // Создание графика.
        var exporter = new PngExporter(filePath: "capacitor/", width: 3000, height: 1000);
        exporter.Plt.Title(text: "Capacitor Voltage");
        exporter.Plt.XLabel("Time");
        exporter.Plt.YLabel("Voltage");
        exporter.Add(name: "CapacitorVoltage", voltages);
        exporter.Export();
```

Результатом станет следующий график:



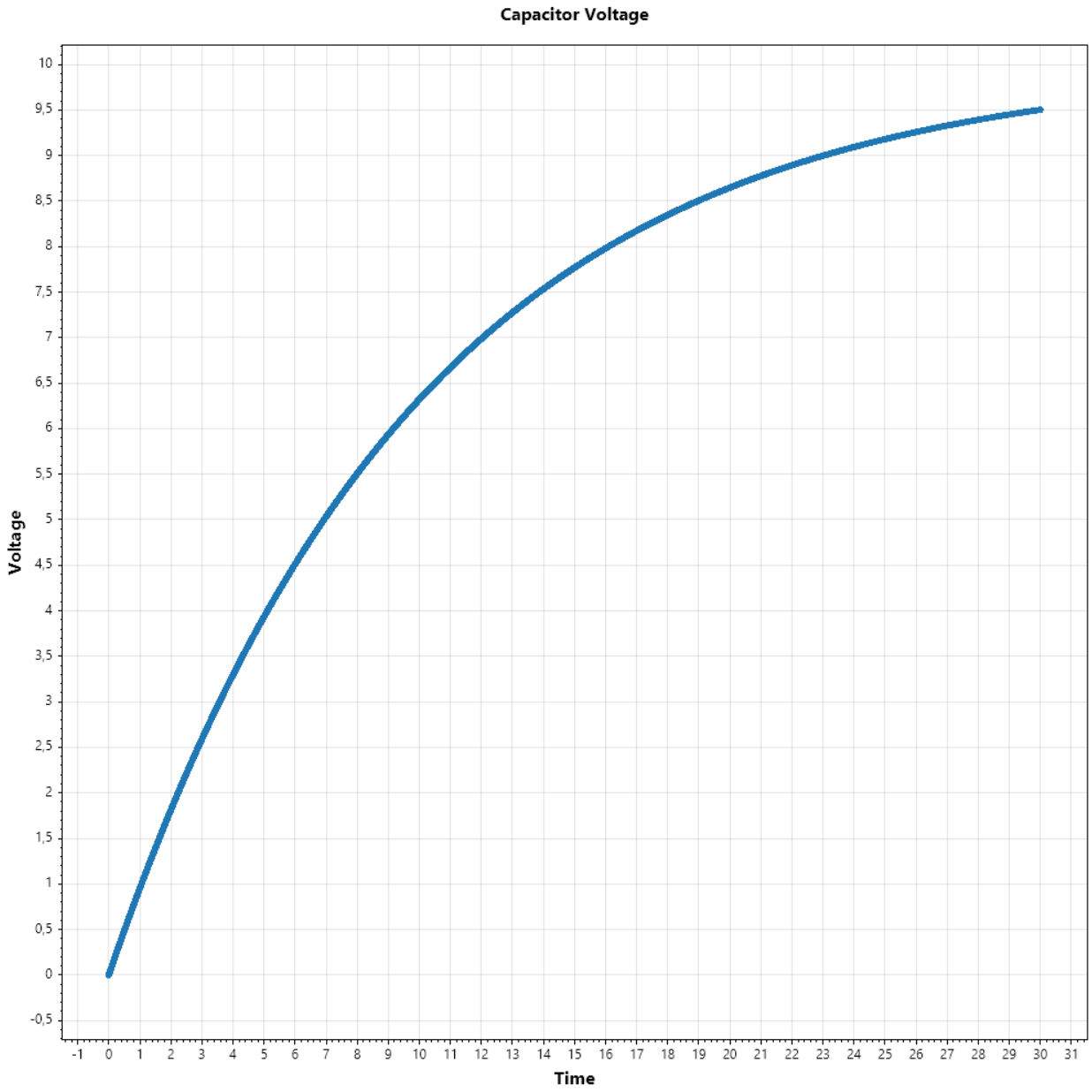
Следующий пример выводит график зарядки конденсатора.

На выходе получаем файлы:



Среди них:

1) Рисунок



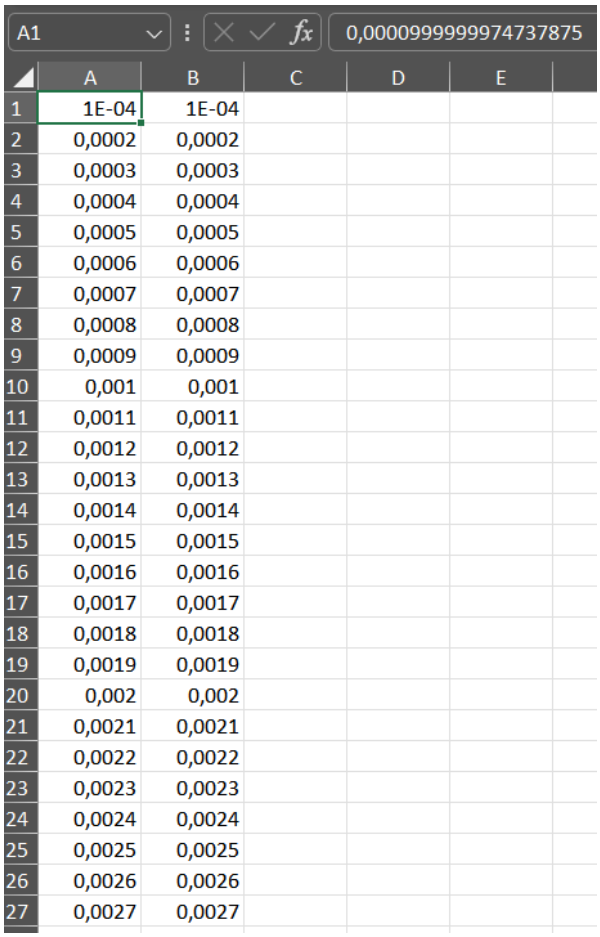
2) Файл .xy файл для Origin

Capacitor.xy

D: > Диплом > CircuitSimulation > CircuitSimulation > bin > Debug > net7.0 > CapacitorVoltage >

```
9,999999747378752E-05 9,999999747378752E-05
0,00019999999494757503 0,00019999899494762554
0,00029999999242136255 0,000299999699243151 41
0,00039999998989515007 0,00039999398993545307
0,00049999999873689376 0,00049999899874694423
0,00059999999848427251 0,00059999849850434815
0,00069999999823165126 0,00069999789826675701
0,00079999999797903001 0,00079999719803517078
0,00089999999772640876 0,00089999639781058940
0,00099999999747378752 0,00099999549759401277
0,00109999999722116627 0,00109999449738644084
0,00119999999696854502 0,00119999339718887350
0,00129999999671592377 0,00129999219700231069
0,00139999999646330252 0,00139999089682775225
0,00149999999621068127 0,00149998949666619808
```

3) Файл .xlsx для Excel.



A1 | 0,0000999999974737875

	A	B	C	D	E
1	1E-04	1E-04			
2	0,0002	0,0002			
3	0,0003	0,0003			
4	0,0004	0,0004			
5	0,0005	0,0005			
6	0,0006	0,0006			
7	0,0007	0,0007			
8	0,0008	0,0008			
9	0,0009	0,0009			
10	0,001	0,001			
11	0,0011	0,0011			
12	0,0012	0,0012			
13	0,0013	0,0013			
14	0,0014	0,0014			
15	0,0015	0,0015			
16	0,0016	0,0016			
17	0,0017	0,0017			
18	0,0018	0,0018			
19	0,0019	0,0019			
20	0,002	0,002			
21	0,0021	0,0021			
22	0,0022	0,0022			
23	0,0023	0,0023			
24	0,0024	0,0024			
25	0,0025	0,0025			
26	0,0026	0,0026			
27	0,0027	0,0027			