Министерство образования и науки Российской Федерации

ТОМСКИЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра промышленной электроники

Ю.Н. Тановицкий, Д.А. Савин

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ПРОГРАММИРОВАНИЕ

Руководство к организации самостоятельной работы для студентов специальности 210106 «Промышленная электроника»

2011

Тановицкий Ю.Н., Савин Д.А.
Математическое моделирование и программирование: Руководство к организации самостоятельной работы. — Томск: Томский университет систем управления и радиоэлектроники. — 49 с.

ОГЛАВЛЕНИЕ

Рабочая программа по дисциплине: "Математическое моделирование и программирование"

Специальность 210106 "Промышленная электроника"

Факультет: Электронной техники

Профилирующая кафедра: Промышленная электроника

Курс: второй, третий

Семестр: четвертый, пятый

Учебный план набора 2008 года и последующих лет

Распределение учебного времени	Всего часов
Лекции (4 семестр)	24
Лабораторные (4 семестр)	28
Курсовое проектирование (5 семестр)	18
Всего аудиторных занятий	70
Самостоятельная работа	74
Общая трудоемкость	144

Зачет 4 семестр

Диф.зачет 5 семестр

1 Цели и задачи курса

1.1 Цель изучения курса

Цель изучения данного курса - дать студентам основы работы с программами автоматизации математических расчетов при проектировании, анализе и моделировании электронных схем, познакомить с основами программирования на современной высокотехнологичной объектно-ориентированной базе.

1.2 Задачи курса

Освоение приемов расчетов и математического моделирования с использованием MathCAD, C++, ASIMEC. Сформировать навыки работы с математическими моделями динамических систем на примере электронных схем.

1.3 Связь с другими дисциплинами

Изучение дисциплины базируется на знаниях, приобретенных в курсах "Математика", "Информатика" в качестве объектов моделирования активно используются электрические схемы, изучаемые в курсе "Теория электрических цепей". Материал, изученный в рамках данного курса имеет применение практически во всех дисцплинах, изучаемых впоследствии "Методы анализа электронных схем", "Теория автоматического управления", «САПР Электронных схем», «Процедурное- и «Объектно-ориентированное программирование» и др.

2 Содержание лекционного курса

2.1 Математическое моделирование в инженерных задачах. (24 часа)

2.1.1 Документно-ориентированное проектирование. (2 часа)

Понятие о современном документно-ориентированном проектировании. Основные этапы проектов. Ошибки при проектировании. Стандарты проектирования. Особенности участия в больших проектах. Понятие о жизненного цикла продукта.

2.2.3. Структура электронных документов на примере HTML/XML стандартов (2 часа)

Теги и разметка документов. Дерево как модель структуры документа. Стили документов. DOM-модель документа. Манипулирование свойствами документов и язык Java-Script. Пример создания простого документа. Справочная информация в сети Интернет.

2.1.3 Задача управления и классификация основных методов ее решения. (2 часа)

Постановка задачи управления. Классификация методов управления. Понятия об обратной связи, адаптации (оптимизации), инкапсуляция, абстрактные модели.

2.1.4 Математические модели в инженерных расчетах. (2 часа)

Цели создания и назначение моделей. Понятия: объект, модель, оригинал, система, структура, параметры и переменные, характеризующие состояние. Динамические модели в форме алгебраических и алгебро-дифференциальных уравнений. Модели процессов в форме алгоритмов.

2.1.5 Автоматизация формирования математических моделей, на примере электронных схем (2 часа)

Топологические уравнения и методы их получения. Net-лист. Структурная матрица и уравнения по первому закону Кирхгофа для токов. Уравнения по второму закону Кирхгофа.

2.1.6 Задача Коши – численно-аналитические методы (2 часа)

Экспоненциальная матрица и ее свойства. Решение уравнений вида: dX/dt=AX+B с помощью экспоненциальной матирцы. Вычисление экспоненциальной матрицы. Пример численно-аналитических расчета с помощью экспоненциальной матрицы.

2.1.7 Задача Коши – численные схемы интегрирования. (2 часа)

Численные схемы Эйлера, трапеций. Устойчивость численных схем. Особенности интегрирования консервативных систем. Сравнение численно-аналитических и численных методов.

2.1.8 Метод узловых потенциалов (2 часа)

Решение задачи Коши. Анализ линейных цепей на переменном синусоидальном токе. Пакеты программ (ASIMEC, EWB, PSPICE) реализующие изученные методики.

2.1.9 Типичные затруднения, возникающие при решении задачи Коши с помощью пакетов программ и способы их устранения (2 часа)

Выбор шага интегрирования в программах автоматизированного моделирования. Проблема «жесткости» уравнений. Сверхдобротность. Управление точностью. Примеры моделирования схем.

2.2 Понятие об объектно-ориентированном программировании. (6 часов)

2.2.1 Модели памяти компьютеров и структуры данных (2 часа).

Модель памяти: регистры, динамическая память RAM, дисковая, удаленная (сетевые ресурсы), кэширование. Области памяти для хранения программ, для данных стеки и «кучи». Типизированные и не типизированные языки. Встроенные типы данных (целые и с плавающей точкой). Составные данные (структуры в C++). Обен данными, указатели и ссылки.

2.2.2 Контейнеры (2 часа)

Массивы. Связные списки. Карты. Деревья. Специализированные контейнеры. Контейнеры из библиотеки C++ STL.

2.2.3 Основы объектно-ориентированного программирования (2 часа)

Инкапсуляция данных. Инкапсуляция процессов. Интерфейсы.

3. Перечень лабораторных работ

Лабораторная работа №1. Основы языка гипертекстовой разметки HTML/XML. Структура XML документов. Создание html-странички, согласно варианту. (4 часа, баллов 8)

Лабораторная работа №2. Разделение представления и содержимого в документах. Каскадные таблицы стилей. (4 часа, баллов 8).

Лабораторная работа №3. Объектная модель документа DOM. (4 часа, баллов 9)

Лабораторная работа №4. Интерактивность в документах и язык Java-Script. (4 часа, баллов 9)

Лабораторная работа №5. Автоматизация формирования математических моделей электронных схем. (4 часа, баллов 9)

Лабораторная работа №6. Построение частотных характеристик электрической цепи MathCad (4 часа, баллов 9)

Лабораторная работа №7. Знакомство со средой программирования Microsoft VisualStudio и особенности работы консольных приложений на языке C++ в ней. (4 часа, рейтинг 4, баллов 9)

4. Рейтинговые индивидуальные задания

Задание №1. Разработка сайта html, удовлетворяющего заданным параметрам (баллов 15).

Задание № 2. Построение графика переходного процесса в среде Matcad методом узловых потенциалов. (баллов 15)

5. Контрольные работы

1. Составить алгоритм и краткое математическое описание заданной функции матричной алгебры. (баллов 10)

2. Изобразить текст С++ программы, реализующей заданную операцию над трехмерными вектор векторами. (баллов 3)

3. Привести текст программы С++, позволяющей принимать и хранить в памяти произвольное количество объектов заданного типа. (баллов 7)

6. Балльная раскладка по дисциплине (четвертый семестр)

Элементы учебной деятельности	Максимальный балл на 1-ую КТ с начала семестра	Максимальный балл за период между 1КТ и 2КТ	Максимальный балл за период между 2КТ и на конец семестра	**Всего за семестр**
Посещение занятий	3	3	3	**9**
Индивидуальные задания	0	15	15	**30**
Контрольные работы на практических занятиях	0	10	10	**20**
Выполнение и защита результатов лабораторных работ	20	20	21	**61**
Итого максимум за период:	23	48	49	**120**
Нарастающим итогом	**23**	**48**	**49**	**120**

Перевод текущего рейтинга в оценку по КТ: При рейтинге 80%-100% от максимального за текущую КТ – оценка отлично, 60%-79% - хорошо, 40%-59% - удовлетворительно, меньше 39% неудовлетворительно

7. Самостоятельная работа студентов

№	Содержание работы	Объем часов	Форма контроля
1.	Подготовка к лабораторным занятием	8	Защита лабораторных работ
2.	Выполнение рейтинговых индивидуальных заданий	14	Проверка работ
3.	Выполнение курсовой работы	16	Защита курсовой работы
Всего		38	

8. Содержание курсового проекта

Курсовой проект выполняется в 5-м семестре. Цель проекта – получение практических навыков в применении математического процессора MathCAD для интерактивного моделирования процессов в электрических и электронных схемах.

В процессе выполнения проекта происходит освоение специальной технологии формирования рабочего листа MathCAD , при которой весь расчет основан на применении функций пользователя. Вычислительный процесс должен строиться аналогично процессу обработки данных в электронных таблицах.

Процесс проектирования состоит из нескольких этапов (см. п.8.1). При выполнении проекта необходимо составить систему уравнений Кирхгофа для одного из 30 вариантов схемы электрической цепи с индуктивностью и емкостью, представить эту систему уравнений в матричной форме и сформировать пользовательскую функцию MathCAD для порождения матрицы системы уравнений Кирхгофа. Далее необходимо построить ряд операторов, формирующих из матрицы системы уравнений Кирхгофа матричные элементы модели цепи в пространстве состояний. Затем получить ряд функций для аналитического решения системы дифференциальных уравнений, осуществить выбор таких величин активных сопротивлений, при которых переходный процесс станет колебательным и цепь приобретет резонансные свойства. На завершающем этапе следует выполнить Моделирование средствами MathCad резонансной электрической цепи под воздействием входного периодического сигнала в виде последовательности прямоугольных импульсов. Повторение экспериментов в ASIMEC.

8.1.Временная и балльная раскладка этапов работы над курсовым проектом

Этапы работы над курсовым проектом	**Срок выполнения (недели семестра)**	Баллы
1. Построение системы уравнений Кирхгофа	1 – 2	10
2. Преобразование системы уравнений Кирхгофа в матричную форму. Построение функции расчета матрицы в MathCad	2 – 3	10
3. Выражение части связанных переменных через переменные состояния и входное воздействие с помощью функций MathCad.	3	10
4. Получение основных матриц модели цепи в пространстве состояний	3 – 4	12
5. Построение функций для расчета матричной экспоненты	4 – 5	12
6. Аналитическое решение системы дифференциальных уравнений модели цепи в пространстве состояний для ступенчатого входного воздействия	6 – 7	14
7. Исследование резонансных явлений в цепи. Подбор величин сопротивлений резисторов для обеспечения колебательного переходного процесса	8 – 9	14
8. Исследование переходных и установившихся процессов в цепи при воздействии на входе периодической последовательности прямоугольных импульсов	10 – 11	14
9. Оформление пояснительной записки	12 – 14	10
10.Подготовка к защите и защита курсового проекта	14 – 16	14
Всего	17	120

Перевод текущего рейтинга в оценку по КТ:

При рейтинге 80%-100% от максимального за текущую КТ – оценка отлично, 60%-79% - хорошо, 40%-59% - удовлетворительно, меньше 39% неудовлетворительно

9 Литература

9.1. Основная литература

1. Тановицкий Ю.Н., Егоров И.М., Савин Д.А. Математическое моделирование и программирование. Руководство к организации самостоятельной работы. http://www.ie.tusur.ru/ docs/mmip.zip
2. Егоров И.М.Программирование: Учебно-методическое пособие (Курсовое проектирование). – Томск: Томский государственный университет систем управления и радиоэлектроники, 2007. – 80 с

9.2. Дополнительная литература

1. Дьяконов В.П., Абраменкова И.В. MathCAD 7.0 в математике, физике и в Internet. - М.: Нолидж, 1998. - 352 с.
2. Очков В.Ф.Mathcad 8 Pro для студентов и инженеров. - М.: КомпьютерПресс, 1999. - 523 с.
3. Кудрявцев Е.М. Mathcad 8.Символьное и численное решение разнообразных задач. - М.: ДМК, 2000. - 320 с.
4. Дьяконов В. Mathcad 2001: Учебный курс/ В. Дьяконов. -СПб.: Питер, 2001.-621 с.
5. Бабэ Б. Просто и ясно о Borland C++ БИНОМ, 1995, 400 с.
6. Луис Д. С и C++ /Перевод с нем. - М.:Восточная Книжная Компания.БИНОМ,1997.-592 с.
7. Вайнер Р.,Пинсон Л. C++ изнутри. НПИФ"ДиаСофт", 1993, 304 с.
8. Дьюхарст С., Старк К. Программирование на C++, НИПФ "ДиаСофт", 1993, 272 с.
9. Лукас П. C++ под рукой. НИПФ "Диа-Софт", 1993, 176 с.
10. Справочник по функциям Borland C++ 3.1/4.0, "Диалектика", 1994, 416 с.
11. Шилдт Г. Самоучитель C++, BHV-Санкт-Петербург, 1997, 512 с.
12. Пол А. Объектно-ориентированное программирование на C++. Невский диалект. М: "Издательство БИНОМ", 1999, 464 с.

ЛАБОРАТОРНАЯ РАБОТА № 1
Изучение HTML

1 ВВЕДЕНИЕ

Целью работы является знакомство с древовидными и сетевыми структурами данных на примере языка гипертекстовой разметки (HTML – HyperText Markup Language — «язык разметки гипертекста»)

Язык HTML разрабатывался для создания документов распространяеых через Интернет. Используя HTML можно создавать документы в любом текстовом редакторе. Просмотр документов осуществляется с помощью специальных приложений, назваемых веб браузерами (от англ Web Browser). Наиболее распространенными браузерами являются Internet Explorer и Mozilla Firefox. Язык HTML является подмножеством XML – стандарты таких языков доступны на сайте (www.w3c.org).

2 ОПИСАНИЕ ЯЗЫКА HTML

Любой HTML документ состоит из набора элементов, начало и конец каждого элемента обозначается специальными пометками – тегами. Между тегами находится содержимое элемента – данные или текст (например элемент заголовка: *<title>Hello, world!</title>*, где *<title>* - открывающий тег, *</title>* - закрывающий). Элемент может быть и пустым, т.е. не содержащим внутри себя данных (например элемент перевода строки: *
*). Помимо данных, элемент может иметь атрибуты, определяющие свойства элемента (например способ выравнивания текста, цвет шрифта, размер картинки). Атрибуты указываются в открывающем теге в виде: **Этот текст – белый**. Здесь *color* – название атрибута, а *#FFFFFF* – его значение, при этом значение должно указываться в кавычках (*#FFFFFF* – это белый цвет заданный в формате RGB в шестнадцатиричном виде). Элементы могут быть вложенными, например: *<p align=”center”>*Этот

текст размещен по центру<*br/*><*b*>Этот текст выделен жирным и размещен по центру<*/b*><*/p*>. Теги и атрибуты можно указывать только в нижнем регистре.

3 СТРУКТУРА HTML ДОКУМЕНТА

Для указания совестимости документа с версией HTML в начало документа помещается следующая строка:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

которая содержит ссылку на стандарт в соответсвии, с которым создан документ.

Затем указывается элемент *html*, содержащий в себе заголовок и тело документа:

```
<html>
  <head>
    <title>Hello, world!</title>
  </head>
  <body>
    Hello, world!
  </body>
</html>
```

Элемент *head* (голова документа) содержит служебную информацию о документе, в него помещается, например, элемент *title* (в этом элементе можно указать текст, который будет выведен в заголовке окна браузера). А элемент *body* – это тело документа, содеражащие то, что будет видно пользователю.

Некоторые элементы тела документа:

a – Элемент для создания гипертекстовых ссылок. Имеет атрибуты:

href – определяет документ, на который ссылается элемент. Например: *Главная страница* определяет гипертекстовую ссылку на документ с именем main.html. При этом в тексте документа появится надпись «Главная страница», нажатие на которую приведет к переходу на указанную страницу.

name – помечает элемент как метку, на которую можно сослаться. Например: *Параграф 1* позволяет

перейти к этому элементу при использовании гиперссылки следующего вида: *Перейти к параграфу 1*.

h1, h2, ..., h6 – Элементы заголовков. Существует 6 уровней заголовков, отличающихся размером шрифта. Например: *<h1>Самый большой заголовок</h1> <h6>Самый маленький заголовок</h6>*. Атрибуты:

align – определяет способ выравнивания заголовка по горизонтали, возможные значения: *left*, *center*, *right*.

p – Используется для создания параграфов. Атрибуты:

align – определяет выравнивание по горизонтали, возможные значения: *left*, *center*, *right*.

br – Осуществляет перевод строки.

img – Элемент позволяет поместить рисунок в документ. Атрибуты:

src – Обязательный атрибут, указывает файл рисунка;

height – ширина рисунка;

width – высота рисунка;

name – имя рисунка, уникальное для данного документа;

alt – текст, отображаемый браузером на месте рисунка, если рисунок по какой-то причине не может быть отображен;

border – ширина рамки вокруг рисунка.

Пример:

**

ul – Элемент неупорядоченного списка, между начальным и конечным тегами должны содержаться один или несколько элементов *li*.

ol – Элемент упорядоченного списка, между начальным и конечным тегами должны содержаться один или несколько элементов *li*. Атрибуты:

start – число, с которого начинается нумерация;

type – тип нумерации, может принимать значения: *A* – заглавные буквы , *a* – строчные буквы, *I* – большие римские числа, *i* – маленькие римские числа, *1* – арабские числа (по умолчанию).

li – Элемент пункта списка. Определяется внутри элементов *ul* или *ol*. Атрибуты:

value – позволяет указать текущий номер пункта в упорядоченном списке, нумерация последующих пунктов также изменится.

table – Элемент для создания таблицы. По умолчанию размеры таблицы определяются автоматически, в зависимости от информации в ячейках. Ячейки создаются с помощью элементов *tr*, *td*, *th*. Атрибуты:

width – ширина таблицы в пикселях или процентах от ширины элемента, в котором находится таблица;

height – высота таблицы в пикселях или процентах от высоты элемента, в котором находится таблица;

bgcolor – цвет фона таблицы;

border – ширина рамки таблицы.

tr – Элемент строки таблицы. Ячейки в строке создаются с помощью элементов *td* или *th*. Атрибуты:

align – способ выравнивания всех ячеек строки по горизонтали (значения: *left, center, right*);

valign – способ выравнивания всех ячеек строки по вертикали (значения: *top, middle, bottom*).

td, th – Элементы ячейки таблицы, создаются внутри элемента строки *tr*. Элемент *th* отличается от элемента *td* только стилем оформления (*td* – данные, *th* - заголовок). Атрибуты:

align – способ выравнивания ячейки по горизонтали (значения: *left, center, right*);

valign – способ выравнивания ячейки строки по вертикали (значения: *top, middle, bottom*);

width – ширина ячейки в пикселях или процентах от ширины таблицы;

height – высота ячейки в пикселях или процентах от высоты таблицы(при указании высоты в процентах, должна быть задана высота всей таблицы);

colspan – определяет количество столбцов, которые занимает ячейка (по умолчанию 1);

rowspan – определяет количество строк, которые занимает ячейка (по умолчанию 1).

bgcolor – цвет фона ячейки.

4 ПРОГРАММА РАБОТЫ

1. Создайте с помощью текстового редактора (например notepad) на локальном диске файл с именем *index.html*.
2. Запишите в файле структуру HTML документа, напишите текст заголовка и текст в теле документа (С использованием перевода строк и заголовков). Добавьте гиперссылку на сайт тусура (http://www.tusur.ru) и любую картинку.
3. Сохраните файл (С помощью клавиш Ctrl-S или через меню Файл-Сохранить)
4. Запустите веб браузер (Internet Explorer) и через меню Файл-Открыть откройте созданный документ. В результате должна открытся страница, имеющая заголовок и текст документа. Покажите преподавателю.
5. Отредактируйте документ и добавьте список (не менее 10 пунктов) в соответствии с вариантом. Сохраните документ. Обновите страничку в браузере (с помощью клафиши F5). Результат покажите преподавателю.
6. Отредактируйте документ и добавьте таблицу в соответствии с вариантом. Результат покажите преподавателю.

5 ВАРИАНТЫ РАБОТЫ

1. Список №1, таблица №1
2. Список №1, таблица №2
3. Список №2, таблица №1
4. Список №2, таблица №2
5. Список №3, таблица №1
6. Список №3, таблица №2

Варианты списков:

1. Вложенный неупорядоченный список

Список предметов и работ:

- Математическое моделирование и программирование
  - Лабораторная №1
  - Лабораторная №2
  - Лабораторная №3
- Физика
  - Лабораторная №1
  - Лабораторная №2
  - Контрольная №1
- Высшая математика
  - Контрольная №1
  - Контрольная №2
  - Контрольная №3

2. Упорядоченный список, нумерация которого начинается не с единицы, нумерация арабскими цифрами

Список студентов группы:

10. Иванов
11. Петров
12. Сидоров

3. Упорядоченный список, нумерация которого разрывается в середине списка, нумерация большими римскими цифрами

Список студентов группы:

I. Иванов
X. Петров
XI. Сидоров

Варианты таблицы:

1.

Расписание занятий

Время\№ группы		360-1	360-2	360-3	360-4
Понедельник	8:50-10:25	ММиП ЛР 320Ф	Физика ЛР 230Ф	ММиП ЛР 301Ф	Физика ЛР 228Ф
	10:40-12:15	ММиП ЛР 320Ф		ММиП ЛР 320Ф	
	13:15-14:50	ММиП лекция 311Ф			
	15:00-16:35	Высшая математика лекция 204Ф			

2.

Расписание занятий

Время\№ группы		360-1	360-2	360-3	360-4
Вторник	8:50-10:25	ММиП ЛР 320Ф	Физика ЛР 230Ф	ММиП ЛР 301Ф	Физика ЛР 228Ф
	10:40-12:15		Физика ЛР 230Ф		Физика ЛР 228Ф
	13:15-14:50	ММиП лекция 311Ф			
	15:00-16:35	Высшая математика лекция 204Ф			

ЛАБОРАТОРНАЯ РАБОТА № 2
Изучение CSS

1 ВВЕДЕНИЕ

Целью работы является знакомство технологией описания внешнего вида HTML документа – каскадными таблицами стилей (Cascading Style Sheets, CSS).

Каскадные таблицы стилей – это набор правил, описывающих представление элементов HTML-кода и хранящихся отдельно от него. Одно такое правило, описывающее представление одного элемента или группы элементов, называется стилем. Каскадные таблицы стилей используются для оформления веб страниц (HTML), могут использоваться с любыми видами документов в формате XML. С помощью таблиц стилей можно задавать цвета, шрифты, расположение и другие свойства представления элементов документа. Основной целью разработки CSS являлось разделение содержимого (созданного с помощью HTML) и представления документа (созданного с помощью CSS), что позволяет достичь большей гибкости и получить возможность управления представлением документа.

2 ОПИСАНИЕ CSS

CSS при отображении страницы может быть взят из разных источников:

1. Стили предоставляемые автором HTML страницы в виде:
   a. Внешних таблиц стилей, то есть отдельного файла с расширением .css, на который делается ссылка в документе (внутри элемента *head* помещается элемент: *<link href="styles.css" rel="stylesheet" type="text/css"*>, где атрибут *href* содержит имя CSS файла);
   b. Встроенных стилей – блоков CSS внутри документа, при этом внутри элемента *head* помещается элемент *style*, содаржащий стили:

```
<style type="text/css">
<!—
  a {
    color: #FF0000;
  }
-->
</style>
```

с. Inline-стилей, когда стиль одного элемента указывается в его атрибуте style (например: *<li style="color:#00CCFF; font-size:16px;">Лабораторная №1
*);

2. Пользовательские стили (Локальныей .css файл, указанный пользователем в настройках браузера и переопределяющий авторские стили);

3. Стандартные стили браузера (используются браузером по-умолчанию, при отсутствии первых двух источников).

Стандарт CSS определяет приоритеты, в порядке которых применяются правила стилей, если для какого-то элемента подходят несколько правил одновременно. Это называется «каскадом» в котором для правил рассчитываются приоритеты, что делает результат предсказуемым. При этом вложенные элементы, если для них не определены собственный свойства, наследуют свойства элементов, в которые они помещены.

Таблица стилей состоит из набора правил (стилей). Каждое правило в свою очередь состоит из одного или нескольких селекторов, разделенных запятыми, и блока определений. Блок определений заключается в фигурные скобки, и состоит из набора свойств и их значений. Каждое правило имеет следующий синтаксис:

```
Селектор1, селектор2 {
   Свойство1: значение1;
   Свойство2: значение2;
}
```

Виды селекторов:

1. Селекторы применяющиеся ко всем элементам с заданным именем:

```
h1, h2 {
 font-size: 24px;
}
```

Все элементы *h1* и *h2* в документе будут иметь размер шрифта 24 пикселя;

2. Контекстные селекторы:

```
h1 i {
 color: red;
}
```

Все элементы **, находящиеся внутри элементов *<h1>* будут красного цвета (например: *<h1>Заголовок частично красным курсивом</h1>*);

3. Классы:

```
.class_name {
 background-color: black;
}
```

Задает всем элементам, имеющим атрибут *class* со значением, равным имени класса черный цвет фона (например: *<td class=”class_name”>Данные</td>*);

4. ID-классы:

```
p#pname {
 margin: 0;
}
```

Задает всем элементам *p*, имеющим атрибут *id* со значением, равным части селектора после символа решетки, задать нулевой отступ (например: * Параграф без отступа *);

5. Псеводклассы:

```
a:link    { color: red; }
a:visited { color: green; }
a:hover { color: blue; }
a:active  { color: white; }
```

link, *active*, *visited*, *hover* – эти псевдоклассы используются только с элементом *<a>* и определяют стиль отображения обычной, активной, посещенной ссылок, а также ссылки, над которой находится курсор мыши. Стиль *a:hover* должен располагаться после стилей *a:link* и *a:visited*, иначе правила каскадирования скроют свойство *"color"* стиля *a:hover*.

Аналогично, благодаря тому, что *a:active* определен после *a:hover*, активная ссылка отображается белым цветом, когда пользователь устанавливает указатель поверх элемента <*a*> и одновременно активизирует его.

Приоритетность стилей определяется по нескольким классификациям.

1. По источнику, в порядке возрастания приоритета:
   a. Внешняя таблица стилей;
   b. Таблица встроенных стилей;
   c. Inline-стили;
2. По виду селектора, в порядке возрастания приоритета:
   a. Селекторы применяющиеся ко всем элементам с заданным именем;
   b. Контекстные селекторы, псевдоклассы;
   c. Классы, ID-классы;
3. По контексту: Стили применяющиеся к узкому контексту приоритетнее стилей применяющихся к широкому контексту;
4. По порядку вхождения: при прочих равных приоритетнее правило объявленное позже.

Некоторые свойства стилей и их описания:

color – определяет цвет текста элемента (например: *color: white* – устанавливает белый цвет, *color: #FFFFFF* – устанавливает белый цвет, запись цвета в шестнадцатиричном виде);

background-color – задает цвет фона элемента (например: *background-color: yellow* – задает желтый цвет фона);

font-family – устанавливает семейство шрифта элемента. Список шрифтов может иметь одно или несколько имен шрифтов, разделенных через запятую. Если на компьютере отсутствует первый шрифт, будет взят следующий и т.д. (например: *font-family: Arial, Helvetica, sans-serif*);

font-size – позволяет задать размер шрифта текста (например: *font-size: 16px* – размер шрифта 16 пикселей, *font-size: 9pt* – размер шрифта 9 пунктов, *font-size: 10mm* – размер шрифта 10 миллиметров) ;

font-style – определяет начертание шрифта, может принимать значения: *normal* – обычное начертание, *italic* – курсив, *oblique* – наклонный шрифт;

text-decoration – задает оформление текста, в виде мигающего текста, подчеркиваний, перечеркиваний. Можно задавать несколько значений через пробел. Значения: *blink* – мигающий текст, *line-through* – перечеркнутый текст, *overline* – линия над текстом, *underline* – линия под текстом, *none* – отменяет все эффекты;

text-align – определяет выравнивание текста элемента по горизонтали, может принимать значения: *left* – по левому краю, *right* – по правому краю, *center* – по центру, *justify* – выравнивание по ширине;

border – устанавливает одновременно цвет, толщину и стиль рамки вокруг элемента. Записывается в виде: *border: border-width border-style color*. Где *border-width* – ширина рамки, *border-style* – стиль рамки (значения: *dotted*, *dashed*, *solid*, *double*, *groove*, *ridge*, *inset*, *outset*), *color* – цвет. Например *border: 3px solid red* задает красную рамку, нарисованную сплошной линией толщиной в 3 пикселя.

Список свойств стилей с описанием можно посмотреть по ссылке: http://www.htmlbook.ru/css/

2 ПРОГРАММА РАБОТЫ

1. Создайте с помощью текстового редактора (например notepad) на локальном диске, в той же папке что и файл первого задания, файл с именем *styles.css*.
2. Добавьте в файл стилей четыре псевдокласса с разными цветами для элемента <*a*>. Сохраните файл.
3. Отредактируйте файл первого задания и пропишите в нем ссылку на внешнюю таблицу стилей styles.css Сохраните файл.
4. Запустите веб браузер, откройте созданный документ. Проверьте как меняется цвет ссылки, в зависимости от различных событий(Активная ссылка не должна изменять свой цвет при наведении на неё курсора мыши). Измените файл стилей таким

образом, чтобы активная ссылка изменяла свой цвет при наведении на неё курсора мыши. Результат покажите преподавателю.

5. Отредактируйте документ и таблицу стилей и измените стиль списка в соответствии с вариантом (стиль списка создайте с помощью селектора применяющегося ко всем элементам с заданным именем, стиль элементов списка - с помощью контекстного селектора, оба стиля должны быть встроенными. Стиль одного элемента списка объявите Inline). Результат покажите преподавателю. Поясните результат.

6. Отредактируйте документ и таблицу стилей и измените цвета таблицы в соответствии с вариантом. Стили объявите как классы и поместите во внешнюю таблицу стилей. Результат покажите преподавателю.

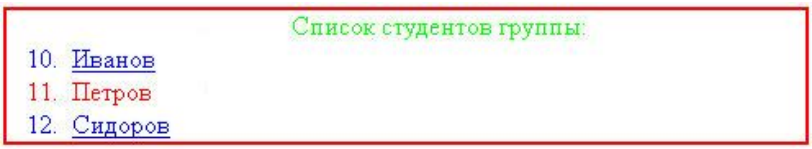
3 ВАРИАНТЫ РАБОТЫ

1. Список №1, таблица №1
2. Список №1, таблица №2
3. Список №2, таблица №1
4. Список №2, таблица №2
5. Список №3, таблица №1
6. Список №3, таблица №2

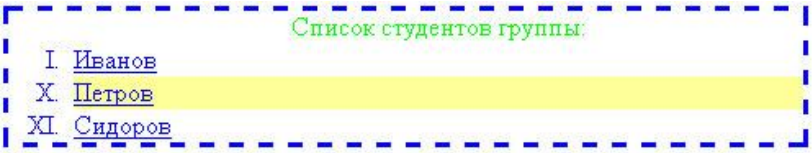
Варианты списков:

1.

2.



3.



Варианты таблиц:

1.

Расписание занятий

Время\№ группы		360-1	360-2	360-3	360-4
Понедельник	8:50-10:25	ММиП ЛР 320Ф	Физика ЛР 230Ф	ММиП ЛР 301Ф	Физика ЛР 228Ф
	10:40-12:15	ММиП ЛР 320Ф		ММиП ЛР 320Ф	
	13:15-14:50	ММиП лекция 311Ф			
	15:00-16:35	Высшая математика лекция 204Ф			

2.

Расписание занятий

Время\№ группы		360-1	360-2	360-3	360-4
Вторник	8:50-10:25	ММиП ЛР 320Ф	Физика ЛР 230Ф	ММиП ЛР 301Ф	Физика ЛР 228Ф
	10:40-12:15		Физика ЛР 230Ф		Физика ЛР 228Ф
	13:15-14:50	ММиП лекция 311Ф			
	15:00-16:35	Высшая математика лекция 204Ф			

ЛАБОРАТОРНАЯ РАБОТА № 3
Изучение DOM

1 ВВЕДЕНИЕ

Целью работы является знакомство с объектной моделью документа (Document Object Model – DOM).

Объектная модель документов – это независящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому документов, а также изменять содержимое, структуру и оформление документов.

Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый узел которого представляет собой элемент, атрибут, текстовый, графический или любой другой объект. Узлы связаны между собой отношениями родительский-дочерний.

Изначально различные браузеры имели собственные модели документов (DOM), не совместимые с остальными. Для того, чтобы обеспечить взаимную и обратную совместимость, специалисты международного консорциума W3C классифицировали эту модель по уровням, для каждого из которых была создана своя спецификация. Все эти спецификации объединены в общую группу, носящую название W3C DOM

2 ОПИСАНИЕ DOM

Существует 4 уровня объектной модели:

Уровень 0: Описывает объектные модели документов различных браузеров, существовавшие до появления Уровня 1. Фактически является только информацией о том, что существовало до появления спецификации

Уровень 1: Описывает базовые функции DOM, для получения дерева узлов документа, возможность добавлять и изменять данные.

Уровень 2: Помимо базовых функций, введено понятие пространства имен XML (XML namespaces), имеется поддержка стилей и обработчиков событий

Уровень 3: Описывает набор расширений относительно уровня 2

Текущим поддерживаемом уровнем спецификаций, является уровень 2. Справочник по API можно посмотреть по ссылке: http://experiment.net.ru/dom/

DOM представляет документ, как иерархию узлов, существуют узлы таких типов:

1. Text (текстовое содержание внутри тегов). Например: "Курсивный текст"
2. CDATASection (Character data –символьные данные). Например: "<![CDATA[строка]]>"
3. Element (элемент дерева). Например: "<a>…"
4. Comment (комментарий). Например: "<!-- Этот текст является комментарием --!>"
5. Attr (аттрибут). Например: "Сайт ТУСУР"
6. DocumentType (тип документа). Например: "<!DOCTYPE HTML PUBLIC>"
7. Notation (нотация документа определенная в DTD).

Например документ такого содержения:

```
<html>
  <head>
    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
  </body>
</html>
```

Будет выглядеть как на рисунке 1.

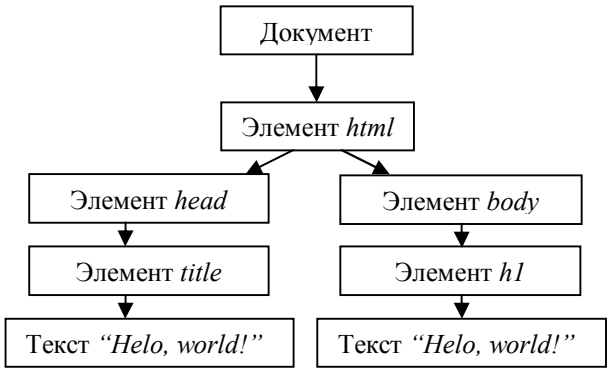


Рисунок 1. Дерево объектов DOM

Для браузера Mozilla Firefox существует специальный плагин (DOM Inspector), позволяющий посмотреть объектную модель открытого документа.

Пример документа, динамически создающего элементы неупорядоченного списка (по щелчку левой кнопки мыши на название списка):

```
<html>
<head>
  <title>Динамическое создание списка</title>
  <script language="JavaScript" type="text/javascript">
   function clickfunc()
     { // начало функции
// Инициализируется переменная node - для узла элемента
 var node;
// В переменную node присваивается элемент
// документа, имеющий строку dynamic в атрибуте id
node = document.getElementById("dynamic");
        if(node) // Если такой элемент был найден
 { // начало условия
   var attr; // Объявляется переменная attr
// Создается новые атрбут с именем type
// и присваивается в переменную attr
   attr = document.createAttribute("type");
// устанавливается значение атрибута равное строке square
   attr.value = "square";
```

```
// У элемента, присвоеного в переменную node
// устанавливается атрибут (имя и значение)
// из переменной attr
  node.setAttributeNode(attr);
  var element; // Объявляется переменная attr
// Создается новый элемент с тэгом
// li(элемент списка) и присваивается в переменную element
  element = document.createElement("li");
// Устанавливается содержимое
// элемента li, присвоенного в переменную element
// (Работает для FireFox и Opera)
  element.textContent = "Элемент списка";
  if (navigator.appName == "Microsoft Internet Explorer")
  {
// Если браузер - интернет эксплорер, то
// другим способом устанавливается содержимое элемента li
    element.innerText = "Элемент списка";
  }
// Добавить элемент li в элемент, присвоенный в
//  переменную node
  node.appendChild(element);
} // конец условия
     } // конец функции
   </script>
</head>

<body>
<ul id="dynamic" onClick="clickfunc()"> Список предметов и работ:
</ul>

</body>
</html>
```

2 ПРОГРАММА РАБОТЫ

1. Создайте с помощью javascript (по щелчку) список элементов, идентичный созданному в первой лабораторной работе, согласно вашего варианта (не менее 10 элементов).

Лабораторная работа № 4
Изучение языка JavaScript

1 ВВЕДЕНИЕ

Целью работы является знакомство с базовыми принципами создания программ с использованием объектно-ориентированного подхода на примере языка JavaScript.

JavaScript — скриптовый язык, чаще всего использующийся при создании сценариев поведения браузера, встраиваемых в веб-страницы. Является одной из реализаций языка ECMAScript.

2 ОПИСАНИЕ ЯЗЫКА JAVASCRIPT

Для встраивания сценария JavaScript в документ HTML используется специальный элемент, прописанный внутри элемента *head*:

```
<html>
  <head>
    <script type="text/javascript">
      alert('Hello World!');
    </script>
  </head>
  <body>
    Hello World script
  </body>
</html>
```

Все, что находится внутри тегов <*script*> и </*script*> является сценарием JavaScript.

3 СИНТАКСИС ЯЗЫКА JAVASCRIPT

Программа, написанная на JavaScript, состоит из набора операторов. Оператор (или инструкция) – это наименьшая автономная часть языка программирования, команда. Каждый оператор должен заканчиваться символом точка с запятой (;). Типы данных языка: Null (пустое тип, может принимать только

значение *null*), Boolean (булевый тип данных, принимает значения *true*(истина), *false*(ложь)), Number (числовой тип), String (строковый тип). Строковые константы заключаются в кавычки, либо апострофы (например: *“Это строка”*, *'Это строка'*, *“Это строка содержит внутри ‘апострофы’ ”*, *‘Эта строка содержит внутри “кавычки”’*).

Операторы JavaScript:

Блок операторов: *{ operator1; operator2; ... }*

Объявление переменной: *var i;*

Объявление массива: *var arr = new Array();*

Объявление функции: *function myfunc(param1, param2) { }*

Оператор присвоения: *i = 10; s = ‘Hello’;*

Доступ к элементу массива: *s = arr[i];*

Доступ к свойствам и методам объектов: *n = arr.length();* Записывает в *n* результат вызова метода определения длинны массива.

arr. *document.getElementById(“dynamic”);* вызывает метод *getElementbyId* объекта *document* со строковым параметром.

Математические операторы:

Сложение: *c = a+b;* или *a=a+b;* эквивалентно *a+=b;*

Вычитание: *c = a-b;* или *a=a-b;* эквивалентно *a-=b;*

Умножение: *c = a*b;* или *a=a*b;* эквивалентно *a*=b;*

Деление: *c = a/b;* или *a=a/b;* эквивалентно *a/=b;*

Остаток от деления: *c = a%b;* или *a=a%b;* эквивалентно *a%=b;*

Инкремент: *c++;* эквивалентно *c=c+1;* эквивалентно *c+=1;*

Декремент: *c--;* эквивалентно *c=c-1;* эквивалентно *c-=1;*

Условный оператор:

```
if(условие) {
 блок операторов, если условие истинно
}
else {
блок операторов, если условие ложно
}
```

Например: *if(a < b) { a = b;} else {b = a;}* Если значение *a* меньше значения *b*, то переменной *a* присвоится значение переменной *b*. Иначе, переменной b присвоится значение переменной *a*.

Переключатель:

```
switch (a) {
  case 1: func1();
     break;
  case 2: func2();
     break;
  default: funcdefault();
     break;
}
```

В зависимости от значение переменной *a*, вызывается соответствующий блок операторов, или блок операторов по-умолчанию (*default*)

Операторы цикла:

Цикл «пока» (условие проверяется перед выполнением тела цикла):

```
while (условие) {
блок операторов выполняется, пока выполняется условие
}
```

Цикл «пока» с постусловием (условие проверяется после выполнения тела цикла):

```
do {
блок операторов выполняется, пока выполняется условие
} while (условие);
```

Цикл for:

for (начальные условия; условие выполнения; итерационные операции) {

блок операторов выполняется, пока выполняется условие. После каждого вызова блока операций, вызываются итерационные операции

}

Например: *for(i = 0; i < 10; i++) { alert(string[i]); }* Перед выполнением цикла обнулится переменная *i*. До тех пор, пока значение переменной *i* меньше 10, будет выполнятся вызов функции *alert(sring[i])* с текущим значением *i*. И после вызова, значение *I* будет инкрементироваться.

Операторы сравнения:

Оператор меньше: *a < b* возвращает истину, если *a* меньше чем *b*

Оператор больше: *a > b* возвращает истину, если *a* больше чем *b*

Оператор равно: *a == b* возвращает истину, если *a* равно *b*

Оператор неравно: *a != b* возвращает истину, если *a* не равно *b*

Оператор меньше или равно: *a <= b* возвращает истину, если *a* меньше или равно *b*

Оператор больше или равно : *a >= b* возвращает истину, если *a* больше или равно *b*

Логические операторы:

Оператор «не»: *! a* возвращает истину, если *a* ложно, возвращает ложь, если *a* истинно

Логическое «И»: *a&&b* возвращает истину, если *a* и *b* истинны, иначе возвращает ложь

Логическое «ИЛИ»: *a||b* возвращает истину, если хотя бы одно из *a* или *b* истинно, иначе возвращает ложь

Логическое «Исключающее ИЛИ»: *a^^b* возвращает истину, если истинно только одно из значений *a* или *b*, иначе возвращает ложь

Побитовые операторы:

Побитовое «И»: *c = a&b;*

Побитовое «ИЛИ»: *c = a|b;*

Побитовое «Исключающее ИЛИ»: *c = a^b;*

JavaScript является регистрозависимым языком, т.е. все операторы, все вызовы функций, использование переменных, должны быть записаны с учетом регистра.

Пример скрипта, осуществляющего сложение двух чисел, взятых из текстовых полей формы, с выводом результата в третье поле:

```
<html>
<head>
 <title>Математическое моделирование и программирование. Лабораторная №4</title>
```

```
 <meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
 <script language="JavaScript" type="text/javascript">
  function Calculate()
      { // начало функции
       var p1; // Инициализируется переменная p1 - для узла элемента
         p1 = document.getElementById("p1"); // В переменную p1 присваивается элемент
       // документа, имеющий строку p1 в атрибуте id
       var p2; // Инициализируется переменная p2 - для узла элемента
         p2 = document.getElementById("p2"); // В переменную p2 присваивается элемент
       // документа, имеющий строку p2 в атрибуте id
       var res; // Инициализируется переменная res - для узла элемента
       res = document.getElementById("res"); // В переменную p2 присваивается элемент
       // документа, имеющий строку res в атрибуте id
       var r; // Инициализируется переменная для помещения результата
       r = 1*p1.value + 1*p2.value; // с помощью умножения на единицу переводятся
       // значения текстовых полей из строковых в числовые, после этого складываются
       res.value =  r //результат помещается в текстовое поле res
       if(r == 0) { // Если результат равен 0
        alert('Результат расчета равен 0'); // Вывести сообщение
       } // конец условия
      } // конец функции
 </script>
</head>
<body>
<form name="calc">
 <input name="p1" type="text" id="p1" size="10">
 +
```

```
<input name="p2" type="text" id="p2" size="10">
=
<input name="res" type="text" id="res" size="10">
<input type="button" name="Button" value="Посчитать" onClick="Calculate();">
</form></body>
</html>
```

4. ПРОГРАММА РАБОТЫ

1. Создайте с помощью текстового редактора (например notepad) на локальном диске, html документ и поместите в него пример.
2. Откройте получившийся документ в браузере и убедитесь, что все работает нормально
3. Отредактируйте документ, для выполнения вычислений по формуле, в соответствии с вариантом задания. Результат покажите преподавателю

5. ВАРИАНТЫ РАБОТЫ

1. Входные данные: a, b, c, d. Выходные данные: r. Формула: r = (a+b) * (b * c + d * a), выводить сообщение, если результат больше либо равен 100;
2. Входные данные: a, b, c, d. Выходные данные: r. Формула: r = (a*a - 4*b*b) * d*c, выводить сообщение, если результат отрицательный;
3. Входные данные: a, b, c, d. Выходные данные: r. Формула: r = (4*b - a*c + d*d) * b, выводить сообщение, если результат не больше нуля;
4. Входные данные: a, b, c, d. Выходные данные: r. Формула: r = (a*b + c*d) / (b+d), выводить сообщение, если знаменатель равен 0;
5. Входные данные: a, b, c, d. Выходные данные: r. Формула: r = ((a - c)*(a+c)) / (b+d), выводить сообщение, если знаменатель равен 0;

6. Входные данные: a, b, c, d. Выходные данные: r. Формула: r = (b*b – 4*a*c)*(d+a*b) выводить сообщение, если знаменатель равен (b*b – 4*a*c) равно 0;

ЛАБОРАТОРНАЯ РАБОТА № 5
Автоматизация формирования математических моделей электронных схем

1. ВВЕДЕНИЕ

Целью работы является знакомство с принципами получения матриц топологических уравнений.

2. ОПИСАНИЕ РАБОТЫ

Операции получения структурной матрицы просты и рассматриваются на лекции.

При использовании матричных технологий токи, напряжения, параметры, упаковываются в вектора – одномерные массивы, и матрицы – двумерные массивы, каждому элементу при этом соответствует определенное место в таких массивах.

Поэтому сначала важно выбрать порядок следования элементов (отсортировав их, сначала по типам), который далее будет оставаться постоянным. При этом важно помнить, что нумерация элементов в массивах маткада начинается с нуля или, говоря иначе, первый элемент массива имеет индекс 0.

Выберем следующий порядок следования элементов в схеме

R,L,C,V, J, I

Где R-резистивные элементы, L-индуктиваности, C-емкости, V-вольтметры источников тока управляемых напряжением (ИТУН); J-источники тока, управляемые напряжением, I- независимые источники токов.

При этом вектор токов будет иметь вид -

$I_B = [I_R,I_L,I_C,I_V, J, I]^T$,

а вектор всех напряжений -

$U_B = [U_R,U_L,U_C,V, U_J,U_I]^T$

В матричном виде уравнение по 1-му закону Кирхгофа можно записать с использованием структурной матрицы следующим образом:

$A_{стр} * I_B = 0$

Схематично матрицу Астр, с учетом принятого деления ветвей, можно изобразить как показано в таблице ниже

	Ветви					
	I_R	I_L	I_C	I_V	J	I
Узлы	A_R	A_L	A_C	A_V	A_J	A_I

где матрице $A_{стр}$ соотвествует часть таблицы, выделенная бирюзовым цветом.

Также 1-й закон Кирхгофа можно записать в виде

$$A_R * I_R + A_L * I_L + A_C * I_C + A_V * I_V + A_J * J + A_I * I = 0.$$

$$A_{стр} = [A_R \mid A_L \mid A_C \mid A_V \mid A_J \mid A_I]$$

Матицы Ai – состоят из элеменов -1, 0 или +1. Каждый столбец содержит -1 ставится в клетке соответствующей у

3. ПРОГРАММА РАБОТЫ

1. Для заданной схемы составьте (на листе бумаги) направленный граф, обозначьте ветви и пронумеруйте узлы. При необходимости замените многовыводные компоненты двух выводными.
2. Запишите нет-лист (на листе бумаги).
3. Отсортируйте элементы в следующем порядке R,L,C, V, J, I. Порядок сортировки V элементом должен быть таким же как и J-элементов, это позволит использовать компонентное уравнение для J вида J=K*V, где K – диагональная матрица коэффицентов преобразования (напряжений вольтметров в токи зависимых источников J).

4. Запустите Маткад. Запишите нет-лист в форме матриц в программе MatCad

Как показано ниже для элементов R-типа

назв. Элемента	Положит. Узел (от которого отходит стрелка на графе)	Отриц. узел, в который направлена стрелка графа
R2	1	0
R3	3	6
R4	7	2

Обозначим полученную матрицу символом NL_R (Net-List for R-)

5. Запишите аналогичные матрицы для всех остальных элементов

6. Составьте функцию, создающую части структурной матрицы

A_R = makeA(nodes,NL_R)

где nodes – общее число узлов в схеме.

Получите все структурные подматрицы схемы - A_R | A_L | A_C | A_V | A_J | A_I

7. Составьте функцию, создающую одностолбцовые матрицы значений компонентов Dx-

D_R = makeD(NL_R)

где $D_R = [R1,R2,\ldots,RN]^T$

8.Получите все одностолбцовые матрицы - D_R D_L D_C D_K

4. ВАРИАНТЫ СХЕМ

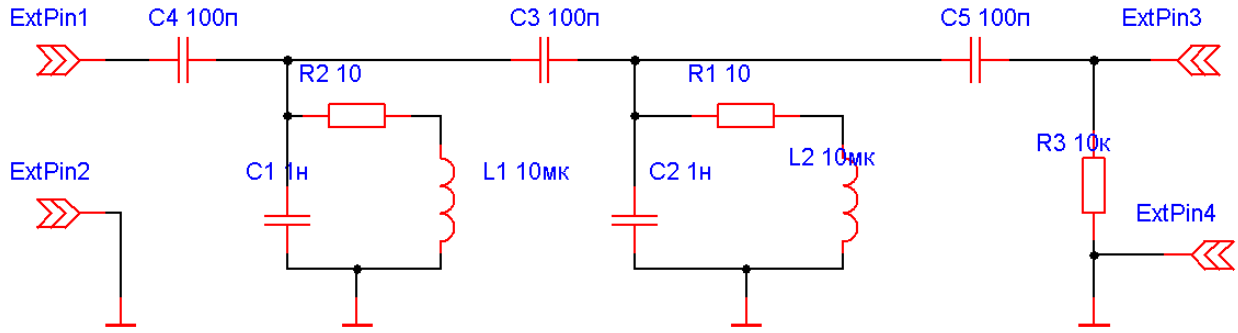


Схема 1

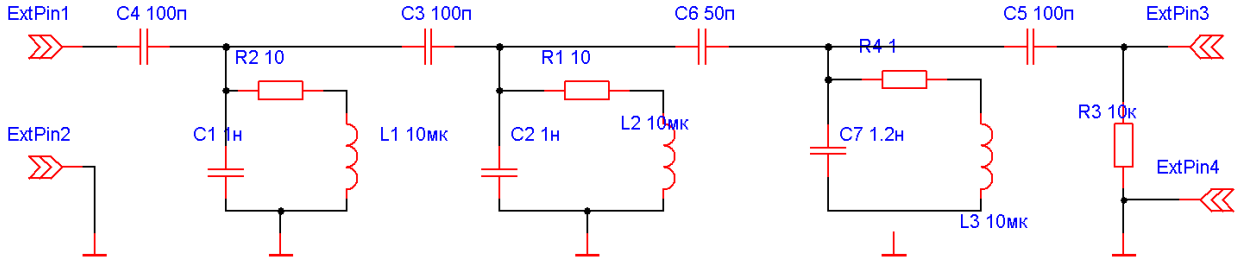


Схема 2

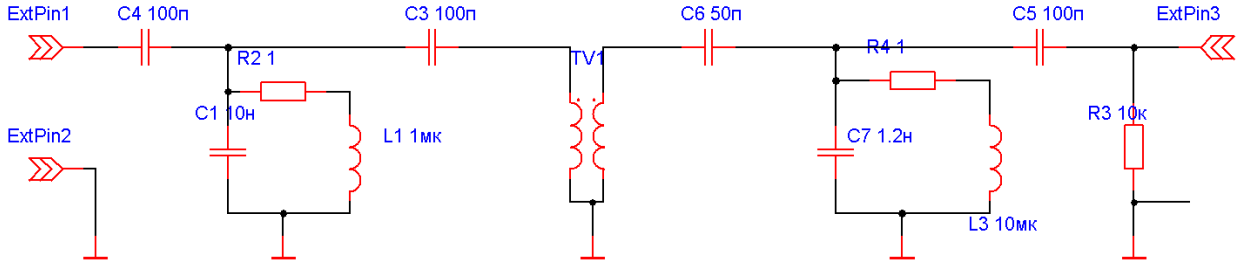


Схема 3

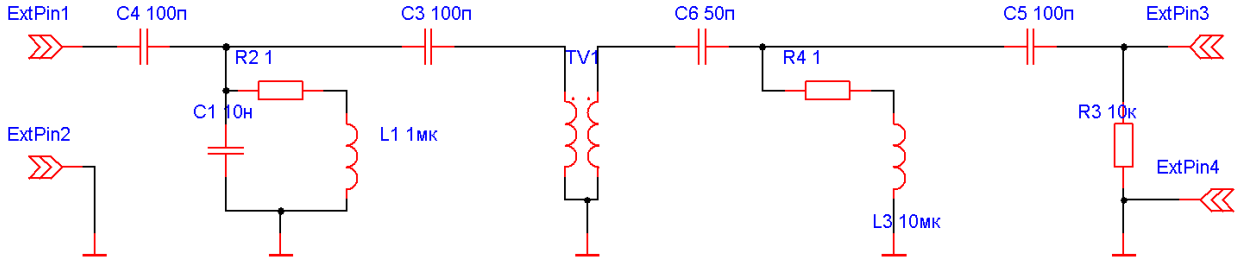


Схема 4

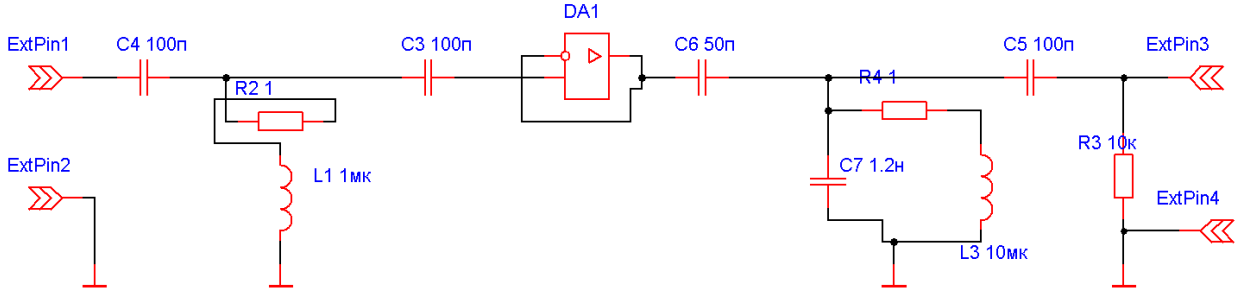


Схема 5

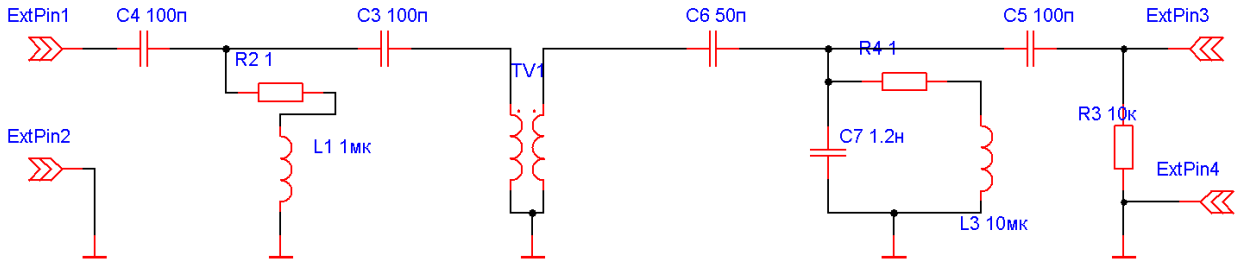


Схема 6

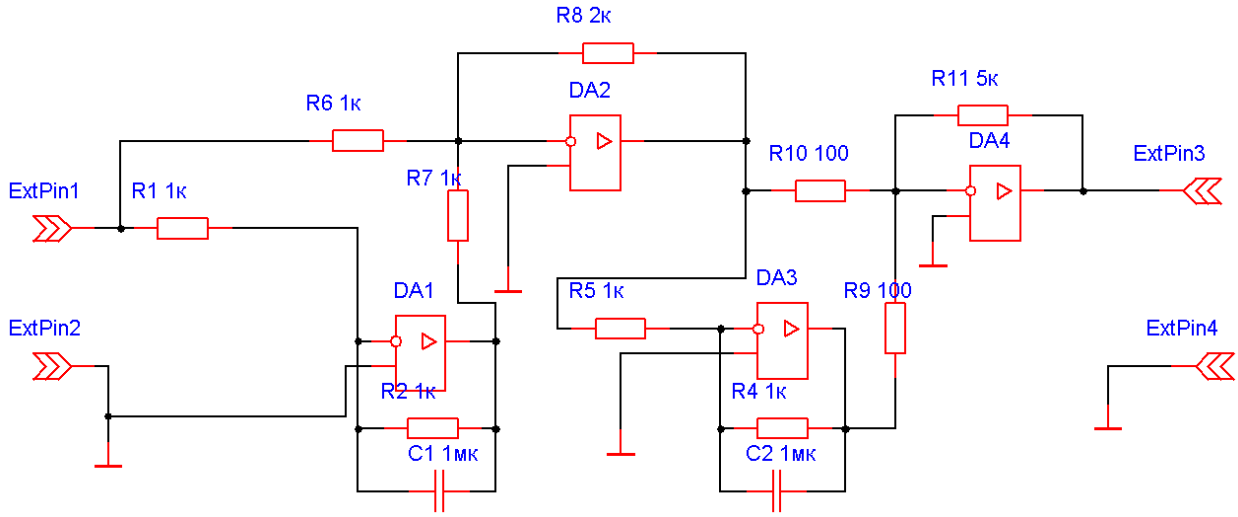


Схема 7

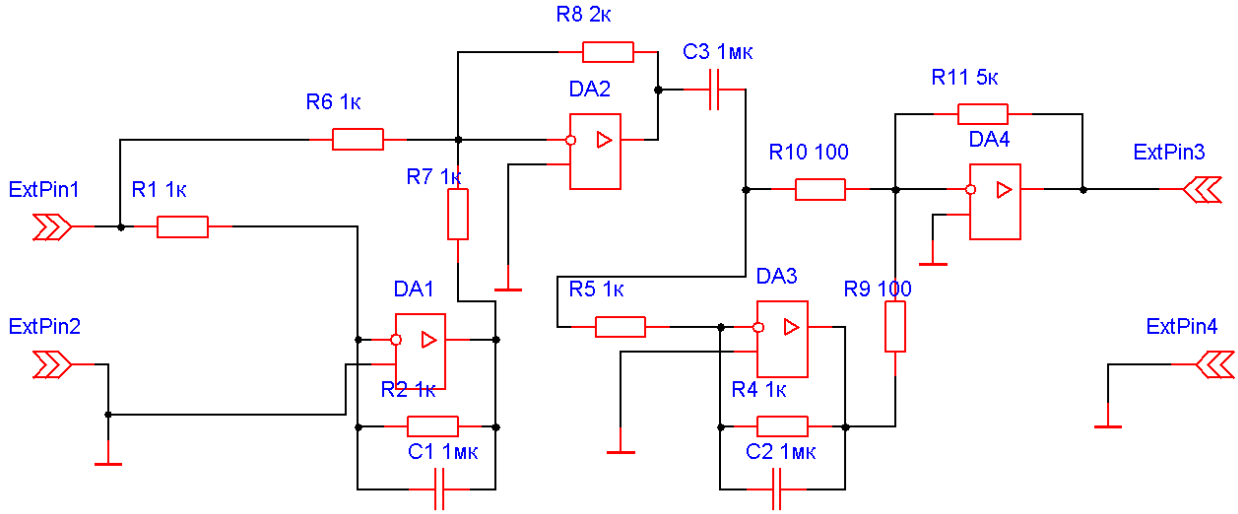


Схема 8

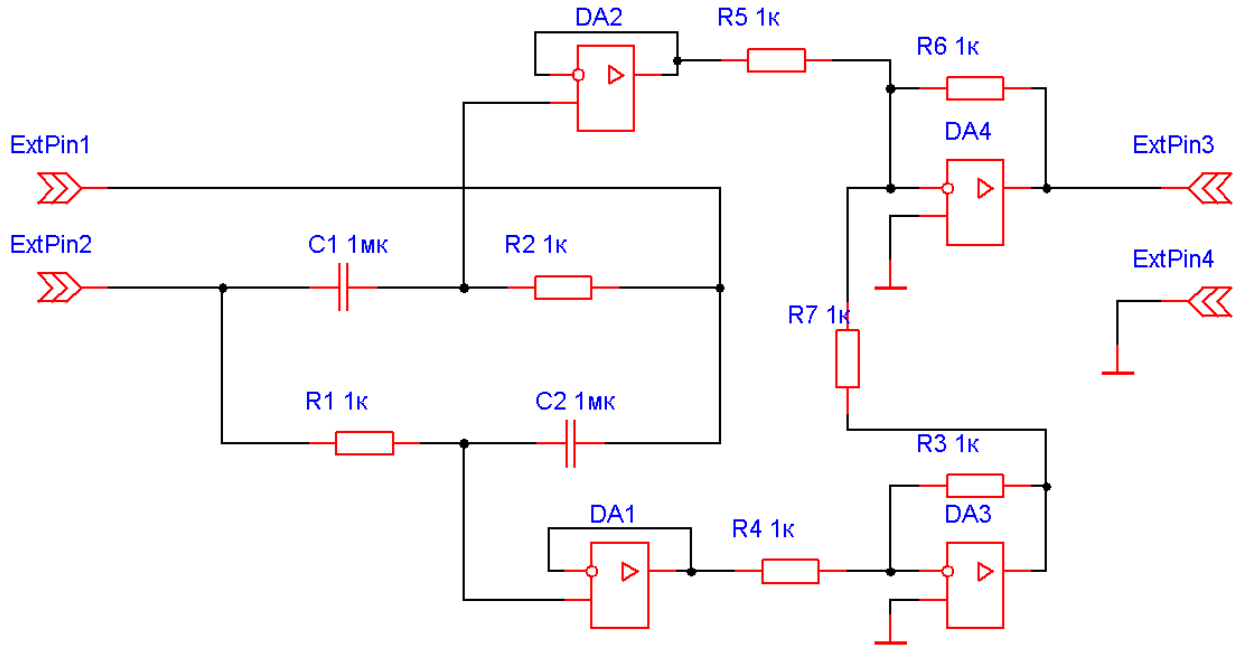


Схема 9

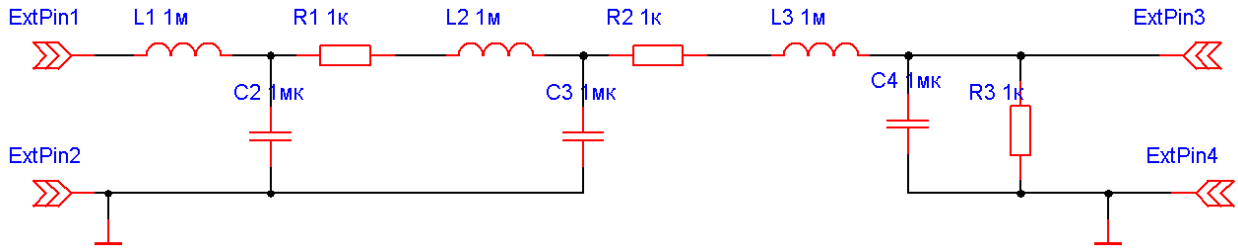


Схема 10

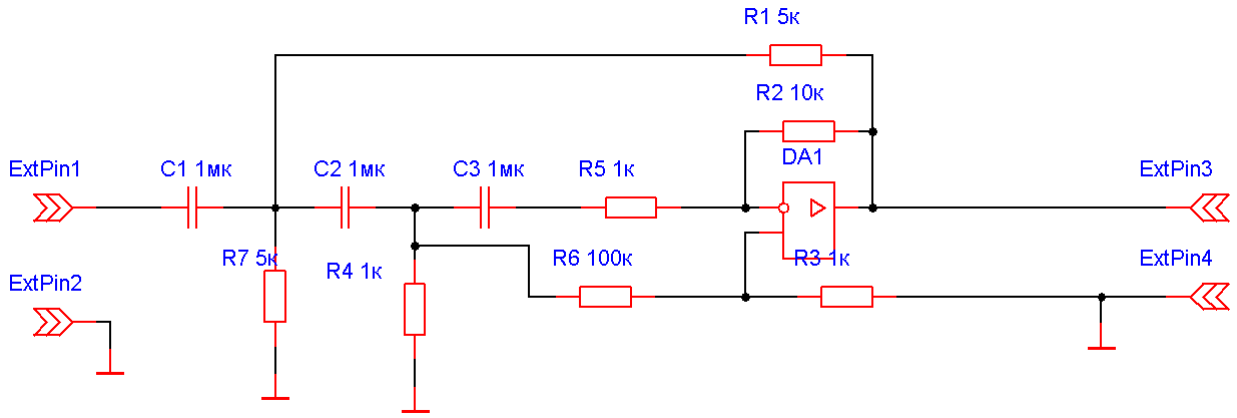
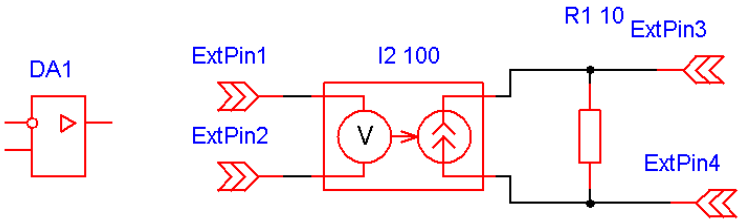


Схема 11

ПРИЛОЖЕНИЕ №1

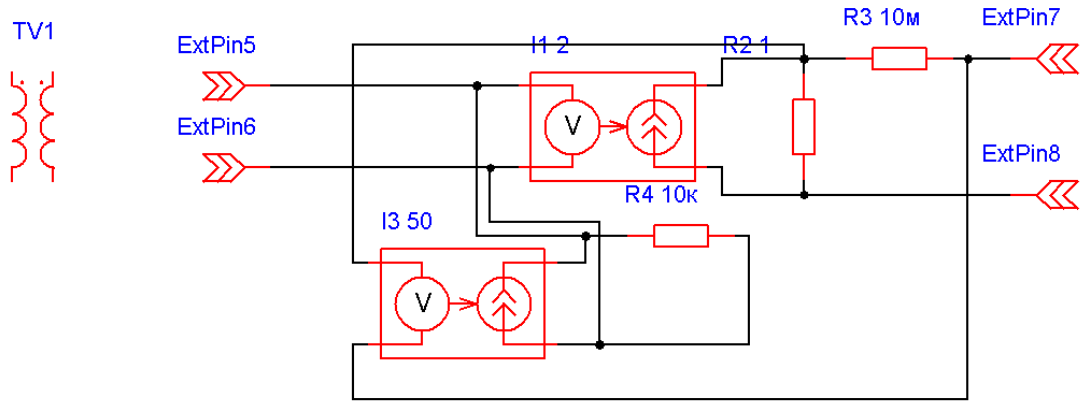
Схемы замещения многовыводных компонентов

Операционный усилитель замещается тремя ветвями, как показано на рисунке



Обратите внимание! Обычно вывод «земля» на выходе усилителях не обозначается, но физически, конечно, присутствует. Поэтому ExtPin4 заземляется. Вход усилителя, помеченный кружочком, соответствует ExtPin2, без кружочка - ExtPin1. Выходу соответствует ExtPin3.

Идеальный трансформатор с коэффициентом передачи 2, ко вторичной обмотке замещается 7-ю ветвями



СТАЦИОНАРНЫЕ СОСТОЯНИЯ НА ПЕРЕМЕННОМ СИНУСОИДАЛЬНОМ ТОКЕ. ПОСТРОЕНИЕ ЧАСТОТНЫХ ХАРАКТЕРИСТИК

Для построения частотных характеристик применим метод узловых потенциалов. Для нахождения узловых потенциалов будем решать уравнение -

$$Y*\phi=Y_I \qquad (1)$$

здесь

$$Y = Ag*G*Ag^T + A_J*K*A_V^T,$$

$$Y_I = -A_I*I;$$

структурная матрица ветвей – проводимостей Ag

$$Ag = [A_R|A_L|A_C]$$

может быть собрана из соответствующих структурных подматриц в Маткад-е методом
Ag =**augment(**$A_R|A_L|A_C$**)**;
матрица G – диагональная, она состоит из комплексных проводимостей резестивных, индуктивных и емкостных ветвей и зависит от частоты

$$\omega = 2 * \pi * f$$

$$G(\omega) = \begin{vmatrix} 1/R & 0 & 0 \\ 0 & 1/(j*\omega*L) & 0 \\ 0 & 0 & j*\omega*C \end{vmatrix}$$

Используйте матричные функции stack и diag для получения матриц G и K

$$G(\omega)=diag(stack(DR,j*w*DL,1/(j*w*DC))^{-1}$$

$$K= diag(DK)$$

Решив уравнение (1) мы найдем комплексные узловые потенциалы.
Из полученного вектора потенциалов возьмите тот потенциал, который соответствует выходу вашей схемы. Найдите модуль - Am и фазу - ϕ выходного узлового потенциала.

Постройте графики амплитуды и фазы выходного сигнала (при единичном входном), используйте логарифмические оси по частоте и по амплитуде. Такие графики называются «Амплитудно-частотная характеристика» (АЧХ) и Фазо-частотная характеристика (ФЧХ) схемы соответственно. Выведите амплитуду в децибелах – y=20*lg(x), здесь x –входной сигнал, y-значение в децибелах.

Соберите и сохраните схему в программе ASIMEC, постройте с ее помощью амплитудную и частотные характеристики, используя плоттер Боде. Для демонстрации его работы запустите **c6_tac.cir** из каталога DEMO программы.

Добейтесь совпадения полученных с помощью MatCad и ASIMEC характеристик.

ЗАДАЧА КОШИ. ПОСТРОЕНИЕ ГРАФИКА ПЕРЕХОДНОГО ПРОЦЕССА

Задача Коши предполагает нахождение частного решения исходной системы уравнений – нахождение функции $x(t)$ - для одной или нескольких выходных переменных при заданных начальных условиях $x(0)$.

Для нахождения переходного процесса будем использовать метод численного интегрирования с помощью **неявной схемы Эйлера –**

$$X=X_p+h*dX/dt$$

Здесь X – вектор состояния искомого решения в момент времени $t=t_p+h$, $X_p=X(t_p)$ – значение вектора состояния в момент t_p, соответствующий предшествующему состоянию.
Индекс p – означает previous (предшествующий).
Реализация схемы следующая – при известном X_p в момент времени t_p – находим X, соответствующий времени $t=t_p+h$, при этом вектор X и вектор его производных по времени dX/dt связанны между собой посредствам нашей матричной системы уравнений
для токов

$$A_R* I_R + A_L* I_L + A_C * I_C + A_V * I_V + A_J * J + A_I * I = 0;$$

напряжений

$$U_R = A_R^T*\phi,\ U_L = A_L^T*\phi,\ U_C = A_C^T*\phi,\ U_V = A_V^T*\phi;$$

и компонентных уравнений

$$U_R=R*I_R,\ J=K*V,\ L*dI_L/dt=U_L,\ C*dU_C/dt=I_C$$

Два последних компонентных уравнения являются дифференциальными, следовательно -
$dX/dt = [dU_C/dt, dI_L/dt]^T$, a $X = [U_C, I_L]^T$.

Дальнейшие выражения для расчета будем получать простой подстановкой одних уравнений в другие, исключая переменные, тем самым упрощая систему до удобного вида. Сначала подставим выражение для неявной схемы Эйлера в компонентные уравнения для емкостей

$$I_C=C/h*(U_C-U_{Cp}), \qquad (1^*)$$

затем для индуктивностей

$$I_L =I_{Lp} + h/L*U_L. \qquad (2^*)$$

I_R – выразим через узловые потенциалы при помощи первого уравнения для напряжений и первого компонентного уравнения –

$$I_R = R^{-1}*A_R^T*\phi$$

Результат подставим в уравнение для токов -

$$A_R * R^{-1} * A_R^T * \phi + A_L * I_L + A_C * I_C + A_J * J + A_I * I = 0$$

Этим действием мы исключили из дальнейшего рассмотрения переменные I_R и U_R.
Действуя по аналогии для зависимых источников J –

$$A_R * R^{-1} * A_R^T * \phi + A_L * I_L + A_C * I_C + A_J * K * A_V^T * \phi + A_I * I = 0$$

Для емкостей, подставив уравнение для напряжений $U_C = A_C^T * \phi$ в (1^*) получим

$I_C = C/h * (A_C^T * \phi - U_{Cp})$ и

$$A_R * R^{-1} * A_R^T * \phi + A_L * I_L + A_C * C/h * A_C^T * \phi - A_C * C/h * U_{Cp} + A_J * K * A_V^T * \phi + A_I * I = 0$$

Заменим в (2^*) $U_L = A_L^T * \phi$ и подставим в полученное уравнение для токов -

$$A_R * R^{-1} * A_R^T * \phi + A_L * I_{Lp} + A_L * h/L * A_L^T * \phi + A_C * C/h * A_C^T * \phi - A_C * C/h * U_{Cp} + A_J * K * A_V^T * \phi + A_I * I = 0$$

Неизвестными в полученном выражении будут только узловые потенциалы. Введем обозначения -

$$Y_R = A_R * R^{-1} * A_R^T, \; Y_L = A_L * h/L * A_L^T, \; Y_C = A_C * C/h * A_C^T, \; Y_J = A_J * K * A_V^T$$

Тогда

$$[Y_R + Y_C + Y_L + Y_J] * \phi = -A_L * I_{Lp} + A_C * C/h * U_{Cp} - A_I * I$$

С точки зрения вычислительных затрат выгодно создать матрицу

$$Y_{inv} = [Y_R + Y_C + Y_L + Y_J]^{-1}$$

поскольку операция обращения самая трудоемкая, здесь она выполняется 1 раз.
тогда узловые потенциалы –

$$\phi = Y_{inv} [-A_L * I_{Lp} + A_C * C/h * U_{Cp} - A_I * I] \qquad (3^*)$$

Для построения графиков переходных процессов нам понадобятся следующие функции –

$$\phi = \text{calcFi}(I_{Lp}, U_{Cp})$$

реализующая (3^*),

$$U_C = \text{calcUc}(\phi),$$

реализующая, соответствующее (третье) уравнение для напряжений, и

$$I_L = \text{calcUL}(\phi, I_{Lp}),$$

реализующая $I_L = I_{Lp} + h/L * U_L$, где $U_L = A_L^T * \phi$.

Задание – постройте график переходного процесса из нулевых начальных условий

$I_{Lp}(t=0)=0$ и $U_{Cp}(t=0)=0$

для разных значений шага интегрирования. Подберите шаг, формирующий достаточно гладкий график.
Постройте такой же график с помощью ASIMEC. Добейтесь совпадения результатов.

Отчет о проделанной работе.

Запакуйте в архив файл MatCad-а и файл программы ASIMEC, и скан графа схемы. При необходимости сопроводите фрагменты выполненной работы комментариями в произвольной форме.

Отправьте преподавателю архив электронной почтой !!! в теме письма поставьте номер группы, фамилию, инициалы и вариант задания.