Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования

Томский государственный университет систем управления и радиоэлектроники (ТУСУР)

Руководство по установке и эксплуатации программного обеспечения «Система компьютерного моделирования электромагнитной совместимости ТУСУР.ЭМС (TUSUR.EMC)»

Томск, 2024

АННОТАЦИЯ

В данном программном документе приведена инструкция по установке и эксплуатации программного обеспечения «Система компьютерного моделирования электромагнитной совместимости ТУСУР.ЭМС (TUSUR.EMC)» (далее система TUSUR.EMC).

В разделе «Назначение системы TUSUR.EMC» указаны сведения о назначении системы и информация, достаточная для понимания её функций и эксплуатации.

В разделе «Условия выполнения системы TUSUR.EMC» указаны условия, необходимые для работы системы TUSUR.EMC (минимальный состав аппаратных и программных средств и т.п.).

В разделе «Выполнение системы TUSUR.EMC» указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение системы TUSUR.EMC, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением системы, а также её ответы на эти команды. Приведено руководство по встроенному в систему интерпретируемому скриптовому языку TALGAT_Script и описаны особенности использования в системе интерпретатора языка Python.

В разделе «Модули системы TUSUR.EMC» приведены сведения необходимые для работы с модулями системы, а также примеры их использования.

В разделе «Сообщения оператору» приведены тексты сообщений, выдаваемых в ходе выполнения системы TUSUR.EMC, описание их содержания и соответствующие действия оператора (действия оператора в случае сбоя, возможности повторного запуска системы и т.п.).

Официальный сайт системы TUSUR.EMC – http://emc.tusur.ru/.

Оформление программного документа произведено по требованиям ЕСПД (ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов, ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов, ГОСТ 19.104-78 ЕСПД. Основные надписи, ГОСТ 19.105-78 ЕСПД. Общие требования к программным документам, ГОСТ 19.106-78 ЕСПД. Общие требования к программным документам, выполненным печатным способом, ГОСТ 19.604-78 ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом, ГОСТ 19.505-79 ЕСПД. Руководство оператора).

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ГА	Генетический алгоритм
МПЛП	Многопроводная линия передачи
СЛАУ	Система линейных алгебраических уравнений
ЭМС	Электромагнитная совместимость
ЭП	Эвристический поиск
ЭС	Эволюционные стратегии

СОДЕРЖАНИЕ

1 УСТАНОВКА СИСТЕМЫ TUSUR.EMC 5
2 НАЗНАЧЕНИЕ СИСТЕМЫ TUSUR.EMC 9
2.1 Цели и назначение системы 9
2.2 Структура системы 9
2.3 Выполняемые функции системы, состав, назначение и характеристики её комплексов 10
3 УСЛОВИЯ ВЫПОЛНЕНИЯ СИСТЕМЫ TUSUR.EMC 11
3.1 Минимальный состав технических средств 11
3.2 Минимальный состав программных средств 11
3.3 Требования к персоналу 11
4 ВЫПОЛНЕНИЕ СИСТЕМЫ TUSUR.EMC 12
4.1 Загрузка и запуск системы TUSUR.EMC 12
4.2 Описание функций системы TUSUR.EMC 12
4.2.1 Графический интерфейс пользователя 12
4.2.2 Скриптовый язык TALGAT_Script 16
4.2.3 Интерпретатор Python 23
5 МОДУЛИ СИСТЕМЫ TUSUR.EMC 24
5.1 Вычислительные модули 24
5.1.1 Модуль двумерного электростатического анализа MOM2D 24
5.1.2 Модуль трёхмерного электростатического анализа MOM3D 44
5.1.3 Модуль трёхмерного электродинамического анализа MOMW 53
5.1.4 Модуль вычисления временного и частотного откликов RESPONSE 66
5.2 Модули оптимизации 77
5.2.1 Модуль генетических алгоритмов GA 77
5.2.2 Модуль эволюционных стратегий ES 86
5.3 Модули утилит 88
5.3.1 Модуль команд общего назначения UTIL 88
5.3.2 Модуль работы с матрицами MATRIX 97
5.3.3 Модуль инфиксной записи выражений INFIX 101
5.3.4 Модуль построения графиков GRAPH 104
6 СООБЩЕНИЯ ОПЕРАТОРУ 110

1 УСТАНОВКА СИСТЕМЫ TUSUR.EMC

Для установки экземпляра системы TUSUR.EMC нужно запустить установочный файл TUSUREMC_setup.exe дистрибутива и выполнить все шаги мастера установки в следующей последовательности.

1. В первом открывшимся окне необходимо ознакомиться с условиями лицензионного соглашения (рисунок 1.1) и выбрать «Я принимаю условия соглашения», а затем нажать кнопку «Далее».

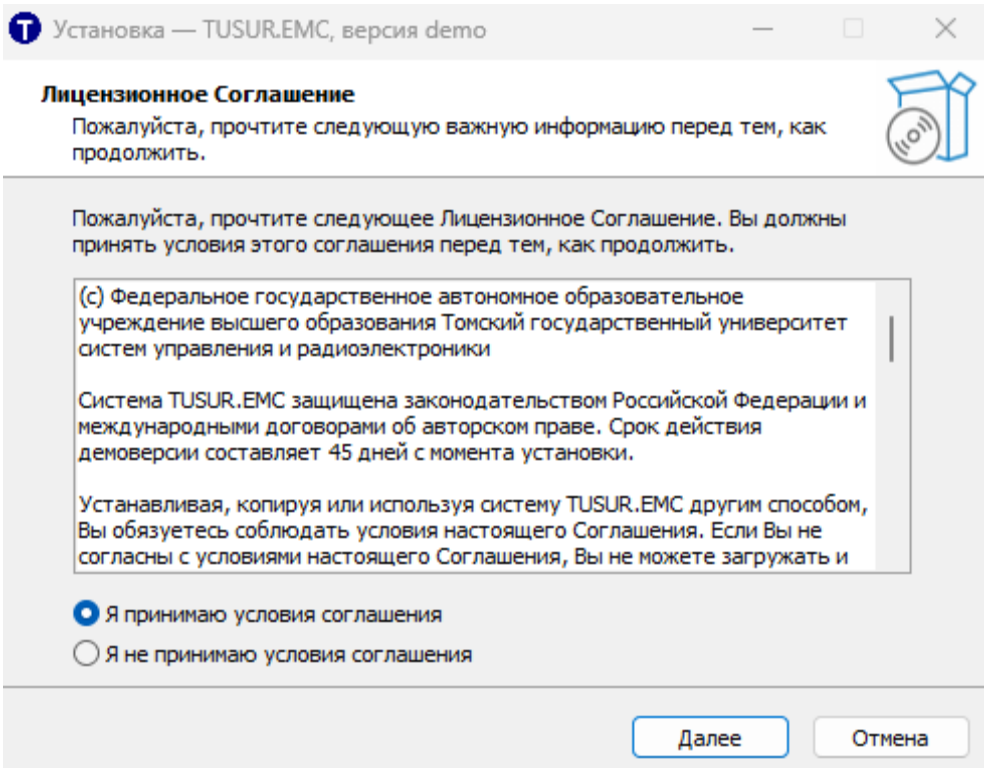


Рисунок 1.1 – Окно лицензионного соглашения системы «TUSUR.EMC» в мастере установки

2. Затем во втором окне во втором окне мастера установки необходимо выбрать путь установки ПО (рисунок 1.2). Для этого нужно нажать кнопку «Обзор» и указать путь установки на диске. Для продолжения установки нужно нажать кнопку «Далее».

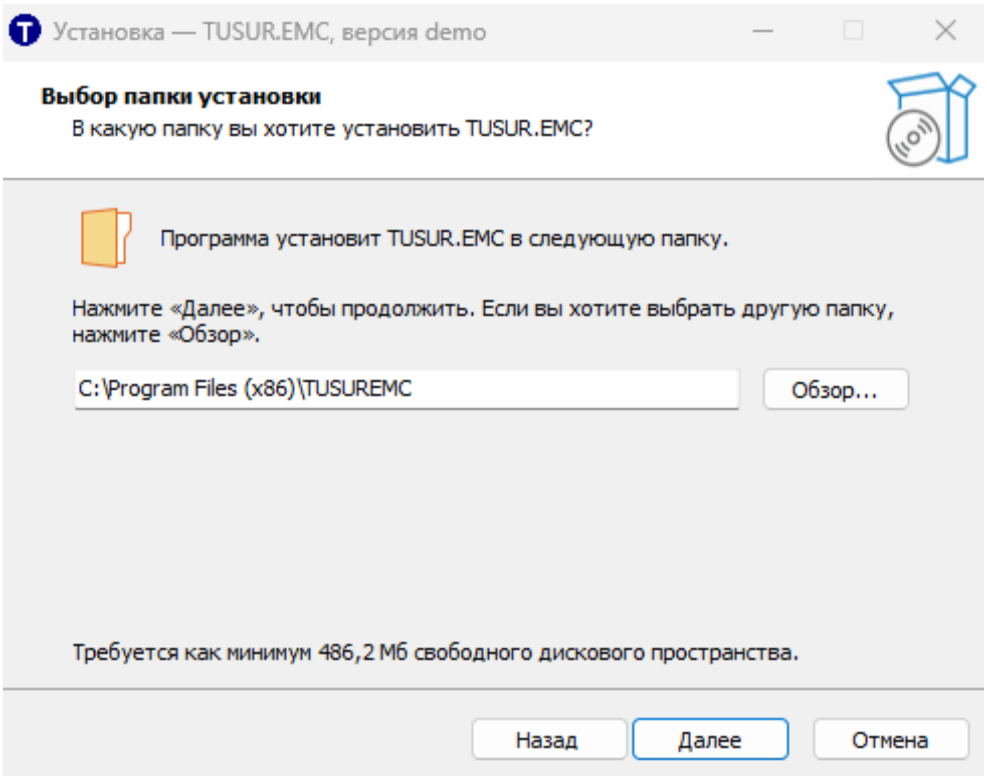


Рисунок 1.2 – Окно выбора папки установки системы «TUSUR.EMC» в мастере установки

3. Затем мастер установки предложит создать папку с возможностью выбора имени папки в меню «Пуск» для создания ярлыка ПО (также в это папке будет размещено руководство пользователя). При установке есть возможность отказаться от создания папки. Для этого в окне мастера установки нужно нажать флаговую кнопку «Не создавать папку в меню «Пуск» (рисунок 1.3). Затем нужно нажать кнопку «Далее».

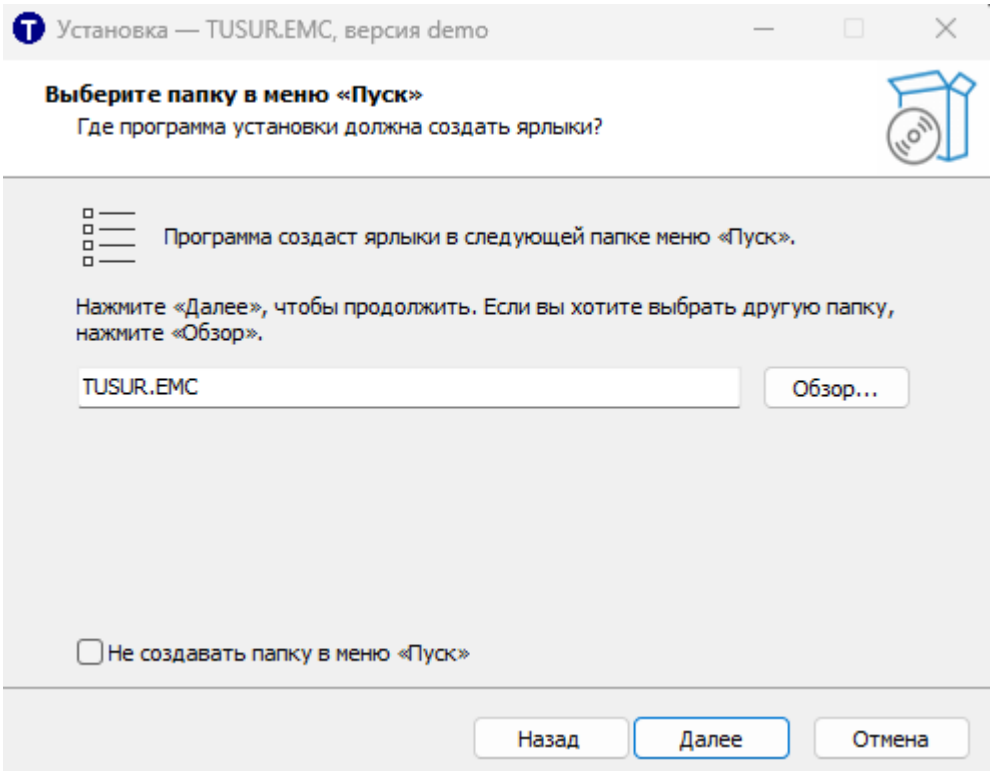


Рисунок 1.3 – Выбор папки в меню «Пуск» в мастере установки

4. Затем мастер установки проинформирует, что всё готово к установке и представит опции установки для проверки (путь установки ПО и название папки в меню «Пуск»). Чтобы приступить к установке нужно нажать кнопку «Установить» (рисунок 1.4).

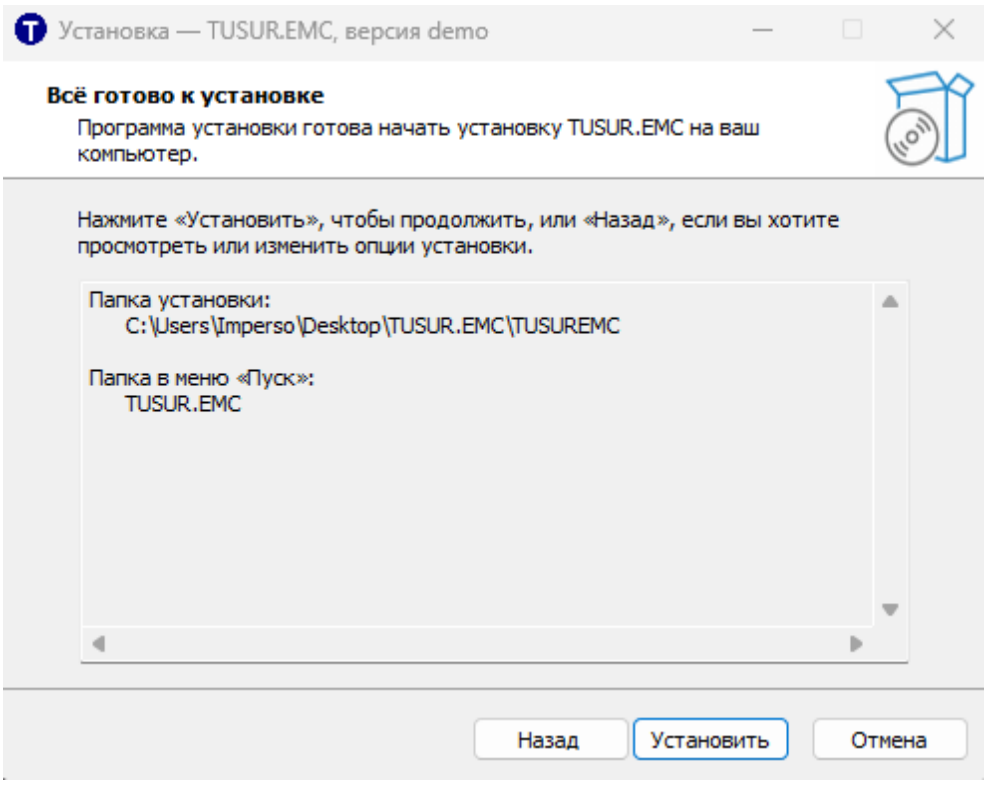


Рисунок 1.4 – Окно проверки опций установки в мастере установки ПО

5. Необходимо дождаться установки ПО на компьютер. После завершения установки необходимо нажать кнопку «Завершить». ПО готово к эксплуатации (рисунок 1.5).

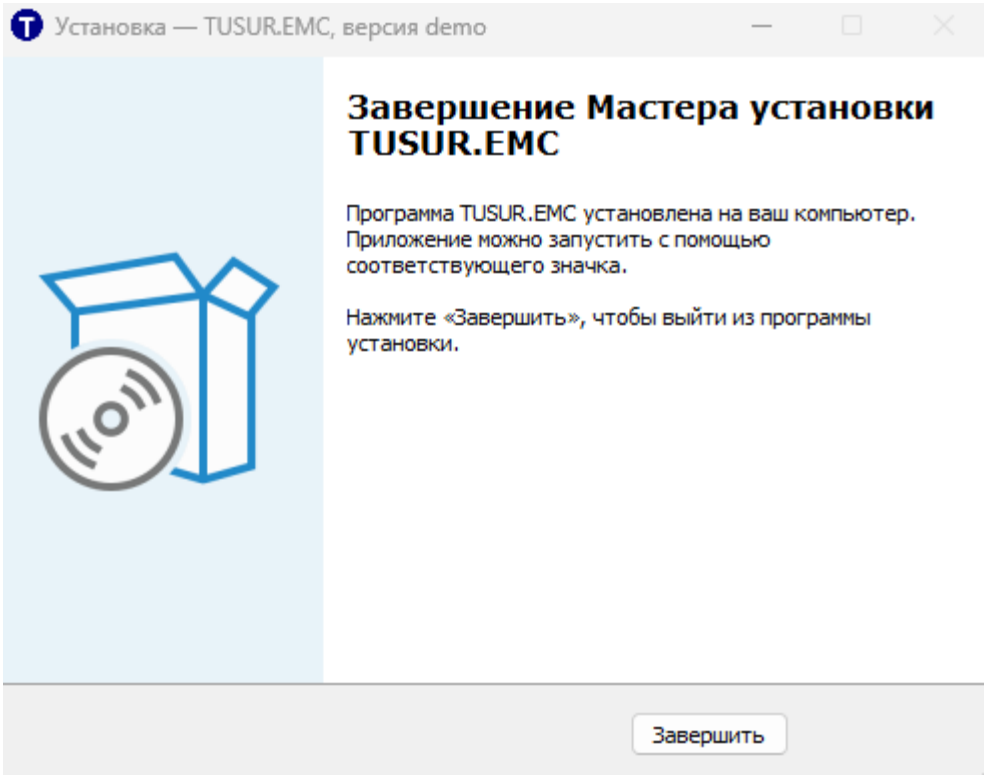


Рисунок 1.5 – Окно завершения установки в мастере установки ПО

После завершения установки система TUSUR.EMC готова к эксплуатации.

2 НАЗНАЧЕНИЕ СИСТЕМЫ TUSUR.EMC

2.1 Цели и назначение системы

Программное обеспечения «Система компьютерного моделирования электромагнитной совместимости ТУСУР.ЭМС (TUSUR.EMC)» (далее система TUSUR.EMC) базируется на математических моделях на основе метода моментов (главный численный метод), где практически все вычисления сведены к матричным операциям, и разрабатывается с 2002 г. на базе собственных научных исследований её авторов с использованием языка программирования C++. Программная реализация системы выполнена по модульному принципу, что позволяет при разработке нового модуля использовать все ранее реализованные возможности системы. Система предназначена для компьютерного моделирования различных электромагнитных задач, включая задачи электромагнитной совместимости.

2.2 Структура системы

Одной из особенностей системы TUSUR.EMC является возможность задания большого количества параметров для возможности проведения различного рода исследований с её использованием, что делает её очень гибкой. При этом практически все параметры имеют значения по умолчанию, что позволяет избежать долгих и рутинных расчётов по выявлению их значений и сразу преступить к использованию системы и вычислениям в ней. Это особенно критично при использовании системы новыми исследователями и в образовательных целях.

Система TUSUR.EMC включает следующие модули: вычислительные для двумерных и трехмерных структур проводников и диэлектриков с произвольными границами (MOM2D, MOM3D), трехмерных структур из проводов с RLC-нагрузками (MOMW), получения временного и частотного откликов (RESPONSE); оптимизации с помощью генетических алгоритмов (GA) и эволюционных стратегий (ES); утилит (UTIL); интерпретации инфиксных выражений (INFIX); построения графиков (GRAPH); матричных операций (MATRIX). Также включены графический интерфейс пользователя (VisualClient), программа для управления системой из командной строки (BasiClient) и ядро системы (TLCORE). Для «общения» пользователя с системой TUSUR.EMC используется интерпретируемый скриптовый язык TALGAT_Script, использующий префиксную запись выражений. Структурная схема системы схематично показана на рисунке 2.1.

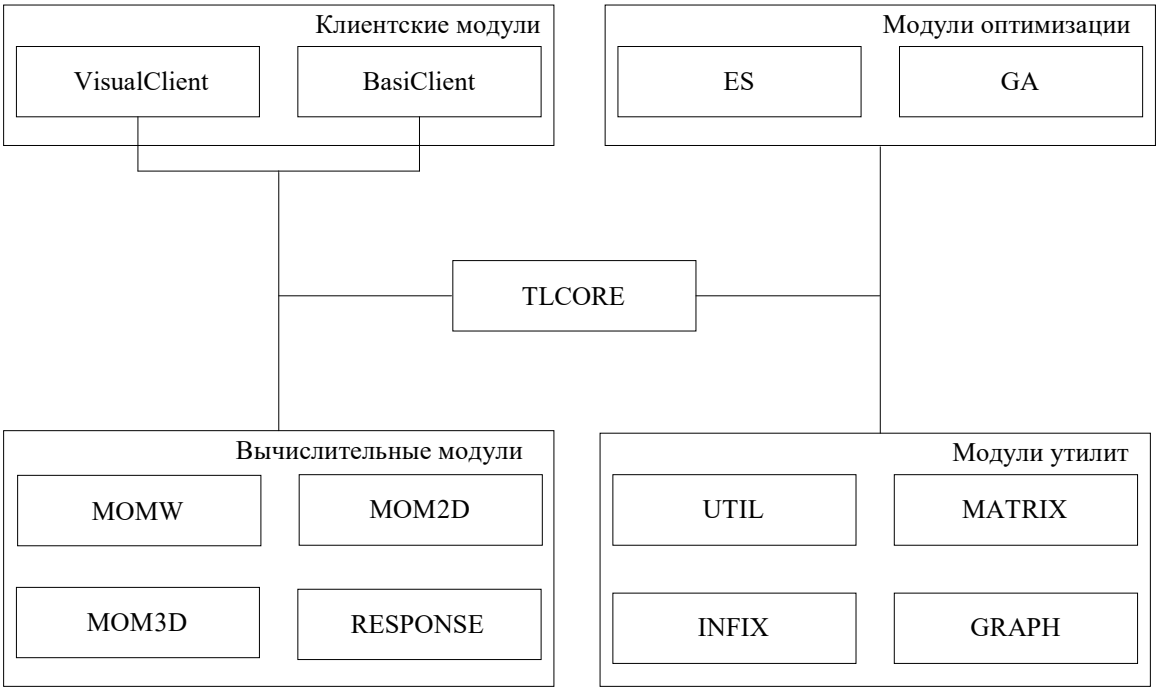


Рисунок 2.1 – Структурная схема системы TUSUR.EMC

При работе с системой действия пользователя над элементом графического интерфейса или ввод текстовой команды преобразуется клиентом ядра системы (исполняемый файл, запускаемый пользователем для начала работы с системой) в текстовую команду и передается ядру системы. Ядро системы распознает команду и вызывает ее обработчик, размещенный в соответствующем динамическом модуле системы (исполняемые файлы, экспортирующие специальные функции, с помощью которых ядро системы при загрузке динамического модуля подключает содержащиеся в модуле программы к системе).

2.3 Выполняемые функции системы, состав, назначение и характеристики её комплексов

Функции, выполняемые системой TUSUR.EMC, описаны в подразделе 3.2.

3 УСЛОВИЯ ВЫПОЛНЕНИЯ СИСТЕМЫ TUSUR.EMC

3.1 Минимальный состав технических средств

В состав технических средств должен входить IBM-совместимый персональный компьютер, включающий в себя: процессор с тактовой частотой, МГц – 366, не менее; оперативную память объемом, Мб – 512, не менее; свободное дисковое пространство объемом, Гб – 1, не менее. 64-битная версия системы требует для работы 64-битный процессор.

3.2 Минимальный состав программных средств

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией одной из операционных систем семейства MS Windows 7 и выше. 64-битная версия системы требует для работы 64-битную операционную систему.

3.3 Требования к персоналу

Минимальное количество персонала, требуемого для работы системы TUSUR.EMC, должно составлять не менее 2 штатных единиц – системный администратор и конечный пользователь программы – оператор. Персонал должен быть аттестован на II квалификационную группу по электробезопасности.

Системный администратор должен иметь высшее профильное образование и сертификаты компании-производителя операционной системы. В перечень задач, выполняемых системным администратором, должны входить задачи: поддержания работоспособности технических средств; установки (инсталляции) и поддержания работоспособности системных программных средств – операционной системы; установки (инсталляции) самой системы. Конечный пользователь программы (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы. Отказы системы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов системы TUSUR.EMC по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

4 ВЫПОЛНЕНИЕ СИСТЕМЫ TUSUR.EMC

4.1 Загрузка и запуск системы TUSUR.EMC

Чтобы запустить установленную ранее систему TUSUR.EMC, необходимо найти на рабочем столе ярлык для визуального клиента системы. Это можно также сделать, если есть ярлык системы в меню Пуск.

4.2 Описание функций системы TUSUR.EMC

4.2.1 Графический интерфейс пользователя

Графический клиент системы является Windows-приложением с полноценным многодокументным графическим интерфейсом пользователя (рисунок 4.1).

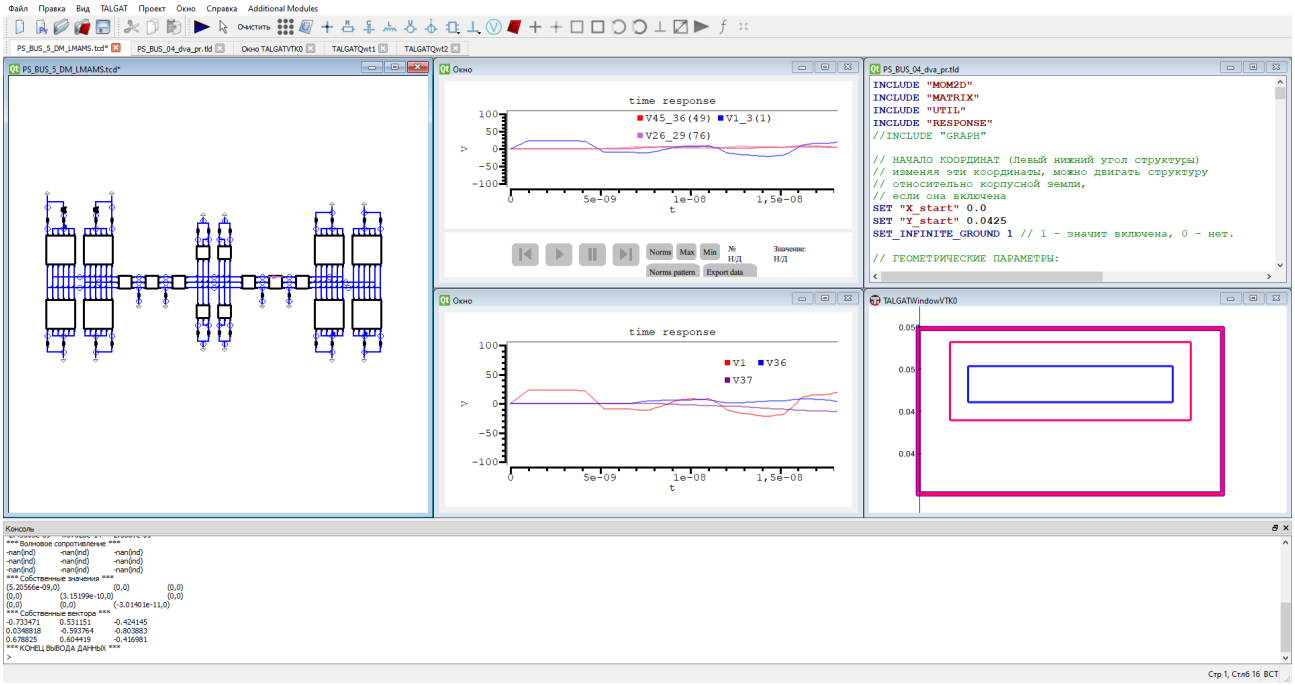


Рисунок 4.1 – Многодокументный интерфейс визуального клиента

Визуальный клиент также позволяет вводить текстовые управляющие команды из текущего активного окна. Данный способ дает возможность заменить набор команд в командной строке одним нажатием пункта «Запуск» в меню «TUSUR.EMC» или нажатием кнопки F5, после чего команды в текущем активном окне обрабатываются так же, как если бы они были набраны построчно в командной строке. Результат исполнения команд выводится в командную консоль (по умолчанию) или в новое окно, создаваемое после

окончания обработки всех команд (если в меню «TUSUR.EMC» выбрана команда «Вывод в новое окно»).

Например, путем ввода команды «DRAW_STRUCTURE структура» (MOMW), «DRAW_CONFIGURATION конфигурация» (MOM2D), «DRAW_CONFIGURATION3D 3d_конфигурация» (MOM3D) в командной панели пользователь может создать окно с графическим отображением исходной структуры (рисунок 4.2). Если отображается двухмерная конфигурация, то пользователь с помощью стрелок на клавиатуре может перемещать изображение конфигурации. Если отображается трехмерная структура, то пользователь с помощью стрелок на клавиатуре может вращать камеру вокруг структуры; с помощью клавиш HOME и END – приближаться и удаляться.

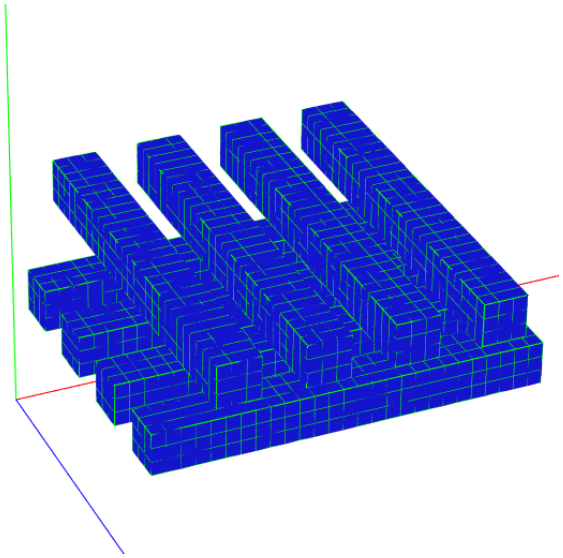


Рисунок 4.2 – Графическое отображение трехмерной конфигурации (MOM3D)

4.2.1.1 Меню Файл

Меню Файл состоит из следующих пунктов:

- Создать документ – создание нового документа на языке TALGAT_Script.
- Новый сценарий Python – создание нового документа на языке Python.
- Создать принципиальную схему – открытие нового окна для создания принципиальных электрических схем.
- Открыть – вывод диалогового окна для открытия существующего файла скрипта.
- Создать конфигурацию – открытие нового окна для создания 2D конфигурации поперечного сечения.
- Открыть конфигурацию… – вывод диалогового окна для открытия существующей 2D конфигурации поперечного сечения.

- Сохранить – сохранение текущего документа.
- Сохранить как – сохранение текущего документа под другим именем.
- Печать – отправка текущего документа на печать.
- Предварительный просмотр – предварительный просмотр текущего документа с учетом параметров страницы.
- Последние файлы – список последних ранее открытых файлов.
- Выход – завершение работы с системой (если при этом ядром системы выполняются вычисления, будет выдан запрос на подтверждение выхода из программы).

4.2.1.2 Меню Правка

Меню Правка состоит из следующих пунктов:

- Отменить – отменить последнюю операцию редактирования.
- Повторить – повторить последнюю отмененную операцию редактирования.
- Выбрать шрифт – вывод диалогового окна для изменения широкого списка параметров шрифта.
- Вырезать – вырезать выделенный текст.
- Копировать – копировать выделенный текст.
- Вставить – вставить текст из буфера обмена.
- Выбрать все – выделить весь текст в активном окне.
- Комментировать/раскомментировать – комментировать/раскомментировать выделенный текст, для его учета или не учета при выполнении скрипта.
- Найти – вывести диалоговое окно поиска.
- Найти далее – перейти к следующему результату поиска.
- Найти предыдущее – перейти к предыдущему результату поиска.
- Заменить – вывести диалоговое окно замены.
- Выполнить выделенное – запустить выделенный участок скрипта.

4.2.1.3 Меню Вид

Меню Вид состоит из следующих пунктов:

- Темная тема – переход на темную тему оформления интерфейса системы.
- Показать панель инструментов – позволяет показать или скрыть панель инструментов.
- Показать консоль – позволяет показать или скрыть панель консоли.

- Показать строку состояния – позволяет показать или скрыть панель состояния.
- Показать проекты – позволяет показать или скрыть панель проектов.
- Показать файловую систему – позволяет показать или скрыть панель физические/виртуальные диски в текущей файловой системе.
- Перенос строк – масштабирование окна файла скрипта.
- Панель рисования – позволяет показать или скрыть панель рисования.
- Показать подписи – позволяет показать или скрыть подписи элементов принципиальных электрических схем.
- Показать значения – позволяет показать или скрыть значения некоторых элементов принципиальных электрических схем.
- Редактировать переменные – вывести панель, содержащую созданные переменные с возможностью их редактирования.

4.2.1.4 Меню TUSUR.EMC

Меню TUSUR.EMC состоит из следующих пунктов:

- Запуск – выполнить скрипт в активном окне.
- Подсветка кода – включение и отключение подсветки синтаксиса.
- Автозавершение кода – включения и отключения автозавершения кода по нажатию сочетания клавиш CTRL + пробел.
- Автоматическая конвертация в Python – включение и отключение автоматического перевода кода из TALGAT_Script в Python при выполнении скрипта в активном окне.
- Вывод в новое окно – переключение вывода результатов вычислений из командной панели (по умолчанию) в новое окно.
- Тихий режим – запрет вывода окна исключений (сообщения об ошибках выводятся вместе с результатами вычислений).
- Переключить на старый интерпретатор/интерпретатор Python – выбор языка на которым будет сформирован и выполнен скрипт (интерпретатор TALGAT_Script или интерпретатор Python).

4.2.1.5 Меню Окно

Меню Окно состоит из следующих пунктов:

- Закрыть – закрыть текущее окно.

- Закрыть все – закрыть все открытые окна.
- Мозаика – разместить все открытые окна в виде мозаики.
- Каскад – разместить все открытые окна в виде каскада.
- Следующее – сделать активным следующее окно.
- Предыдущее – сделать активным предыдущее окно.
- Список открытых окон – список открытых окон, который позволяет быстро переключаться между открытыми окнами.

4.2.2 Скриптовый язык TALGAT_Script

TALGAT_Script – встроенный интерпретируемый скриптовый язык, с помощью которого пользователь «общается» с системой. TALGAT_Script создавался с учетом следующих требований, перечисленных в порядке их приоритета при разработке языка: предоставление пользователю как можно большего числа возможностей системы, доступных при программировании на C++; простота реализации; простота языка; быстродействие.

4.2.2.1 Синтаксис: общие сведения

Скрипт на языке TALGAT_Script представляет собой последовательность символов – поток ввода (input stream). Не имеет никакого значения источник потока ввода – это может быть файл на диске, буфер в памяти или текстовое поле визуального клиента системы.

В ядро системы встроен интерпретатор, распознающий во входном потоке команды (commands). Команда – это последовательность символов, не содержащая символов-разделителей (пробел, табуляция), символов «’», «”» и символа новой строки. Каждая команда во входном потоке должна начинаться с новой строки. Чтобы легко различать команды в больших скриптах, все команды системы пишутся заглавными буквами (например, «SMN»).

Каждой команде соответствует свой обработчик (handler), выполняющий соответствующую операцию. Не имеет никакого значения, где находится обработчик – в ядре, модуле или клиенте системы. После распознавания команды интерпретатор вызывает соответствующий ей обработчик.

Команды могут иметь параметры (parameters), которые отделяются от команды и друг от друга символами-разделителями (пробел, табуляция). Например, «SMN_C conf», где SMN_C – команда, а conf – её параметр. Параметры могут быть шести типов (types):

- пустой тип [NULL] или [null] – значение отсутствует (например, пользователь забыл его ввести);

- длинное целое long – 4-байтное знаковое целое число (например, «-44», «5»);
- двойной точности double – 8-байтное число с плавающей точкой (например, «0.256», «6.»). Следует помнить, что для записи дробных чисел в системе вместо запятой используется точка, которая необходима, даже если после нее нет значащих цифр;
- строка string – последовательность символов, не содержащая символов-разделителей (пробел, табуляция, символ новой строки) и кавычек (например, «это_строка»);
- субъект subject – непрерывная область в памяти, содержащая данные определенного типа, в том числе конфигурации, структуры, матрицы (аналог объектов языка C++, при распечатке выводится в виде строки «[subject]»);
- комплексное число complex – два 8-байтных числа с плавающей точкой (например, «(0.344,-5.6)» – без пробела, точка также необходима, даже если после нее нет значащих цифр, например, «(0.,-5.)»);
- матрица matrix – матрица из чисел типа double или complex, при этом матрицы из чисел типа double обозначаются в документации как тип real_matrix, матрицы из чисел типа complex – как тип complex_matrix.

Если команда имеет параметры, то обработчик «просит» интерпретатор найти их. Интерпретатор читает входной поток до конца текущей строки, распознает типы введенных параметров и передает их значения обработчику. Таким образом, не допускается наличие новых строк между командой и ее параметрами. Допускается использование в качестве параметра, тип которого должен быть, например, double, параметр типа long, и наоборот. При этом выполняется автоматическое преобразование типов. Тем не менее, важно отличать: «4.» (четыре с плавающей точкой) имеет тип double, а «4» (просто четыре без точки) – тип long по аналогии с литералами C++.

После исполнения необходимых операций обработчик возвращает значение-результат (value_result), имеющее один из приведенных выше типов. Данное значение содержит результат проведенных операций и может использоваться как параметр другой команды. Таким образом, команды могут быть вложенными (nested), то есть вместо параметра команды пользователь может ввести имя другой команды, которая возвращает значение-результат того же типа, что и параметр первой команды. Например, «PLUS 2. PLUS 2. 2.» эквивалентно 2 + 2 + 2.

4.2.2.2 Стандартные команды

Стандартные команды (standard commands) – команды, которые реализованы в ядре системы и доступны сразу после запуска любого клиента системы. Все команды, для

которых не указано иное, возвращают в случае успешного исполнения «1» типа long и «0» типа long во всех других случаях. Такое поведение описывается как возвращаемый параметр типа bool.

Рассмотрим стандартные команды системы более подробно. При этом важно помнить, что в этом случае должен быть выбран интерпретатор TALGAT_Script в меню TUSUR.EMC, т.к. по умолчанию для работы с программной консолью выбран интерпретатор Python.

4.2.2.3 Команды ECHO, HELP и LINE_TO_STRING

Распечатывает в консоль или новое окно значение параметра, в том числе значение-результат, возвращаемое обработчиком команды. Так, если параметр – команда, например «ECHO что_то» распечатает строку «что_то», а «ECHO PLUS 2. 2.» – 4 (рисунок 4.3).

Рисунок 4.3 – Пример использования команды ECHO:
вывод результатов в программную консоль (*а*) или новое окно (*б*)

Команда HELP или кратко «?» возвращает информационную строку. Так, «ECHO HELP» или «ECHO ?» распечатает строку, содержащую информацию о запущенном клиенте системы, и список всех доступных команд с указанием типов и назначения параметров и возвращаемых значений, при этом дополнительная информация выводится через знак «_» (рисунок 4.4*а*). Если в качестве параметра ввести имя какой-либо команды, будет выведена информация только по команде с таким именем. Так, результат использования «ECHO HELP LINE_TO_STRING» (LINE_TO_STRING) приведен на рисунке 4.4*б*. Команда LINE_TO_STRING имеет один параметр – строка символов.

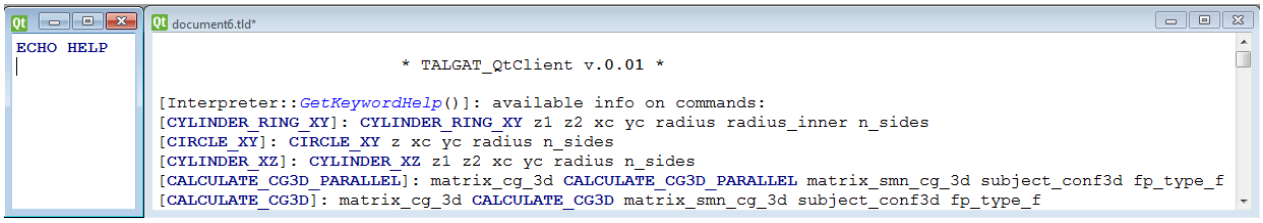


Рисунок 4.4 – Результаты выполнения команд «ECHO HELP» (*а*)
и «ECHO HELP LINE_TO_STRING» (*б*)

4.2.2.4 Команда INCLUDE

Команда INCLUDE позволяет загружать модули системы. Первый параметр – название модуля (обычно имя динамической библиотеки, в которой реализован модуль, без расширения). Второй необязательный параметр – имя субмодуля. Если имя субмодуля не задано, оно считается тем же, что и имя модуля. В результате пользователь получает доступ к командам, реализованным в загруженном модуле. Чтобы избежать конфликтов с командами, имеющими те же имена, что и загружаемые модули (в системе TUSUR.EMC таких команд нет, однако, пользователь может случайно создать такую команду самостоятельно), параметр команды INCLUDE рекомендуется указывать в кавычках, рисунок 4.5.

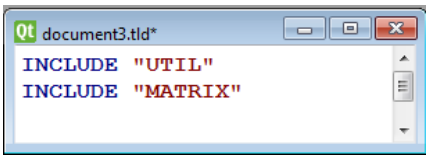


Рисунок 4.5 – Пример загрузки модулей UTIL и MATRIX

4.2.2.5 Команда комментирования

Команда «//» просто игнорирует свой параметр. Такое поведение позволяет использовать команду «//» для комментирования строк. Ставить символ-разделитель (пробел или табуляцию) после команды «//» необязательно. При этом следует помнить, что после других команд использование символа-разделителя (пробел или табуляцию) является обязательным.

4.2.2.6 Команды для создания динамических команд: CREATE_KEYWORD и END_CREATE_KEYWORD

Команда CREATE_KEYWORD позволяет создавать динамические команды (dynamic commands), например myCommand. Первый параметр является обязательным и является именем новой динамической (пользовательской) команды, заключенным в кавычки, т.е. "myCommand". Второй параметр, являющийся необязательным, содержит строку, написанную на латинице, которая будет отображаться при использовании команды «HELP myCommand», и является комментарием по её использованию. После создания динамической команды интерпретатор переходит в специальный режим и сохраняет поток ввода как строку, пока не будет введена команда END_CREATE_KEYWORD которая возвращает интерпретатор в обычный режим. Указание параметров команды END_CREATE_KEYWORD не является обязательным. Однако в качестве параметров может указано возвращаемое значение или другие команды. Пример создания пользовательской команды приведен на рисунке 4.6. Так, пользователь создает новую команду с именем «myCommand», которая будет выводить результат сложения двух чисел с помощью команды ECHO. В качестве комментария к создаваемой команде указано, что она является «example of user command». Создание новой команды завершается вводом команды «END_CREATE_KEYWORD -5». Для вызова новой команды надо указать её имя «myCommand».

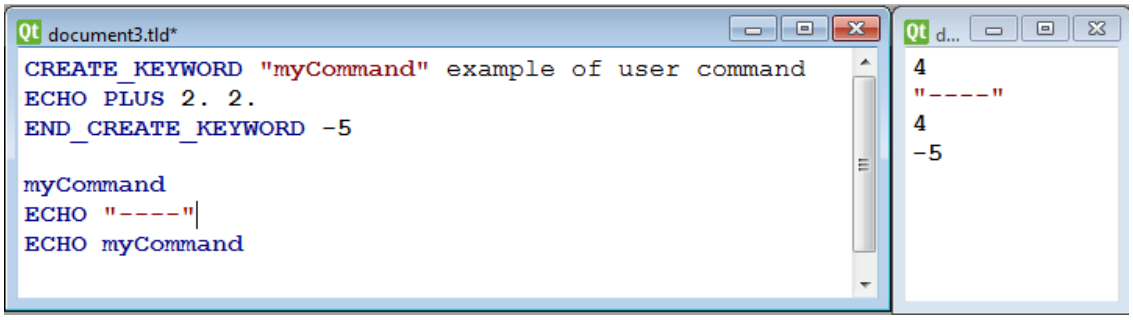


Рисунок 4.6 – Пример создания и использования динамической команды

Пользователь может переопределять (override), то есть создавать заново, ранее созданные динамические команды, однако это запрещено для статических команд (static commands), под которыми понимаются команды, реализованные в системе, в частности, стандартные команды системы и команды из её модулей. Если при загрузке нового модуля системы выяснится, что пользователь создал команду с именем, совпадающим с именем команды из модуля, то динамическая команда будет удалена (без сообщения об этом пользователю).

Создание вложенных динамических команд не поддерживается. Если после вызова команды CREATE_KEYWORD и перехода интерпретатора в специальный режим команда CREATE_KEYWORD используется еще раз (до считывания команды END_CREATE_KEYWORD), то она просто сохраняется как строка и при последующем вызове созданной динамической команды происходит ошибка. Если необходимо использовать внутри динамической команды с именем «myCommand» другую динамическую команду «myCommand2», следует создать динамическую команду «myCommand2» перед созданием «myCommand», рисунок 4.7.

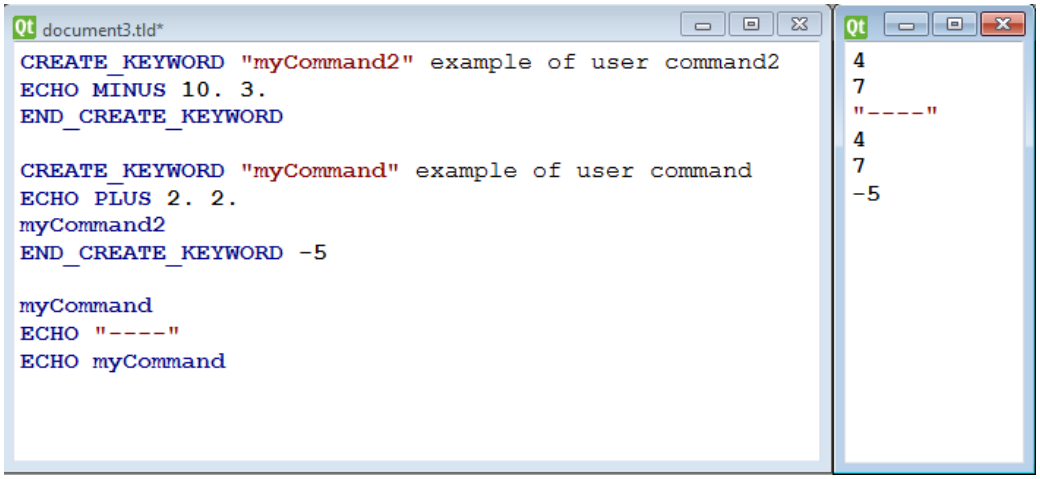


Рисунок 4.7 – Пример совместного использования двух динамических команд

4.2.2.7 Команда RETURN

В рассмотренных примерах созданные динамические команды выводили на печать требуемые данные. Для того, чтобы динамические команды возвращали данные может быть использована команда RETURN, листинг 4.1.

Листинг 4.1 – Команды RETURN

```
CREATE_KEYWORD "myCommand"
   RETURN PLUS 2. 2.
END_CREATE_KEYWORD
ECHO myCommand
```

4.2.2.8 Команды SET и GET: работа с переменными

Команда SET присваивает значение переменной, указанной в качестве первого параметра, в значение второго параметра, т.е. её первый параметр – это имя переменной, заключенное в кавычки, например «"имя_переменной"», второй параметр – значение переменной. Если переменная с таким именем не существует, то она будет создана. Команда

GET позволяет получать значение переменной, заключенное в кавычки, например «"имя_переменной"», (листинг 4.2). Если переменной с таким именем не существует, то генерируется исключение.

Листинг 4.2 – Команды SET и GET

```
SET "var" 55.2 // создание переменной "var" и присвоение её значения 55.2
ECHO GET "var" // выводит на печать значения переменной "var"
```

Для удобства пользователя команда SET также позволяет создавать новую динамическую команду с таким же именем, как и у переменной. Эта команда при вызове «подставляет» вместо себя значение одноименной переменной, листинг 4.3.

Листинг 4.3 – Команда var

```
SET "var" 55.2 // записывает значение переменной "var"
CREATE_KEYWORD "var" // создает команду var
   RETURN GET "var"
END_CREATE_KEYWORD
ECHO var // выводит 55.2
```

В результате два способа получить и распечатать значение заданной переменной:

1. Использовать команду GET с именем переменной в качестве аргумента: «ECHO GET "var"», где команда GET возвращает значение переменной var. При этом, если необходимо использовать имя переменной в качестве параметра команд SET и GET, то имя переменной нужно заключать в кавычки.

2. Использовать команду «ECHO var», где значение переменной var будет получено вызовом одноименной динамической команды var. При этом вместо имени переменной будет «подставлено» значение переменной, что и нужно во всех случаях, за исключением повторного вызова команд SET и GET.

4.2.2.9 Управление выводом отладочной информации

Команда SET_STATUS_MESSAGE с параметром 1 позволяет включить вывод отладочной информации, с параметром 0 – отключить. По умолчанию вывод отладочной информации выключен. Узнать текущее состояние режима вывода отладочной информации можно командой «ECHO GET_STATUS_MESSAGE».

4.2.2.10 Типичные ошибки при написании скриптов

Пожалуй самой типичной ошибкой при написании скриптов является «[Interpreter]: command parameter is missing or has incorrect type.» Существует две причины этой ошибки:

1. Пропущен параметр команды, например: «ECHO GET». В этом случае у команды «GET» отсутствует необходимый параметр.

2. Неверный тип параметра, например: «ECHO GET -1». Команда «GET» принимает в качестве параметра строку с именем переменной. «-1» имеет числовой тип long и не является строковым типом string.

Часто такая ошибка встречается в сложных скриптах, где параметрами команды являются результаты вызова других команд. В этом случае рекомендуется создать для отладки дополнительные переменные, сохранить в них результаты вызова вложенных команд и распечатать содержимое этих переменных для контрольной проверки.

4.2.3 Интерпретатор Python

Как уже упоминалось ранее, кроме использования TALGAT_Script в системе реализована возможность использования интерпретатора Python. С его помощью можно как писать новые скрипты, так и конвертировать готовые скрипты, написанные на TALGAT_Script.

Написание скриптов на языке Python осуществляется в окне «New Python Script», доступном из меню Файл. Для конвертации готовых скриптов надо выбрать пункт «Автоматическая конвертация в Python» в меню «TUSUR.EMC».

5 МОДУЛИ СИСТЕМЫ TUSUR.EMC

5.1 Вычислительные модули

5.1.1 Модуль двумерного электростатического анализа MOM2D

5.1.1.1 Общие сведения

Модуль MOM2D предназначен для электростатического анализа двумерных конфигураций проводников и диэлектриков и позволяет вычислить матрицы погонных первичных параметров **R** (Ом/м), **L** (Гн/м), **C** (Ф/м) и **G** (См/м) линий передачи с магнитодиэлектрическими границами произвольной сложности. Матрицы **R** и **G** называются матрицами погонных сопротивлений и проводимостей соответственно, а **C** и **L** – матрицами погонных коэффициентов электростатической и электромагнитной индукции соответственно. (Матрицу **C** часто называют ёмкостной матрицей.) Кроме того, в системе реализовано вычисление этих двух матриц в воздухе, т.е. **C0** (Ф/м) и **L0** (Гн/м). При этом стоит отметить, что вычисление матрицы **C** является наиболее сложной задачей, т.к. остальные матрицы являются её производными. Вычисленные матрицы затем используются при решении телеграфных уравнений Хевисайда или производных от них для анализа целостности сигналов, получения временного отклика и других параметров в модуле RESPONSE.

Для выполнения задач двухмерного электростатического анализа необходимо загрузить модули UTIL, MATRIX и MOM2D, листинг 5.1 (описание модулей UTIL и MATRIX приведено после вычислительных моделей).

Листинг 5.1 – Подключение модулей, необходимых для 2D электростатического анализа

```
INCLUDE "UTIL"
INCLUDE "MATRIX"
INCLUDE "MOM2D"
```

Особое внимание нужно обратить на правила задания относительных диэлектрических и магнитных проницаемостей ER и MU. Для любого интервала магнитодиэлектрической границы должно быть задано по две пары значений ER и MU – ER_PLUS и ER_MINUS, MU_PLUS и MU_MINUS. При обходе интервала от начала к концу справа от интервала «находятся» ER_PLUS и MU_PLUS, а слева – ER_MINUS и MU_MINUS. На рисунке 5.1 это правило показано на примере ER_PLUS и ER_MINUS. Если

не задавать значения ER и MU, то по умолчанию они инициализируются значениями, приведенными в таблице 5.1.

Таблица 5.1 – Значения ER и MU по умолчанию

ER	ER_PLUS	1.	MU	MU_PLUS	1.
	ER_MINUS	1.00001		MU_MINUS	1.00001

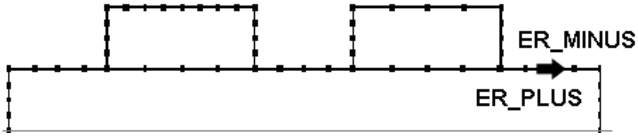


Рисунок 5.1 – Положения ER_PLUS и ER_MINUS в зависимости от направления обхода границы раздела диэлектрик-диэлектрик

Не допускается равенство значений ER_PLUS и ER_MINUS, так как это соответствует случаю, когда граница раздела двух сред отсутствует. Для любого проводникового интервала имеют значение только ER_PLUS и MU_PLUS, которые автоматически распределяются на левую или правую сторону границу проводника в зависимости от того, с какой стороны находится магнитодиэлектрик.

5.1.1.2 Пример создания и визуализации конфигурации с бесконечной плоскостью земли

Рассмотрим пример создания конфигурации на примере 2-проводной микрополосковой линии передачи, состоящей из двух проводников на диэлектрической подложке из материала с относительной диэлектрической проницаемостью равной 2, расположенной над бесконечной плоскостью земли. Её поперечное сечение схематично показано на рисунке 5.2.

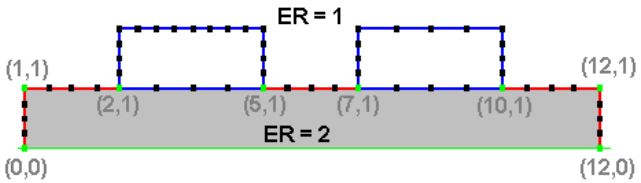


Рисунок 5.2 – Поперечное сечение 2-проводной микрополосковой линии передачи

Под бесконечной плоскостью земли понимается идеально проводящая плоскость при $y = 0$. Для её здания используется команд SET_INFINITE_GROUND со значением единственного параметра равным 1. (Если параметр имеет значение 0, то это соответствует исключению из конфигурации бесконечной плоскости земли).

Рассмотрим процесс создания левого проводника. Для создания проводников используется команда без параметров CONDUCTOR. Начнём с задания границы проводника, соприкасающейся с подложкой. Сначала установим необходимое число подынтервалов для разбиения границ, для примера используем разбиение на 4 подынтервала, командой «SET_SUBINTERVALS 4» Так как относительная диэлектрическая проницаемость подложки равна 2, то устанавливаем значение ER_PLUS равным 2. Далее командой «LINE 2. 1. 5. 1.» создадим интервал, соединяющий точки $x = 2$, $y = 1$ и $x = 5$, $y = 1$. (Здесь и далее все расстояния в метрах.) Поскольку остальные 3 границы проводника граничат с диэлектриком, относительная диэлектрическая проницаемость которого равна 1, то устанавливаем значение ER_PLUS равным 1. Далее строим второй проводниковый интервал «LINETO 5. 2.». Числа 5 и 2 – это координаты конца интервала, за начало которого автоматически выбирается конец предыдущего, то есть $x = 5$, $y = 1$ (также можно воспользоваться полной командой «LINE 5. 1. 5. 2.»). Перед созданием третьего интервала установим число подынтервалов в 8, затем создаём третий интервал, возвращаем исходное число подынтервалов 4 и проводим последний проводниковый интервал. В результате сформируется следующий список команд:

```
SET_INFINITE_GROUND 1
SET "Er" 2.
CONDUCTOR
   SET_ER_PLUS Er
LINE 2. 1. 5. 1.
SET_ER_PLUS 1.
LINETO 5. 2.
   SET_SUBINTERVALS 8
   LINETO 2. 2.
   SET_SUBINTERVALS 4
   LINETO 2. 1.
```

Результирующая конфигурация схематично показана на рисунке 5.3. Видно, что верхний интервал разбит на 8 подынтервалов, а все остальные – на 4.

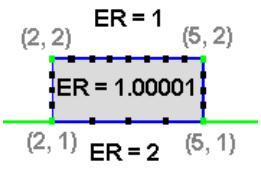


Рисунок 5.3 – Конфигурация проводника

Аналогично создаётся правый проводник:

```
CONDUCTOR
   SET_ER_PLUS Er
   LINE 7. 1. 10. 1.
   SET_ER_PLUS 1.
   LINETO 10. 2.
```

```
LINETO 7. 2.
LINETO 7. 1.
```

Последними создаются границы диэлектрик-диэлектрик (область, закрашенная серым на рисунке 5.2). При этом следует помнить, что создаваемые диэлектрические интервалы (отображаемые визуальным клиентом красными линиями) не должны накладываться на уже созданные проводниковые интервалы (отображаемые визуальным клиентом синими линиями):

```
DIELECTRIC
  SET_ER_PLUS 1.
  SET_ER_MINUS Er
  LINE 12. 0. 12. 1.
  LINETO 10. 1.
  LINE 7. 1. 5. 1.
  LINE 2. 1. 0. 1.
  LINETO 0. 0.
```

Последним шагом является сохранение созданной конфигурации в переменную, например conf_ig, и её визуальное представление в графическом окне с помощью комнд:

```
SET "conf_ig" GET_CONFIGURATION_2D
DRAW_CONFIGURATION conf_ig
```

Итоговый набор команд, содержащий динамическую команду, для построения конфигурация с бесконечной землёй, приведен в листинге 5.2. Результат его выполнения приведен на рисунке 5.4.

Листинг 5.2 – Создание конфигурации с бесконечной плоскостью земли

```
INCLUDE "UTIL"
INCLUDE "MATRIX"
INCLUDE "MOM2D"
SET "Er" 2.
CREATE_KEYWORD "create_conf"
    SET_INFINITE_GROUND 1
    SET_SUBINTERVALS 4
    CONDUCTOR
      SET_ER_PLUS Er
      LINE 2. 1. 5. 1.
      SET_ER_PLUS 1.
      LINETO 5. 2.
      SET_SUBINTERVALS 8
      LINETO 2. 2.
      SET_SUBINTERVALS 4
      LINETO 2. 1.
    CONDUCTOR
      SET_ER_PLUS Er
      LINE 7. 1. 10. 1.
      SET_ER_PLUS 1.
      LINETO 10. 2.
```

```
        LINETO 7. 2.
        LINETO 7. 1.
    DIELECTRIC
        SET_ER_PLUS 1.
        SET_ER_MINUS Er
        LINE 12. 0. 12. 1.
        LINETO 10. 1.
        LINE 7. 1. 5. 1.
        LINE 2. 1. 0. 1.
        LINETO 0. 0.
    SET "conf_ig" GET_CONFIGURATION_2D
    DRAW_CONFIGURATION conf_ig
END_CREATE_KEYWORD
create_conf
```

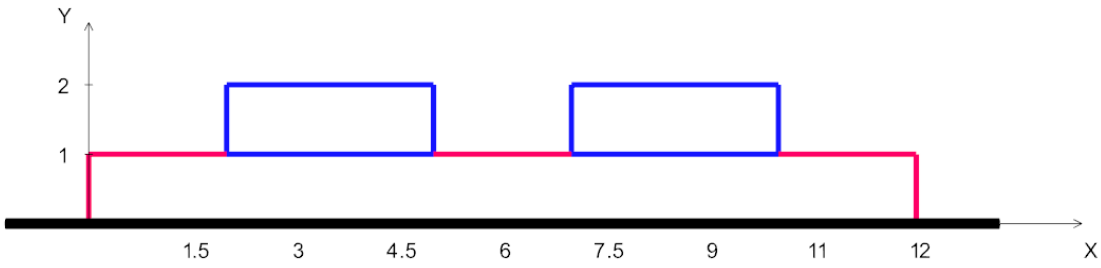


Рисунок 5.4 – Поперечное сечение 2-проводной микрополосковой линии передачи с бесконечной плоскостью земли

При вызове команды DRAW_CONFIGURATION (DRAW_CONFIGURATION2D или DRAW_CONF2D) происходит визуализации конфигурации. Границы диэлектриков отображаются отрезками красного цвета, проводников – синего, заземленных проводников (CONDUCTOR_GROUNDED) – темно-синего, а бесконечной плоскости земли – черного.

При наведении курсора мыши на участок конфигурации информация об участке показывается в следующем формате: номер интервала (количество подынтервалов, номер подынтервала, заземлён ли проводник). Если включен режим отображения распределения плотности заряда, формат отображения информации об участке конфигурации следующий: номер подынтервала (значение плотности заряда).

Список горячих клавиш при визуализации конфигурации MOM2D:

- 'a': скрыть/показать оси координат;
- 'c': показать/скрыть цветовую шкалу плотности зарядов. Каждому цвету ставится в соответствие значение плотности заряда, этим цветом окрашивается соответствующий участок конфигурации;
- 'q': выключить/включить режим визуализации плотности заряда;
- 'r': сменить текущий проводник;

- 'e': сменить режим отображения диэлектрической проницаемости (6 режимов: без отображения, Er – относительная диэлектрическая проницаемость, Mu, мнимая часть Er, TanDelta, номера проводников);
- 's': показать/скрыть границы подынтервалов;
- 'n': показать/скрыть сетку координат;
- 'l': сменить режим отображения интервалов (2 режима: линии с переменной толщиной, линии с постоянной толщиной);
- '+'/'-': увеличить/уменьшить конфигурацию;
- Shift и '+'/'-': увеличить/уменьшить толщину линий.
- Поворот – Ctrl+левая кнопка мыши. Перемещение – правая кнопка мыши. Увеличение/уменьшение конфигурации – колесо прокрутки мыши или Ctrl+правая кнопка мыши.

Пример изменения режима визуализации приведен на рисунке 5.5. Видно, что за счет задания разного числа подынтервалов для каждой границы может быть построена неравномерная сетка.

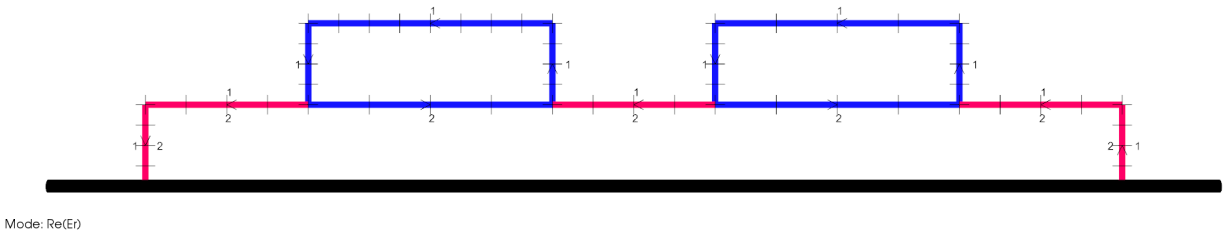


Рисунок 5.5 – Изменение режима визуализации конфигурации

5.1.1.3 Создание конфигурации с автосегментацией

Кроме описанного ранее задания требуемого числа подынтервалов для разбиения интервалов (команда SET_SUBINTERVALS) может быть использована автосегментация по заданной длине подынтервала. При этом число подынтервалов на каждом интервале вычисляется автоматически. При этом следует помнить, что задание очень маленькой длины подынтервала при автосегментации хоть и повышает точность результатов, но значительно увеличивает время выполнения скрипта. Так, увеличение общего числа подынтервалов N в два раза увеличит время вычисления примерно в 8 раз, поскольку затраты времени $\sim O(N^3)$.

Рассмотрим пример использования автосегментации на примере конфигурации из листинга 5.2. Для этого удалим все строки, содержащие команды SET_SUBINTERVALS, а в начале зададим автосегментацию с длиной подынтервалов 0.2

(SET_AUTO_SEGMENT_LENGTH 0.2). После использования автосегментацию её необходимо выключить (SET_AUTO_SEGMENT_LENGTH 0.). Кроме того, выключать автосегментацию рекомендуется перед использованием команды SET_SUBINTERVALS, т.к. автосегментация имеет больший приоритет, и поэтому при её включении команды SET_SUBINTERVALS будут игнорироваться. Итоговый скрипт приведен на листинге 5.3.

Листинг 5.3 – Создание конфигурации с бесконечной плоскостью земли и автосегментацией

```
INCLUDE "UTIL"
INCLUDE "MATRIX"
INCLUDE "MOM2D"
SET "Er" 2.
SET_AUTO_SEGMENT_LENGTH 0.2
CREATE_KEYWORD "create_conf"
    SET_INFINITE_GROUND 1
    CONDUCTOR
       SET_ER_PLUS Er
       LINE 2. 1. 5. 1.
       SET_ER_PLUS 1.
       LINETO 5. 2.
       LINETO 2. 2.
       LINETO 2. 1.
    CONDUCTOR
       SET_ER_PLUS Er
       LINE 7. 1. 10. 1.
       SET_ER_PLUS 1.
       LINETO 10. 2.
       LINETO 7. 2.
       LINETO 7. 1.
    DIELECTRIC
       SET_ER_PLUS 1.
       SET_ER_MINUS Er
       LINE 12. 0. 12. 1.
       LINETO 10. 1.
       LINE 7. 1. 5. 1.
       LINE 2. 1. 0. 1.
       LINETO 0. 0.
    SET "conf_ig" GET_CONFIGURATION_2D
    DRAW_CONFIGURATION conf_ig
END_CREATE_KEYWORD
create_conf
SET_AUTO_SEGMENT_LENGTH 0.
```

На рисунке 5.6 приведен результат выполнения скрипта из листинга 5.3. Видно, что при использовании автосегментации длина всех подынтервалов одинакова (равномерная сетка).

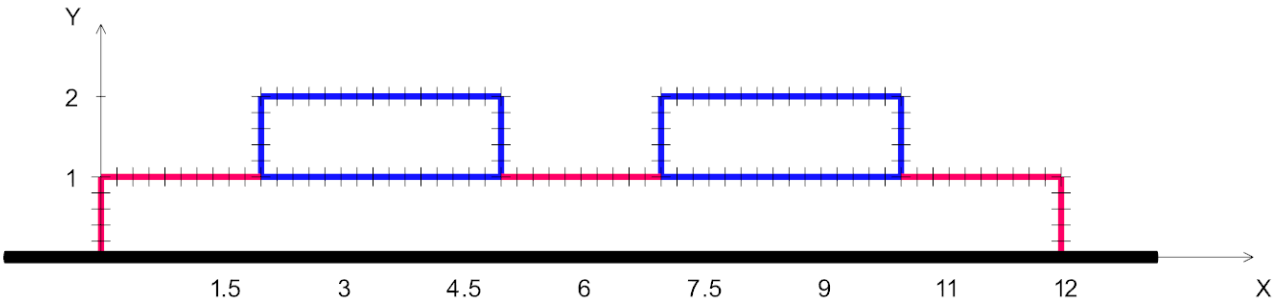


Рисунок 5.6 – Конфигурация с бесконечной плоскостью земли
при автосегментации с длиной подынтервалов 0.2

5.1.1.4 Пример создания конфигурации из круглых проводников с конечной землёй

Рассмотрим особенности создания круглых проводников на примере двух проводной линии, где каждый проводник имеет свой слой изоляции с относительной диэлектрической проницаемостью 3.5, а один из них заземлен. Для создания заземленного проводник используется команд CONDUCTOR_GROUNDED. При этом следует помнить, что заземлённый проводник должен создаваться после всех незаземлённых.

Для сегментации всех интервалов структуры используем 16 подынтервалов, поэтому автосегментацию требуется отключить. Поскольку в данном примере будет только один заземленный проводник, то также требуется отключить и опцию бесконечной плоскости земли. Для описания геометрии проводников используем переменные x1, y1 и x2, y2 – центры первого и второго проводников соответственно, a – радиусы обоих проводников, b – радиусы их диэлектрических оболочек, листинг 5.4.

Листинг 5.4 – Подготовка к созданию конфигурации с конечной землёй

```
SET_INFINITE_GROUND 0
SET_AUTO_SEGMENT_LENGTH 0.
SET_SUBINTERVALS 16
SET "Er" 3.5
SET "x1" 0.
SET "y1" 0.
SET "x2" 1.27e-3
SET "y2" 0.
SET "a" 0.19e-3
SET "b" 0.44e-3
```

Сначала создается незаземленный проводник, заключенный в оболочку с Er = 3.5, с помощью команды CIRCLE. Её первые два параметра – это координаты центра окружности, а третий – радиус окружности. Обход окружности всегда происходит против часовой

стрелки, поэтому ER_PLUS и MU_PLUS всегда снаружи окружности, а ER_MINUS и MU_MINUS всегда внутри окружности. Затем аналогичным образом создается заземленный проводник, листинг 5.5.

Листинг 5.5 – Создание проводников для конфигурации с конечной землёй

```
CONDUCTOR
   SET_ER_PLUS 3.5
   CIRCLE x2 y2 a
CONDUCTOR_GROUNDED
   CIRCLE x1 y1 a
```

После создания проводников необходимо задать диэлектрические границы. Здесь следует помнить, что обход окружностей осуществляется против часовой стрелки, поэтому справа от границы окружности воздух с Er = 1 (SET_ER_PLUS 1.), а слева диэлектрик с Er = 3.5 (SET_ER_MINUS Er). Так как эти значения одинаковы для обоих проводников, то повторно их задавать не требуется. После задания конфигурации её требуется сохранить в переменной, например с именем conf_fg, и тогда её можно визуализировать, листинг 5.6.

Листинг 5.6 – Создание диэлектрика для конфигурации с конечной землёй

```
DIELECTRIC
   SET_ER_PLUS 1.
   SET_ER_MINUS Er
  CIRCLE x1 y1 b
DIELECTRIC
  CIRCLE x2 y2 b
SET "conf_fg" GET_CONFIGURATION_2D
DRAW_CONFIGURATION conf_fg
```

Результат выполнения листингов 5.4, 5.5 и 5.6 представлен на рисунке 5.7.

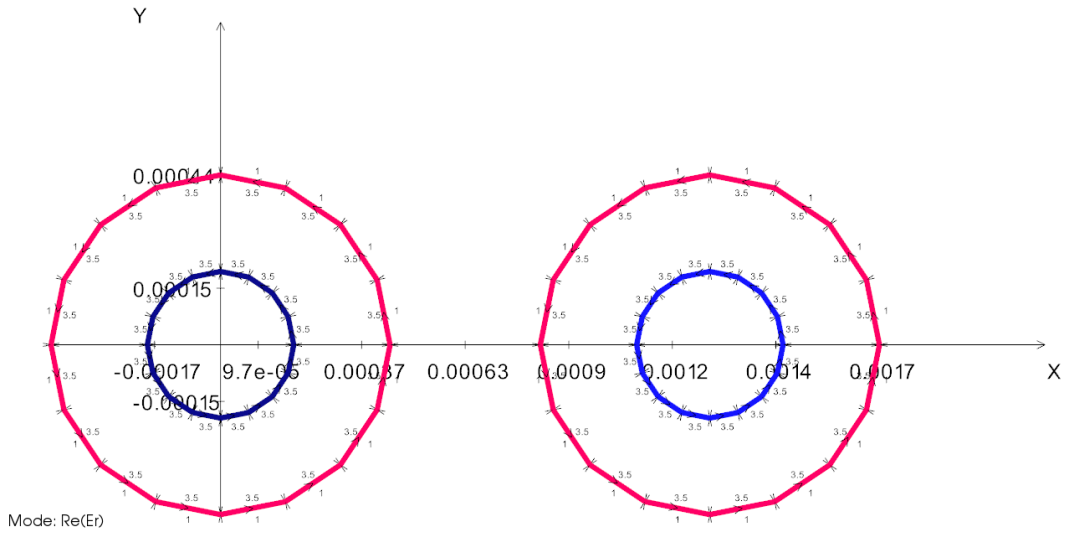


Рисунок 5.7 – Конфигурация с конечной землёй

5.1.1.5 Примеры создания сложных конфигураций

На основе представленного инструментария, могут быть созданы конфигурации повышенной сложности, где границы проводников и диэлектриков могут иметь любую ориентацию. Кроме того, для сегментации могут быть совместно использованы ручная и автоматическая сегментации. Так, на рисунках 5.8–5.14 приведены поперечные сечения созданных конфигураций повышенной сложности.

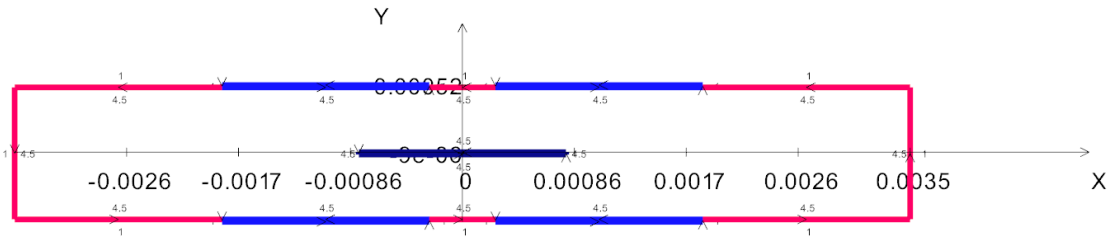


Рисунок 5.8 – Конфигурация с исполнением опорного проводника в центре

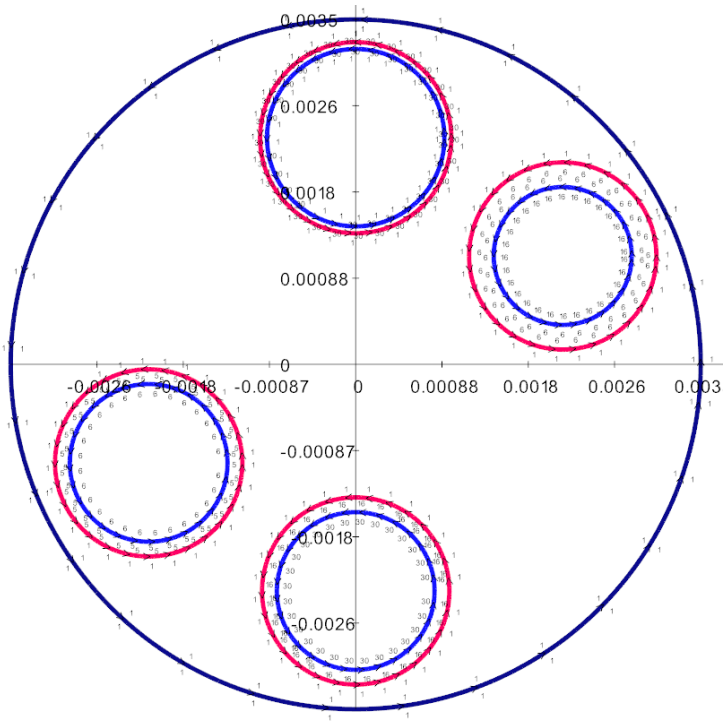


Рисунок 5.9 – Конфигурация с исполнением опорного проводника вокруг

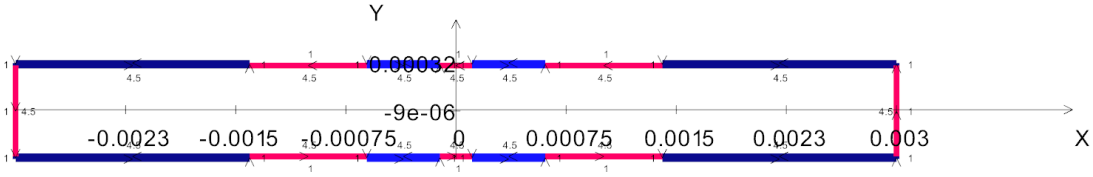


Рисунок 5.10 – Конфигурация с исполнением опорного проводника в виде боковых полигонов

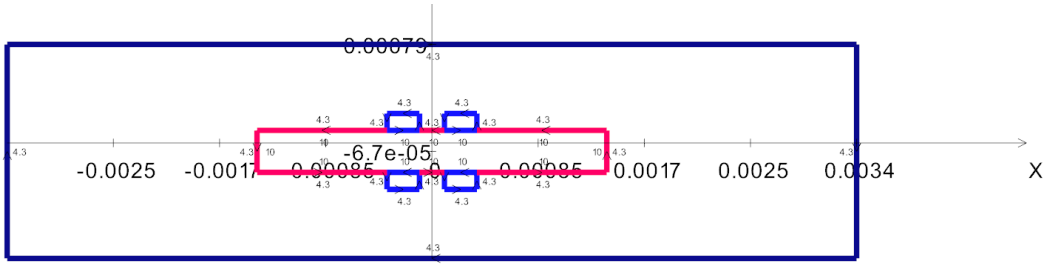


Рисунок 5.11 – Конфигурация с исполнением опорного проводника сверху и снизу

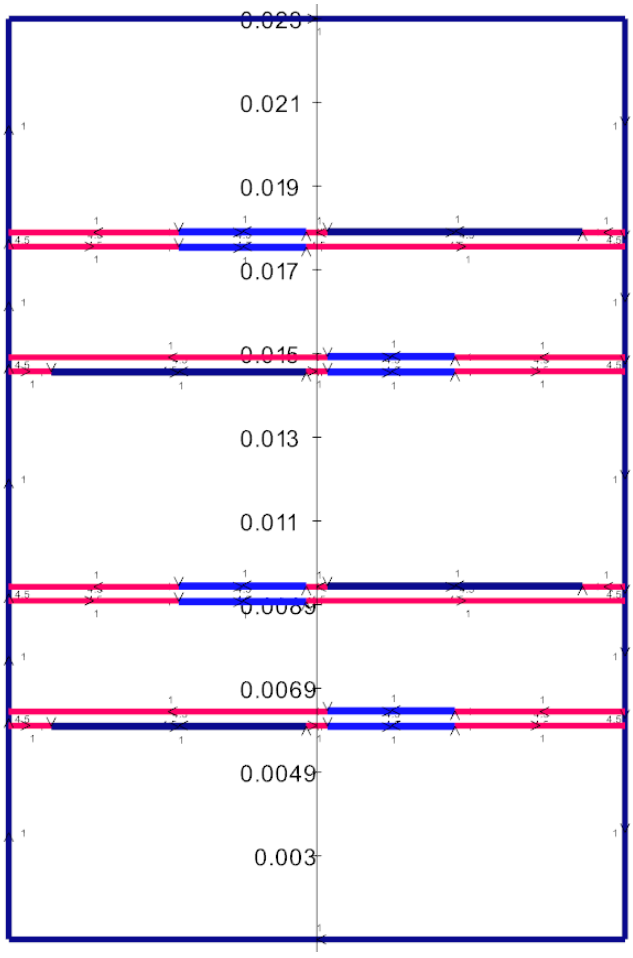


Рисунок 5.12 – Конфигурация с каскадированием

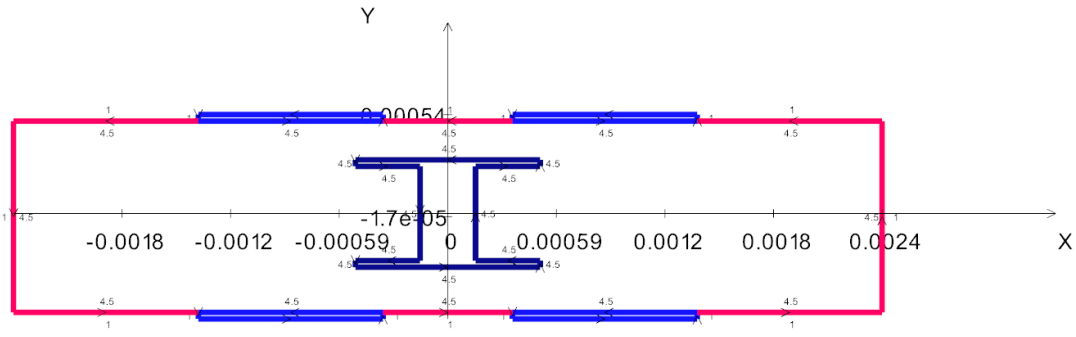


Рисунок 5.13 – Конфигурация с четырьмя проводящими слоями с перемычками

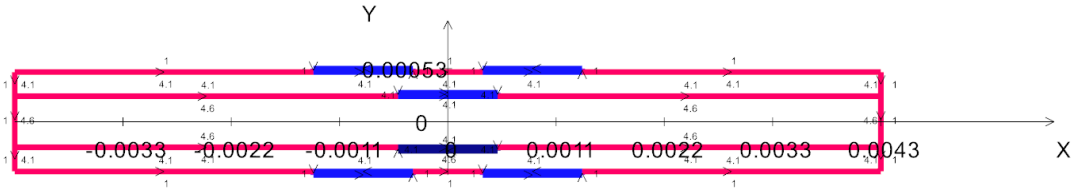


Рисунок 5.14 – Конфигурация с четырьмя проводящими слоями без перемычек

5.1.1.6 Вычисление матриц C, C0, L, L0

Для вычисления матрицы **C** необходимо сначала сформировать матрицу СЛАУ – SMN следующим образом: «SET "smn" SMN_C conf_ig». Затем решить СЛАУ и вычислить матрицу **C** – «SET "matC" CALCULATE_C smn conf_ig» или вывести её на печать – «ECHO CALCULATE_C smn conf_ig». Также доступна комбинация команд: «ECHO CALCULATE_C SMN_C conf_ig conf_ig». Аналогично вычисляются матрицы **C0**, **L** и **L0**. При этом команда SMN_C заменяется на команды SMN_C0, SMN_L и SMN_L0 соответственно, а команда CALCULATE_C – на CALCULATE_C0, CALCULATE_L и CALCULATE_L0 соответственно, листинг 5.7. Результат использования листингов 5.2 и 5.7 приведен на рисунке 5.15. Стоит отметить, что в рассматриваемой конфигурации не были заданы магнитные проницаемости материалов (MU_PLUS и MU_MINUS), поэтому матрицы **L** и **L0** имеют одинаковые значения.

Листинг 5.7 – Вычисление матриц **C**, **C0**, **L** и **L0**

```
ECHO infinite ground configuration
ECHO Cmn
SET "smn" SMN_C conf_ig
ECHO CALCULATE_C smn conf_ig
ECHO C0mn
SET "smn" SMN_C0 conf_ig
ECHO CALCULATE_C0 smn conf_ig
ECHO Lmn
SET "smn" SMN_L conf_ig
ECHO CALCULATE_L smn conf_ig
ECHO L0mn
SET "smn" SMN_L0 conf_ig
ECHO CALCULATE_L0 smn conf_ig
```

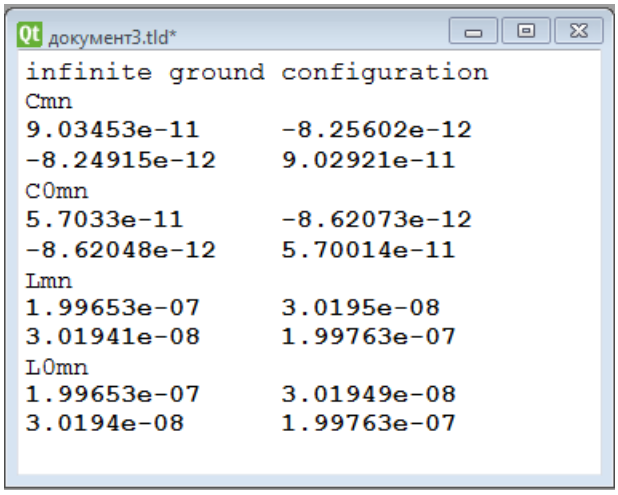


Рисунок 5.15 – Результат выполнения скрипта из листингов 5.2 и 5.7

5.1.1.7 Вычисление матрицы G

Матрица **G** предназначена для учет потерь в диэлектрике и соответственно вычисляется на определенной частоте. Поэтому для её используется та же математическая модель, что и для матрицы **C**, с той лишь разницей, что действительная относительная диэлектрическая проницаемость заменяется на комплексную с использованием тангенса угла диэлектрических потерь. Задание тангенса диэлектрических потерь (команды SET_TAN_DELTA_PLUS и SET_TAN_DELTA_MINUS) осуществляется аналогично заданию относительной диэлектрической проницаемости диэлектрика (команды SET_ER_PLUS и SET_ER_MINUS). При этом задание тангенса диэлектрических потерь должно осуществляться только после задания относительных диэлектрических проницаемостей.

Для вычисления матрицы **G**, по аналогии с вычислением матрицы **C**, используются команды SMN_CG и CALCULATE_CG. Однако у команды CALCULATE_CG добавлен второй параметр – частота (Гц) на которой выполняется вычисление. Результатом выполнения этих двух команд является комплексная матрицы, у которой реальная часть – это матрица **C**, а мнимая – матрица **G**. Поэтому, если при создании конфигурации не задан тангенс угла диэлектрических потерь используемых материалов, то результирующая матрица имеет нулевую мнимую часть.

Для примера используем листинг 5.2. Пусть на частоте 1 МГц тангенс угла диэлектрических потерь используемого материала подложки равен 0.025. Итоговый скрипт с добавленными командами SET_TAN_DELTA_MINUS, SET_TAN_DELTA_PLUS, SMN_CG и CALCULATE_CG приведен в листинге 5.8, а результат его выполнения на рисунке 5.16.

Листинг 5.8 – Создание конфигурации с бесконечной плоскостью земли и вычисление матриц **C** и **G**

```
INCLUDE "UTIL"
INCLUDE "MATRIX"
INCLUDE "MOM2D"
SET "Er" 2.
SET "td" 0.025
SET_AUTO_SEGMENT_LENGTH 0.0
CREATE_KEYWORD "create_conf"
SET_INFINITE_GROUND 1
SET_SUBINTERVALS 4
CONDUCTOR
   SET_ER_PLUS Er
   SET_TAN_DELTA_PLUS td
   LINE 2. 1. 5. 1.
   SET_ER_PLUS 1.
   SET_TAN_DELTA_PLUS 0.
   LINETO 5. 2.
   SET_SUBINTERVALS 8
   LINETO 2. 2.
   SET_SUBINTERVALS 4
   LINETO 2. 1.
CONDUCTOR
   SET_ER_PLUS Er
   SET_TAN_DELTA_PLUS td
   LINE 7. 1. 10. 1.
   SET_ER_PLUS 1.
   SET_TAN_DELTA_PLUS 0.
   LINETO 10. 2.
   LINETO 7. 2.
   LINETO 7. 1.
DIELECTRIC
   SET_ER_PLUS 1.
   SET_ER_MINUS Er
   SET_TAN_DELTA_MINUS td
   SET_TAN_DELTA_PLUS 0.
   LINE 12. 0. 12. 1.
   LINETO 10. 1.
   LINE 7. 1. 5. 1.
   LINE 2. 1. 0. 1.
   LINETO 0. 0.
SET "conf_ig" GET_CONFIGURATION_2D
DRAW_CONFIGURATION conf_ig
END_CREATE_KEYWORD
create_conf

SET "smn" SMN_CG conf_ig
ECHO CALCULATE_CG smn conf_ig 1e+6
```

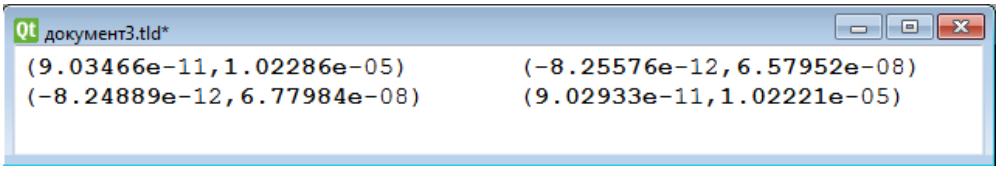


Рисунок 5.16 – Результат выполнения скрипта из листинга 5.8

5.1.1.8 Симметрирование матриц и вычисление матрицы «ошибок»

Известно, что при построении, прежде всего, асимметричных конфигураций асимметрия сетки и погрешность вычислений, обусловленная конечным представление чисел на компьютере, могут приводить к асимметрии матриц погонных параметров, которые по теории должны быть симметричными. Поэтому в системе реализована команда MATRIX_SYMMETRISATION для симметрирования матриц (как среднее арифметическое) относительно главной диагонали. При необходимости оценку асимметрии можно получить с помощью команды MATRIX_ERROR в виде отношения модуля разности элементов матрицы к модулю их суммы, выраженную в процентах. В листинге 5.9 приведен пример использования этих команд на примере матриц **C** и **L**, а результат его выполнения после листинга 5.2 приведен на рисунке 5.17.

Листинг 5.9 – Пример симметрирования матриц **C** и **L** и оценки их асимметрии

```
SET "mC" CALCULATE_C SMN_C conf_ig conf_ig
SET "mL" CALCULATE_L SMN_L conf_ig conf_ig
ECHO LINE_TO_STRING //Матрица C
ECHO mC
SET "mC_s" MATRIX_SYMMETRISATION mC
ECHO LINE_TO_STRING //Симметрированная матрица C
ECHO mC_s
SET "mC_e" MATRIX_ERROR mC_s mC
ECHO LINE_TO_STRING //Матрица «ошибок» для матрицы C
ECHO mC_e
SET "mC_e_max" MATRIX_MAX ABS mC_e
ECHO LINE_TO_STRING //Максимальное значение матрицы «ошибок» для матрицы C
ECHO mC_e_max
ECHO LINE_TO_STRING //Матрица L
ECHO mL
SET "mL_s" MATRIX_SYMMETRISATION mL
ECHO LINE_TO_STRING //Симметрированная матрица L
ECHO mL_s
SET "mL_e" MATRIX_ERROR mL_s mL
ECHO LINE_TO_STRING //Матрица «ошибок» для матрицы L
ECHO mL_e
SET "mL_e_max" MATRIX_MAX ABS mL_e
ECHO LINE_TO_STRING //Максимальное значение матрицы «ошибок» для матрицы L
ECHO mL_e_max
```

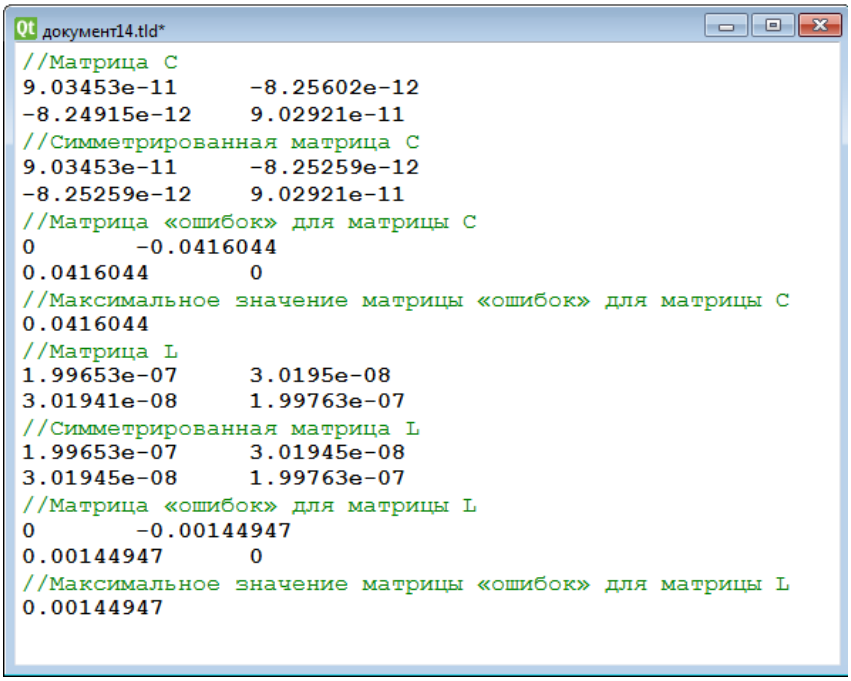


Рисунок 5.17 – Результат выполнения листинга 5.9

5.1.1.9 Вычисление ёмкостной матрицы в диапазоне параметров

Задача вычисления ёмкостной матрицы **C** системы МПЛП сводится к СЛАУ с квадратной матрицей размера $N \times N$, связывающей плотности заряда элементов дискретизации на проводниках и диэлектрических границах, с потенциалами этих элементов. Это уравнение решается N_{COND} раз (N_{COND} – число проводников в системе, не считая опорного), причём в i-м решении, потенциал проводника ($i = 1, \ldots, N_{COND}$), равен единице, а потенциал всех остальных проводников равен нулю.

На практике достаточно часто требуется вычисление матрицы **C** в диапазоне параметров материалов, например, вычисление временного отклика с учетом частотной зависимости параметров материала FR-4, когда для каждой частоты из спектра воздействующего сигнала изменяется значение ε_r и для каждого из них заново вычисляется матрица **C**. Другой пример, многократное вычисление по диапазону параметров диэлектрика или оптимизация по параметрам диэлектрика. Для таких случаев, в системе реализовано метод, основанный на блочном LU-разложении матрицы.

В системе реализованы две команды для реализации вычисления емкостной матрицы, SMN_C и CALCULATE_C. Дополнительно была реализована еще одна команда SMN_C_UPDATE для вычисления матрицы **S** при изменении значения ε_r. Аналогчно применимы команды SMN_L_UPDATE, SMN_C0_UPDATE, SMN_L0_UPDATE и SMN_CG_UPDATE.

Для возможности использования блочного представления матрицы СЛАУ реализована команда SET_MOM2D_BLOCK_SOLVE. При передаче ей в качестве параметра значения 0 будет использован блочный алгоритм, а при передаче 1 – последовательный алгоритм пересчета. Для того, чтобы узнать какое значение установлено в данный момент, реализована команда GET_MOM2D_BLOCK_SOLVE, по возвращаемому значению которой и определяется установленный в данный момент режим. Пример использования этих команд приведен в листинге 5.10, используемом в дополнение к листингу 5.2.

Листинг 5.10 – Использование блочного LU-разложения

```
INCLUDE "UTIL"
INCLUDE "MATRIX"
INCLUDE "MOM2D"
SET_ER_PLUS 1.
SET_MU_PLUS 1.
SET "Er" 2.
SET_MOM2D_BLOCK_SOLVE 1
CREATE_KEYWORD "calc"
SET_INFINITE_GROUND 1
SET_AUTO_SEGMENT_LENGTH 0.
SET_SUBINTERVALS 4
CONDUCTOR
   SET_ER_PLUS Er
   LINE 2. 1. 5. 1.
   SET_ER_PLUS 1.
   LINETO 5. 2.
   SET_SUBINTERVALS 8
   LINETO 2. 2.
   SET_SUBINTERVALS 4
   LINETO 2. 1.
CONDUCTOR
   SET_ER_PLUS Er
   LINE 7. 1. 10. 1.
   SET_ER_PLUS 1.
   LINETO 10. 2.
   LINETO 7. 2.
   LINETO 7. 1.
DIELECTRIC
   SET_ER_PLUS 1.
   SET_ER_MINUS Er
   LINE 12. 0. 12. 1.
   LINETO 10. 1.
   LINE 7. 1. 5. 1.
   LINE 2. 1. 0. 1.
   LINETO 0. 0.
SET "conf_ig" GET_CONFIGURATION_2D
DRAW_CONFIGURATION conf_ig
END_CREATE_KEYWORD
calc
```

```
ECHO Er
ECHO Cmn_
SET "smn" SMN_C conf_ig
ECHO TO_STRING CALCULATE_C smn conf_ig
SET "Er" 4.
calc
ECHO Er
ECHO Cmn_
SET "smn" SMN_C_UPDATE conf_ig smn
ECHO TO_STRING CALCULATE_C smn conf_ig
```

5.1.1.10 Вычисление коэффициентов KC, KL

Вычислим матрицы C и L0 для нашей конфигурации и сохраним их в переменные cm и lm. Затем по формулам $KC = -C_{12}/C_{11}$ и $KL = L_{12}/L_{11}$ найдём коэффициенты KC и KL (см. листинг 5.11 и рисунок 5.18).

Листинг 5.11 – Вычисление коэффициентов KC и KL

```
SET "cm" CALCULATE_C SMN_C conf_ig conf_ig
SET "lm" CALCULATE_L0 SMN_L0 conf_ig conf_ig
SET "kc" DIV MINUS 0. GET_MATRIX_VALUE cm 0 1
GET_MATRIX_VALUE cm 0 0
ECHO FORMAT_STRING nstd kc= kc
SET "kl" DIV GET_MATRIX_VALUE lm 0 1
GET_MATRIX_VALUE lm 0 0
ECHO FORMAT_STRING nstd kl= kl
```

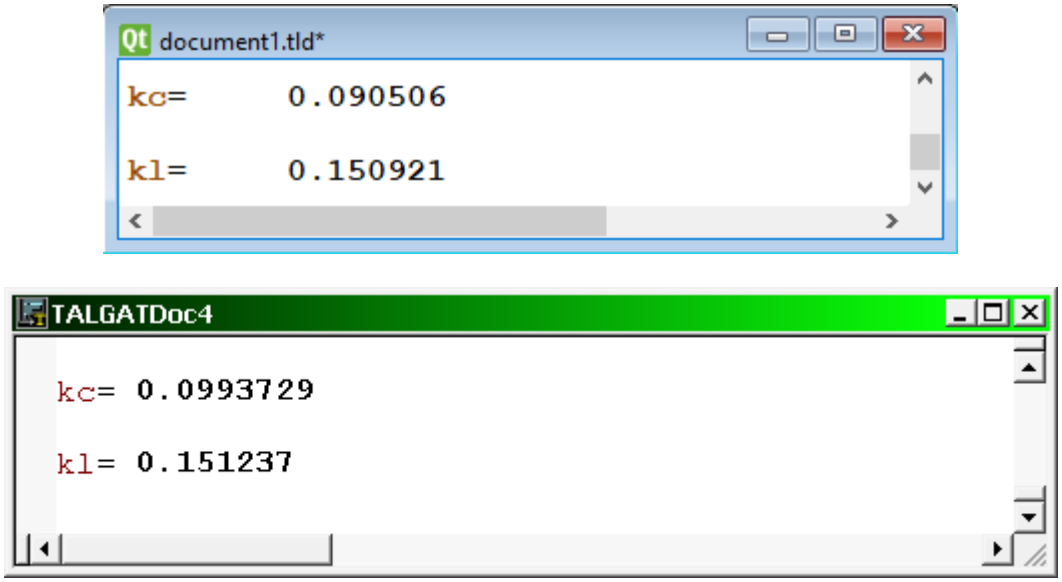


Рисунок 5.18 – Результат выполнения скрипта из листинга 5.11

5.1.1.11 Вычисление распределения плотности зарядов mQ и его визуализация

Для вычисления плотности распределения зарядов mQ вместо команды CALCULATE_C используется команда CALCULATE_CQ, которая при вычислении матрицы **C** дополнительно собирает информацию о плотности заряда. Плотность заряда может быть привязана к конфигурации. Различные граничные участки конфигурации будут иметь цвет в зависимости от плотности заряда. Значение, соответствующие цвету, отображается на цветовой шкале.

Для визуализации распределения плотности заряда необходимо вместо команды DRAW_CONF2D вызывать команду DRAW_CONF2DQ, передавая в качестве аргумента команды матрицу mQ, полученную в результате выполнения команды CALCULATE_CQ.

Пример визуализации распределения зарядов показан на рисунке 5.19.

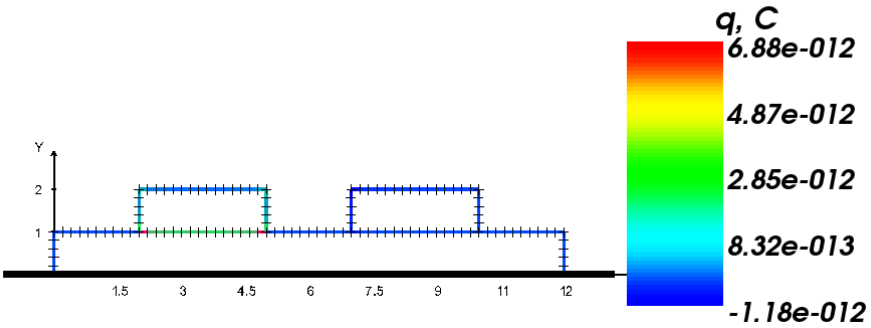


Рисунок 5.19 – Визуализация плотности заряда

5.1.1.12 Графический редактор конфигураций

Графический редактор (рисунок 5.20) конфигураций позволяет указывать границы проводников и диэлектриков, задавать их параметры. Доступны элементы для рисования прямых линий, прямоугольников, окружностей.

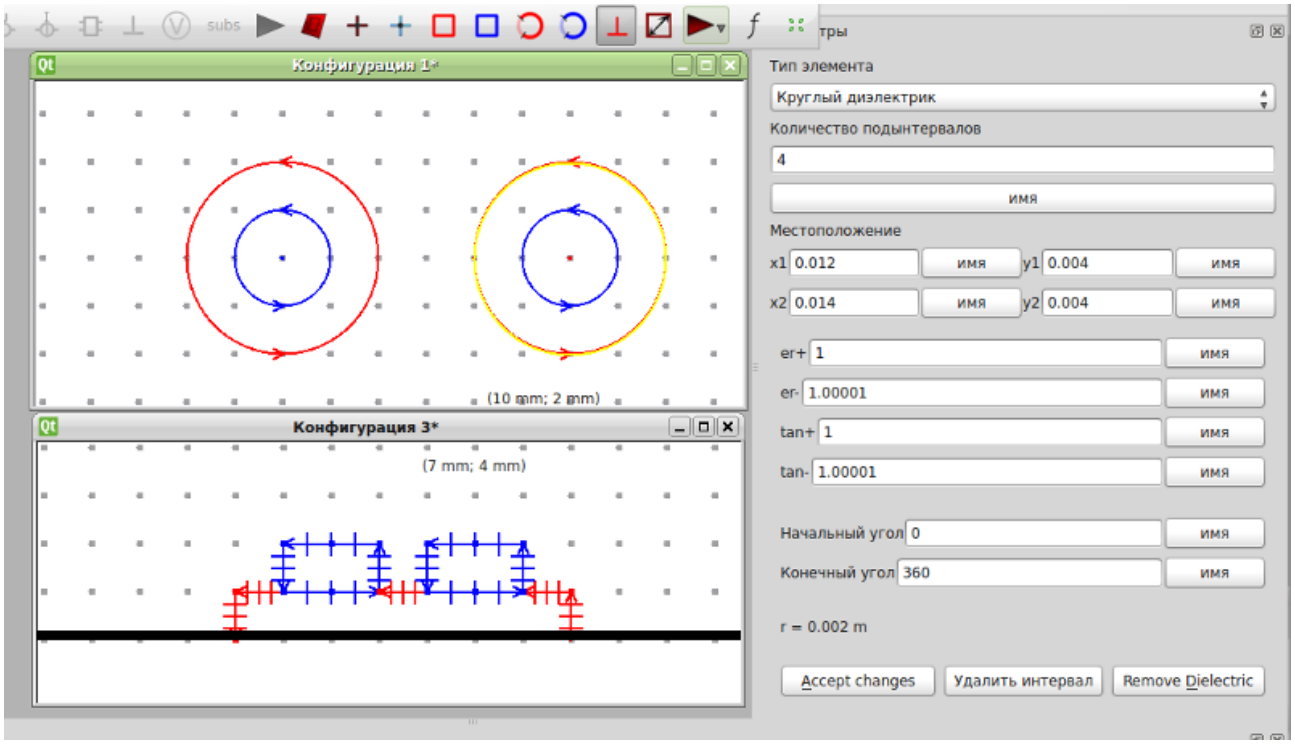


Рисунок 5.20 – Вид графического редактора

Полученная конфигурация может быть сохранена одном из двух форматов: сценарий TUSUR.EMC, описывающий границы конфигурации; сценарий TUSUR.EMC, описывающий печатную плату.

5.1.1.13 Послойный ввод MOM2D-конфигураций

Могут использоваться команды для простого ввода MOM2D-конфигураций поперечного сечения печатных плат

SET_ENVIRONMENT eps_r tan_d Устанавливает диэлектрическую проницаемость eps_r и тангенс угла диэлектрических потерь tan_d среды, в которой находится печатная плата (по умолчанию 1.0 и 0.0 соответственно).

LAYER height eps_r tan_d Добавляет слой высотой height с диэлектрическими параметрами eps_r и tan_d

COND space [TO_EDGE,TO_CENTER] width thickness [depth] Добавляет на текущий слой прямоугольный проводник. Параметры: space - расстояние от края платы (если это первый проводник на слое) или от соседнего проводника до левого края проводника (опционный параметр TO_EDGE, установлен по умолчанию) или до центра проводника (TO_CENTER). Значение space первого проводника также используется для создания крайней правой диэлектрической границы текущего слоя. width - ширина проводника

thickness - толщина проводника depth - (опционный параметр) глубина погружения проводника в слой. По умолчанию 0.0, т.е. проводник находится на поверхности слоя

COVER height eps_r tan_d Аналогична команде LAYER, за исключением того, что после её использования нельзя добавлять новые слои командой LAYER. Иммитирует покрытие лаком или маской, добавляет слой диэлектрика высотой height , огибающий выступающие проводники.

При выполнении команды GET_CONFIGURATION_2D строятся диэлектрические и проводниковые границы на основе данных, заданных с помощью команд LAYER, COND и COVER.

5.1.2 Модуль трёхмерного электростатического анализа MOM3D

5.1.2.1 Общие сведения

Модуль MOM3D предназначен для электростатического анализа трёхмерных конфигураций проводников и диэлектриков и позволяет вычислить матрицу коэффициентов электростатической индукции с заданным диэлектрическим заполнением – C (Ф).

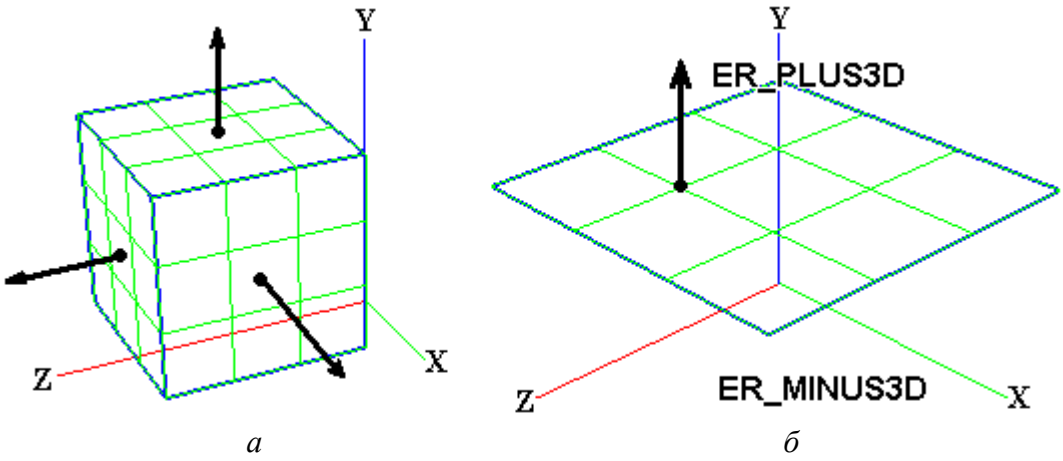
В основном, работа с модулями MOM2D и MOM3D аналогична, так как они являются двумя частными случаями одного и того же подхода к выполнению электростатического анализа заданных пользователем конфигураций. Перед использованием модуля трёхмерного электростатического анализа его необходимо сначала загрузить (листинг 5.12).

Листинг 5.12 – Подключение модулей, необходимых для трёхмерного электростатического анализа

```
INCLUDE "UTIL"
INCLUDE "MATRIX"
INCLUDE "MOM3D"
```

Обратите внимание на правила задания ER и MU. Для любой диэлектрической площадки должны быть заданы два значения ER - ER_PLUS3D и ER_MINUS3D. Для этого условно зададим на площадке положительную и отрицательную стороны: мысленно построим единичный вектор для оси, которой перпендикулярна данная площадка, так, чтобы он пересекал данную площадку. Отсюда положительная сторона площадки – та, с которой находится конец единичного вектора (ей соответствует ER_PLUS3D), отрицательная сторона площадки – та, с которой находится начало единичного вектора (ей соответствует ER_MINUS3D). Для любой проводниковой площадки имеет значение только ER_PLUS3D, которая автоматически распределяется на положительную или отрицательную сторону площадки в зависимости от того, с какой стороны прилегает диэлектрик.

В данной версии системы площадки трехмерных конфигураций могут быть по форме только прямоугольными, а по ориентации – только ортогональными осям декартовых координат. Для каждой оси координат есть единичные векторы - они направлены в сторону возрастания значений координат по этой оси и лежат на ней (рисунок 5.21, а). Например, для оси Y один из возможных единичных векторов (вектор Oy) выходит из точки (0,0,0) и кончается в точке (0,1,0). Если мы построим любой вектор той же длины, который параллелен оси (а значит, и единичному вектору Oy), то его тоже можно назвать единичным.



а *б*

Рисунок 5.21 – Правила задания ER и MU: диэлектрические плоскости с единичными векторами (*а*); распределение ER_PLUS3D и ER_MINUS3D относительно плоскости с единичным вектором (*б*)

Так как в модуле MOM3D все плоскости, которые возможно построить, перпендикулярны осям координат, то для каждой из этих плоскостей можно построить единичный вектор, который будет начинаться в точке, принадлежащей этой плоскости. Плоскость делит пространство на два полупространства (рисунок 5.21, б). При этом то полупространство, в котором находится единичный вектор, будет положительным (ему соответствует ER_PLUS3D), а другое полупространство будет отрицательным (ему соответствует ER_MINUS3D).

При моделировании трёхмерных конфигураций полезно знать, что ось X – зелёная, Y – синяя, Z – красная. При этом по умолчанию ось Y направлена вверх.

5.1.2.2 Создание конфигурации с бесконечной землёй

Команды SET_ER_PLUS3D и SET_ER_MINUS3D устанавливают соответствующие диэлектрические проницаемости.

Команда SET_INFINITE_GROUND3D с параметром 1 устанавливает бесконечную землю, с параметром 0 — конечную.

Создание проводника начинается после команды CONDUCTOR3D. Команда DIELECTRIC3D начинает создание диэлектрика. Команды SET_SUBINTERVALS_X и SET_SUBINTERVALS_Y устанавливают число подынтервалов для сегментации прямоугольников в двух направлениях.

Команды RECT_XY, RECT_XZ и RECT_ZY создают прямоугольники в соответствующих плоскостях. Первый параметр — расстояние от плоскости, которой параллелен создаваемый прямоугольник, до самого прямоугольника. Второй и третий параметры — координаты левого нижнего угла прямоугольника, четвёртый и пятый параметры — координаты правого верхнего угла.

Команда GET_CONFIGURATION_3D возвращает созданную трёхмерную конфигурацию.

Создадим конфигурацию, показанную на рисунке 5.22, в. Обратите внимание, что площадки не накладываются друг на друга, то есть в диэлектрике на месте соприкосновения с металлом не создаётся никаких площадок, так как они уже созданы. Скрипт для создания этой конфигурации показан в листинге 5.13, а результат его работы – на рисунке 5.23.

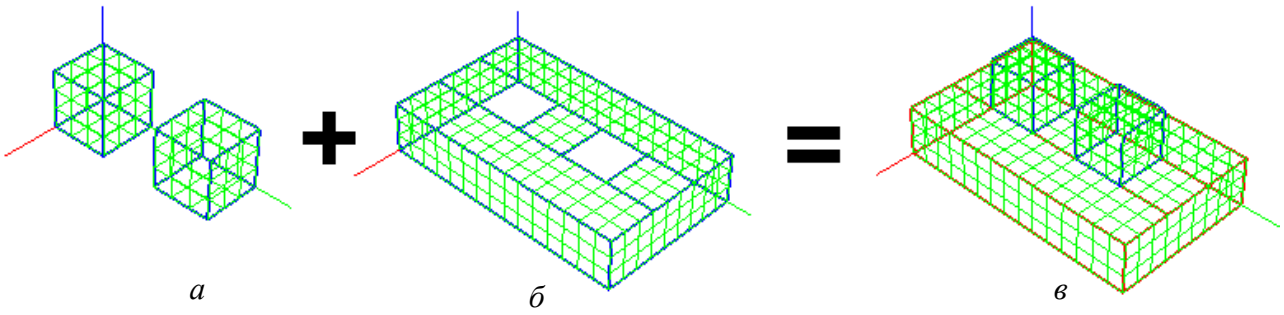


Рисунок 5.22 – Получение конфигурации с бесконечной землёй: проводники (*а*); диэлектрик (*б*); конфигурация (*в*)

Листинг 5.13 – Создание трёхмерной конфигурации с бесконечной землёй

```
CONDUCTOR3D
   SET_SUBINTERVALS_X 3
   SET_SUBINTERVALS_Y 3
   SET_ER_PLUS3D 10.
   RECT_XZ 1. 1. 1. 2. 2.
   SET_ER_PLUS3D 1.
   RECT_XZ 2. 1. 1. 2. 2.
   RECT_XY 1. 1. 1. 2. 2.
   RECT_XY 2. 1. 1. 2. 2.
   RECT_ZY 1. 1. 1. 2. 2.
```

```
DIELECTRIC3D
   SET_SUBINTERVALS_X 15
   SET_SUBINTERVALS_Y 3
   SET_ER_PLUS3D 1.
   SET_ER_MINUS3D 10.
   RECT_XZ 1. 0. 0. 5. 1.
   RECT_XZ 1. 0. 2. 5. 3.
   SET_SUBINTERVALS_X 3
   RECT_XZ 1. 0. 1. 1. 2.
   RECT_XZ 1. 2. 1. 3. 2.
```

```
    RECT_ZY 2. 1. 1. 2. 2.
 CONDUCTOR3D
    SET_SUBINTERVALS_X 3
    SET_SUBINTERVALS_Y 3
    SET_ER_PLUS3D 10.
    RECT_XZ 1. 3. 1. 4. 2.
    SET_ER_PLUS3D 1.
    RECT_XZ 2. 3. 1. 4. 2.
    RECT_XY 1. 3. 1. 4. 2.
    RECT_XY 2. 3. 1. 4. 2.
    RECT_ZY 3. 1. 1. 2. 2.
    RECT_ZY 4. 1. 1. 2. 2.
    RECT_XZ 1. 4. 1. 5. 2.
    SET_ER_PLUS3D 10.
    SET_ER_MINUS3D 1.
    SET_SUBINTERVALS_X 15
    RECT_XY 0. 0. 0. 5. 1.
    SET_ER_PLUS3D 1.
    SET_ER_MINUS3D 10.
    RECT_XY 3. 0. 0. 5. 1.
    SET_ER_PLUS3D 10.
    SET_ER_MINUS3D 1.
    SET_SUBINTERVALS_X 9
    RECT_ZY 0. 0. 0. 3. 1.
    SET_ER_PLUS3D 1.
    SET_ER_MINUS3D 10.
    RECT_ZY 5. 0. 0. 3. 1.
 SET "conf_ig" GET_CONFIGURATION_3D
 DRAW_CONFIGURATION3D conf_ig
```

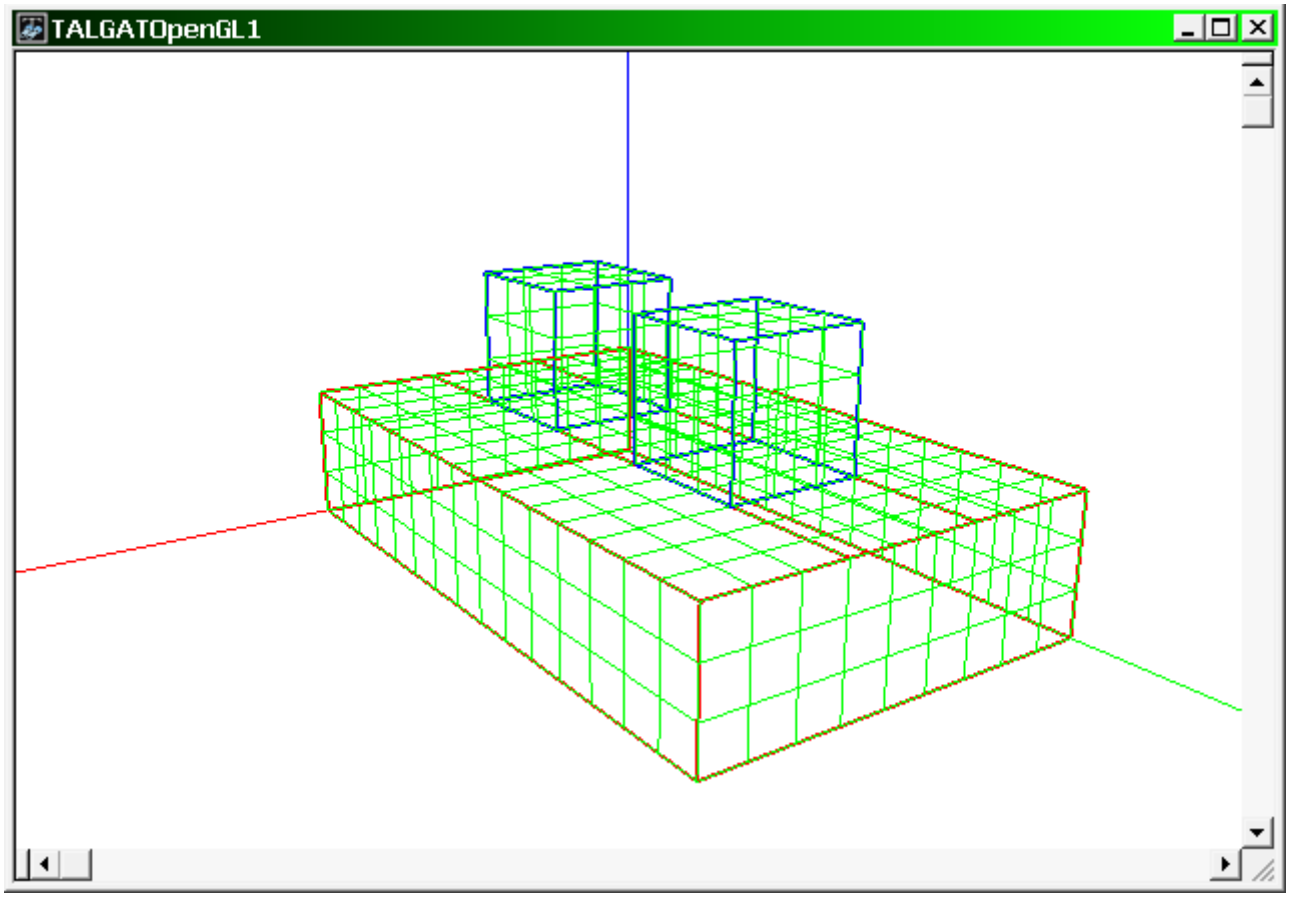


Рисунок 5.23 – Результат выполнения скрипта из листинга 5.13

5.1.2.3 Создание конфигурации с конечной землёй

Команда CONDUCTOR3D_GROUNDED начинает создание заземлённого проводника. Обратите внимание: заземлённый проводник должен создаваться после всех незаземлённых.

Создадим такую же конфигурацию из разд. 5.1.2.2 с заземлённым проводником (листинг 5.14 – Создание трёхмерной конфигурации с конечной землёй). Выглядеть она будет так же, как и в предыдущем примере (рисунке 5.23).

Листинг 5.14 – Создание трёхмерной конфигурации с конечной землёй

```
SET_INFINITE_GROUND3D 0

CONDUCTOR3D
   SET_SUBINTERVALS_X 3
   SET_SUBINTERVALS_Y 3
   SET_ER_PLUS3D 10.
   RECT_XZ 1. 1. 1. 2. 2.
   SET_ER_PLUS3D 1.
   RECT_XZ 2. 1. 1. 2. 2.
   RECT_XY 1. 1. 1. 2. 2.
   RECT_XY 2. 1. 1. 2. 2.
   RECT_ZY 1. 1. 1. 2. 2.
   RECT_ZY 2. 1. 1. 2. 2.

CONDUCTOR3D_GROUNDED
   SET_SUBINTERVALS_X 3
   SET_SUBINTERVALS_Y 3
   SET_ER_PLUS3D 10.
   RECT_XZ 1. 3. 1. 4. 2.
   SET_ER_PLUS3D 1.
   RECT_XZ 2. 3. 1. 4. 2.
   RECT_XY 1. 3. 1. 4. 2.
   RECT_XY 2. 3. 1. 4. 2.
   RECT_ZY 3. 1. 1. 2. 2.
   RECT_ZY 4. 1. 1. 2. 2.

DIELECTRIC3D
   SET_SUBINTERVALS_X 15
   SET_SUBINTERVALS_Y 3
   SET_ER_PLUS3D 1.
   SET_ER_MINUS3D 10.
   RECT_XZ 1. 0. 0. 5. 1.
   RECT_XZ 1. 0. 2. 5. 3.
   SET_SUBINTERVALS_X 3
   RECT_XZ 1. 0. 1. 1. 2.
   RECT_XZ 1. 2. 1. 3. 2.
   RECT_XZ 1. 4. 1. 5. 2.
   SET_ER_PLUS3D 10.
   SET_ER_MINUS3D 1.
   SET_SUBINTERVALS_X 15
   RECT_XY 0. 0. 0. 5. 1.
   SET_ER_PLUS3D 1.
   SET_ER_MINUS3D 10.
   RECT_XY 3. 0. 0. 5. 1.
   SET_ER_PLUS3D 10.
   SET_ER_MINUS3D 1.
   SET_SUBINTERVALS_X 9
   RECT_ZY 0. 0. 0. 3. 1.
   SET_ER_PLUS3D 1.
   SET_ER_MINUS3D 10.
   RECT_ZY 5. 0. 0. 3. 1.

SET "conf_fg" GET_CONFIGURATION_3D
DRAW_CONFIGURATION3D conf_fg
```

5.1.2.4 Вычисление матрицы C

Команда SMN_C3D используется для расчета матрицы SMN для трехмерной конфигурации и требует один параметр – переменную с созданной трёхмерной конфигурацией.

Команда CALCULATE_C3D возвращает матрицу ёмкостей и требует два параметра. Первый параметр — матрица SMN, возвращаемая командой SMN_C3D, второй параметр — переменная с созданной трёхмерной конфигурацией (см. листинг 5.15, рисунок 5.24).

Листинг 5.15 – Вычисление матрицы C

```
ECHO FORMAT_STRING nttt***bsb***n LINE_TO_STRING infinite ground configuration
```

```
SET "smn" SMN_C3D conf_ig
ECHO TO_STRING CALCULATE_C3D smn conf_ig

ECHO FORMAT_STRING nttt***bsb***n LINE_TO_STRING finite ground configuration

SET "smn" SMN_C3D conf_fg
ECHO TO_STRING CALCULATE_C3D smn conf_fg
```

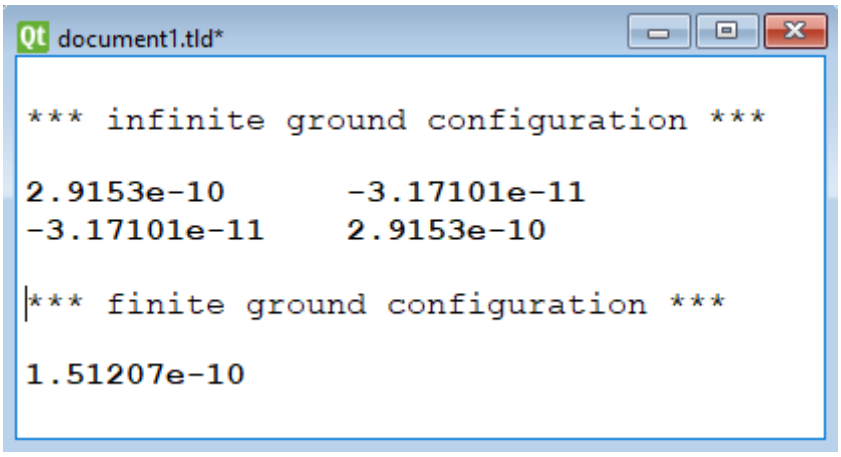


Рисунок 5.24 – Результат выполнения скрипта из листинга 5.15

5.1.2.5 Визуализация конфигурации

При вызове команды DRAW_CONFIGURATION3D происходит визуализации конфигурации. Диэлектрики обозначаются красным, проводники – синим (рисунок 5.25).

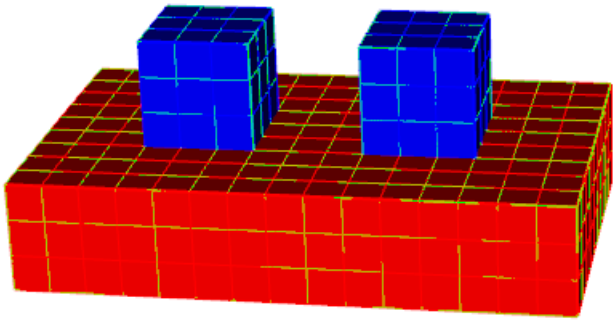


Рисунок 5.25 – Визуализация конфигурации

Поворот: левая кнопка мыши, либо Ctrl+левая кнопка мыши; Перемещение: правая кнопка мыши; Увеличение/уменьшение конфигурации: колесо прокрутки мыши, либо Ctrl+правая кнопка мыши;

Список горячих клавиш при визуализации конфигурации MOM2D:

- 'a': скрыть/показать оси координат;

- 'c': показать/скрыть цветовую шкалу плотности зарядов. Каждому цвету ставится в соответствие значение плотности заряда, этим цветом окрашивается соответствующий участок конфигурации;
- 'q': выключить/включить режим визуализации плотности заряда;
- 'r': сменить текущий проводник;
- 'f': сменить режим визуализации (2 режима:
- цветные поверхности;
- прозрачные поверхности, цветные линии);

Пример визуализации смотрите в DOC/MOM3D_ru.tld.

5.1.2.6 Вычисление распределения плотности зарядов mQ и его визуализация

Для вычисления плотности распределения зарядов mQ вместо команды CALCULATE_C3D используется команда CALCULATE_CQ3D, которая при вычислении матрицы C дополнительно собирает информацию о плотности заряда. Плотность заряда может быть привязана к конфигурации. Различные граничные участки конфигурации будут иметь цвет в зависимости от плотности заряда. Значение, соответствующие цвету, отображается на цветовой шкале.

Для визуализации распределения плотности заряда необходимо вместо команды DRAW_CONFIGURATION3D вызывать команду DRAW_CONFIGURATION3DQ, передавая в качестве аргумента команды матрицу mQ, полученную в результате выполнения команды CALCULATE_CQ3D. Пример визуализации распределения зарядов показан на рисунке 5.26.

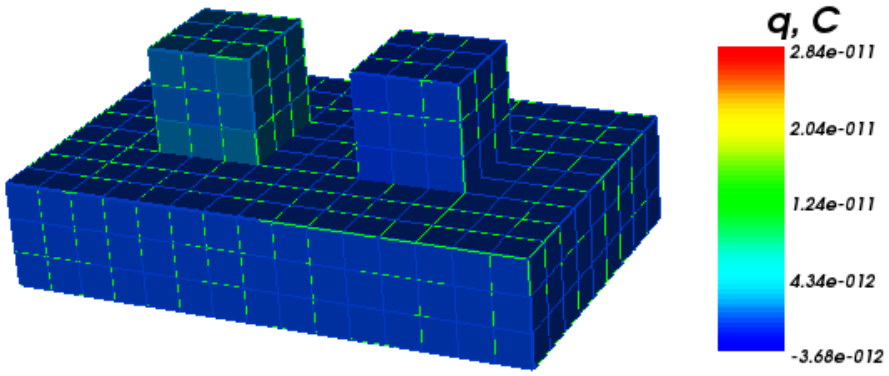


Рисунок 5.26 – Визуализация плотности заряда

5.1.2.7 Дополнительные команды

Для удобства при создании новых конфигураций в модуле MOM3D реализованы следующие команды.

Команда PARALLELEPIPED создает параллелепипед. Количество подсекций на каждой плоскости задается командами SET_SUBINTERVALS_X и SET_SUBINTERVALS_Y по принципу, аналогичному их использованию в командах RECT_**.

Параметры: первый, второй и третий — координаты X, Y, Z опорной точки; четвёртый, пятый и шестой - размеры параллелепипеда вдоль осей Ox, Oy, Oz соответственно.

Команда PARALLELEPIPED_SUBINTERVALS также создаёт параллелепипед, но количество подсекций на каждой плоскости задаётся параметрами команды. Параметры: первый, второй и третий — количество подсекций вдоль осей Ox, Oy, Oz соответственно; четвёртый, пятый и шестой — опорная координата, седьмой восьмой и девятый — размеры параллелепипеда.

Команды RECT_**_SIZE являются аналогами команд RECT_** за тем исключением, что четвёртым и пятым параметрами задаются размеры создаваемой плоскости.

Пример использования команд PARALLELEPIPED, PARALLELEPIPED_SUBINTERVALS и RECT_**_SIZE приведён в листинге 5.16 и на рисунок 5.27.

Листинг 5.16 – Пример использования дополнительных команд

```
CONDUCTOR3D
   SET_SUBINTERVALS_X 2
   SET_SUBINTERVALS_Y 1
   PARALLELEPIPED 0. 0. 0. 10. 10. 20.

CONDUCTOR3D
PARALLELEPIPED_SUBINTERVALS 1 2 3 0. 15. 0. 10. 10. 20.

CONDUCTOR3D
   SET_SUBINTERVALS_X 1
   SET_SUBINTERVALS_Y 2
   RECT_XY_SIZE 0. 15. 0. 10. 20.
   RECT_XZ_SIZE 0. 15. 0. 10. 20.
   SET_SUBINTERVALS_X 2
   SET_SUBINTERVALS_Y 2
   RECT_ZY_SIZE 15. 0. 0. 20. 20.

SET "conf_additional" GET_CONFIGURATION_3D
DRAW_CONFIGURATION3D conf_additional
```

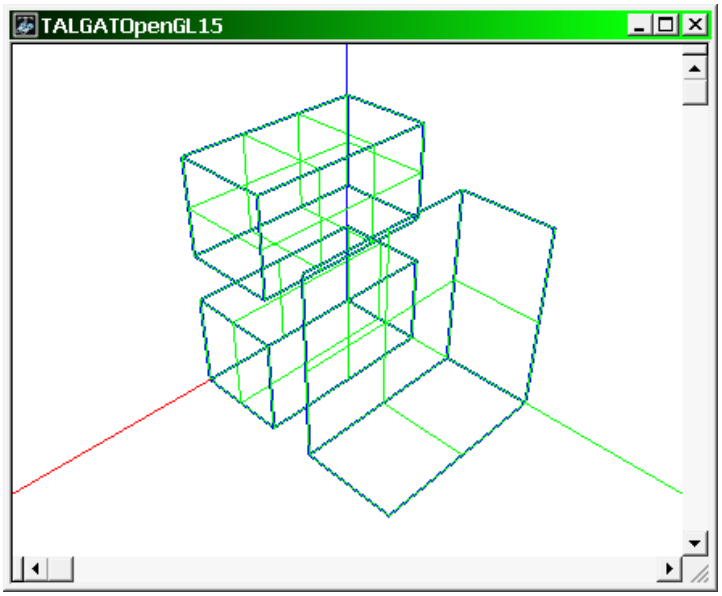


Рисунок 5.27 – Результат выполнения скрипта из листинга 5.16

Команда SPHERE аппроксимирует сферу ортогональными квадратными площадками. Количество подсекций на каждом таком квадрате задается командами SET_SUBINTERVALS_X и SET_SUBINTERVALS_Y. Командой SET_ER_PLUS3D задается значение ER вне сферы, SET_ER_MINUS3D — внутри сферы (для диэлектриков). При аппроксимации сферы автоматически определяются значения ER_PLUS и ER_MINUS так, чтобы данное условие выполнялось. Параметры: первый, второй и третий — координаты центра сферы; четвёртый — сторона куба, в который вписана сфера; пятый — сторона квадратов, которыми аппроксимируется сфера.

Команда HEMISPHERE аппроксимирует полусферу ортогональными квадратами. Количество подсекций на каждом квадрате задается командами SET_SUBINTERVALS_X и SET_SUBINTERVALS_Y. Командой SET_ER_PLUS3D задается значение ER вне полусферы, SET_ER_MINUS3D — внутри полусферы. При аппроксимации полусферы автоматически определяются значения ER_PLUS и ER_MINUS так, чтобы данное условие выполнялось. Параметры: первый, второй и третий – координаты центра полусферы; четвёртый — сторона куба, в который вписана полусфера; пятый — сторона квадратов, которыми аппроксимируется полусфера; шестой – строка, определяющая ось, которой перпендикулярна рассекающая сферу плоскость, возможные значения: «"HalfByX"», «"HalfByY"» или «"HalfByZ"»; седьмой — строка, определяющая, какую часть рассечённой сферы необходимо сохранить, возможные значения: «"Up"» (выше по оси), «Down» (ниже по оси).

Пример использования команд SPHERE и HEMISPHERE приведён в листинге 5.17 и на рисунке 5.28.

Листинг 5.17 – Пример использования дополнительных команд

```
CONDUCTOR3D
SPHERE 0. -15. 0. 10. 1.
CONDUCTOR3D
HEMISPHERE 0. 15. 0. 10. 1. "HalfByY" "Up"
SET "conf_additional" GET_CONFIGURATION_3D
DRAW_CONFIGURATION3D conf_additional
```

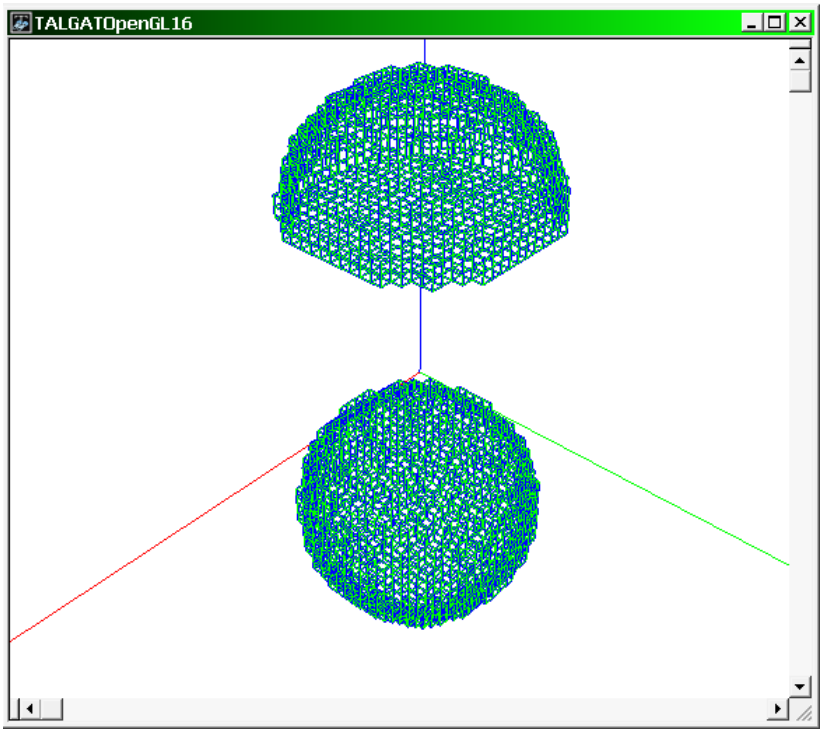


Рисунок 5.28 – Результат выполнения скрипта из листинга 5.17

5.1.3 Модуль трёхмерного электродинамического анализа MOMW

5.1.3.1 Общие сведения

Модуль MOMW предназначен для электродинамического анализа трёхмерных проводных структур и позволяет вычислить распределение токов в структуре произвольной формы, а из этих токов – различные характеристики. Перед использованием модуля трёхмерного проводного электродинамического анализа его необходимо сначала загрузить (листинг 5.18).

Листинг 5.18 – Подключение модулей, необходимых для трёхмерного проводного электродинамического анализа

```
INCLUDE "UTIL"
INCLUDE "MATRIX"
INCLUDE "MOMW"
```

5.1.3.2 Создание структуры

Сначала создадим простой диполь.

Команда CLEAR_STRUCTURE стирает ранее созданную структуру, поэтому должна вызываться перед началом создания каждой новой структуры.

Команда RADIUS имеет единственный параметр типа double, который задаёт радиус провода. Командами BEGIN и END задаются координаты начала и конца отрезка провода. Эти команды принимают по три параметра (числа типа double) – координаты X, Y, Z точек начала и конца отрезка провода.

Команда CREATE_WIRE создаёт отрезок провода, перед использованием этой команды необходимо задать все параметры данного отрезка, например его координаты, с помощью команд BEGIN и END.

Команда EXCITATION указывает, что в отрезке провода присутствует источник синусоидального воздействия, параметр – напряжение воздействия в вольтах (типа complex). По умолчанию точка запитки размещается в центре отрезка. Использование команды EXCITATION с параметром (0.,0.) соответствует отключению источника питания. После создания отрезка провода с использованием команды EXCITATION необходимо отключить источник питания с помощью команды «EXCITATION (0.,0.)».

Команда GET_STRUCTURE возвращает созданную структуру. При этом, в отличие от модулей MOM2D и MOM3D, структура не стирается и может быть дополнена или модифицирована в дальнейшем, а также повторно получена с помощью команды GET_STRUCTURE.

Команда DRAW_STRUCTURE отображает графическое представление структуры, параметр – переменная с сохранённой структурой.

Пример создания структуры приведён в листинге 5.19.

Листинг 5.19 – Пример создания структуры

```
CLEAR_STRUCTURE
RADIUS 1.e-4
   BEGIN 0. -.2418 0.
   END 0. -.025 0.
CREATE_WIRE
   BEGIN 0. .025 0.
   END 0. .2418 0.
CREATE_WIRE
   EXCITATION (1.,0.)
   BEGIN 0. -.025 0.
   END 0. .025 0.
CREATE_WIRE
```

```
   EXCITATION (0.,0.)
SET "dipole" GET_STRUCTURE
DRAW_STRUCTURE dipole
```

На рисунке 5.29 показан результат создания структуры с указанием для ясности координат точек.

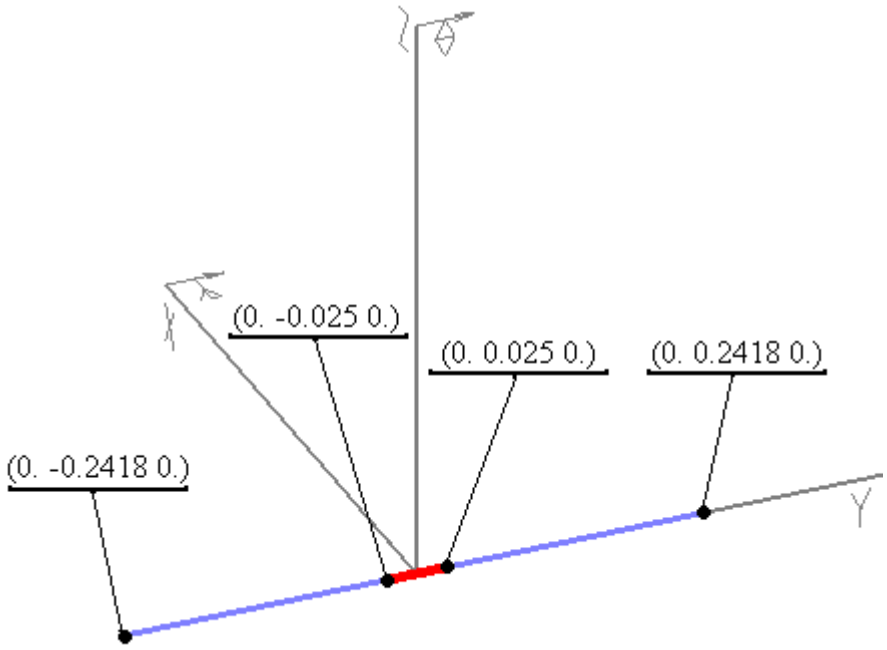
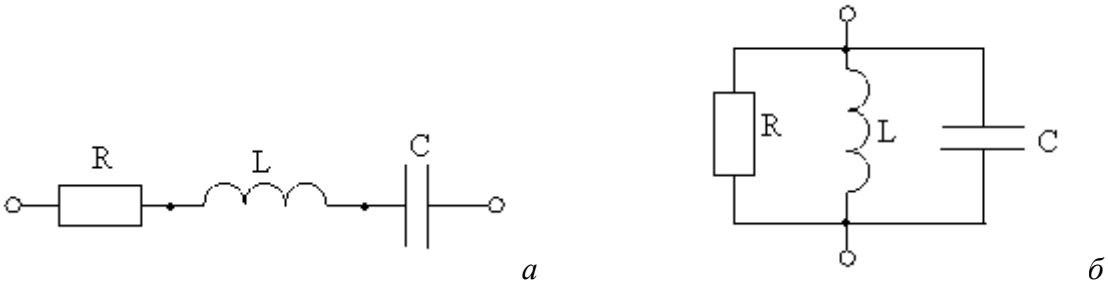


Рисунок 5.29 – Результат выполнения скрипта из листинга 5.19

5.1.3.3 Создание структуры с нагрузками

Команды LOAD_PARALLEL и LOAD_SERIAL создают нагрузку в виде параллельной или последовательной RLC-цепи (рисунок 5.30, а, б). Первый параметр — расстояние от начала отрезка до нагрузки, второй — сопротивление (Ом), третий — индуктивность (Гн), четвертый — ёмкость (Ф).



а *б*

Рисунок 5.30 – Нагрузка в виде: последовательной (*а*) и параллельной (*б*) RLC-цепей

Команда EXCITATION_DISTANCE аналогична команде EXCITATION, однако имеет второй параметр — расстояние от начала отрезка до точки, в которой расположен источник

питания. При этом после создания отрезка провода с использованием команды EXCITATION_DISTANCE также необходимо отключить источник питания с помощью команды «EXCITATION (0.,0.)».

Пример создания структуры с нагрузкой приведён в листинге 5.20.

Листинг 5.20 – Пример создания структуры с нагрузкой

```
CLEAR_STRUCTURE

RADIUS 0.02

   LOAD_SERIAL 0.74 1. 1.e-6 1.e-12
   BEGIN 0.76 0. 0.
   END 3. 0. 0.
CREATE_WIRE
   EXCITATION_DISTANCE (1.,0.) 0.005
   BEGIN 0.75 0. 0.
   END 0.76 0. 0.
CREATE_WIRE
   EXCITATION (0.,0.)

   BEGIN 0. 0. 0.
   END 0.75 0. 0.
CREATE_WIRE

SET "dipole_load" GET_STRUCTURE
DRAW_STRUCTURE dipole_load
```

Обратите внимание: в несегментированной структуре нагрузки не отображаются (рисунок 5.31), так как для размещения нагрузки необходимо знать номер подсекции, в которой она располагается, который становится известен только на этапе сегментации структуры.

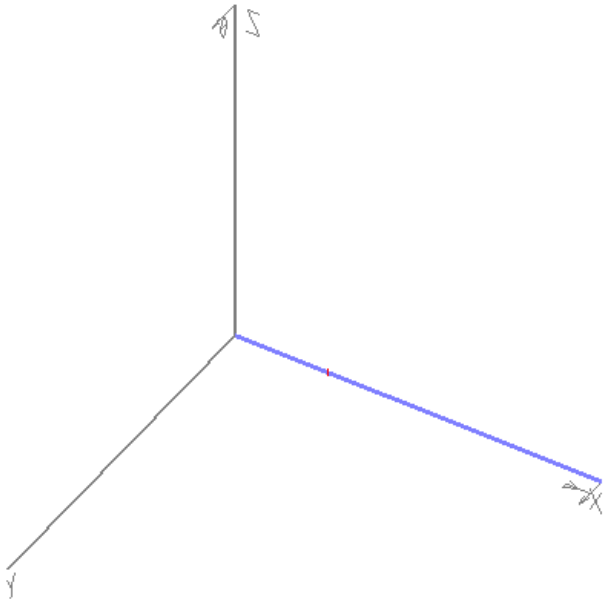


Рисунок 5.31 – Результат выполнения скрипта из листинга 5.20

5.1.3.4 Сегментация

Важные замечания:

- Сегментация представляет собой процесс деления отрезков проводов (секций) исходной структуры на подсекции. При этом исходная структура в виде секций и сегментированная структура в виде подсекций неразличимы для команд модуля.
- Не рекомендуется производить электродинамический анализ несегментированных (исходных) структур.
- Несегментированные (исходные) структуры предназначены только для компактного хранения исходных данных об исследуемых структурах, преобразования этих данных в форматы других программ и прочих сервисных процедур.
- При сегментации также происходит вычисление и проверка координат нагрузок в структуре. При обнаружении ошибок в положении нагрузок (например, нагрузка находится за пределами отрезка) генерируется исключение, и сегментация прерывается.
- При сегментации также происходит заполнение вектора воздействий, который необходим для решения СЛАУ. После выполнения сегментации вектор воздействий доступен через команду GET_EXCITATION_VECTOR.
- Не рекомендуется производить сегментацию структуры для одной частоты, а электродинамический анализ — для другой.

Команда SET_SUBSECTIONS_ON_WAVE задаёт число подсекций на волне при автосегментации. При сегментации структуры все провода делятся на участки, имеющие

длину, равную длине волны, которая соответствует заданной для выполнения сегментации частоте. Затем каждый участок делится на количество подсекций, заданное параметром команды SET_SUBSECTIONS_ON_WAVE. При этом минимальное число подсекций на отрезке провода задается параметром команды SET_SUBSECTIONS_MINIMAL.

Команда CREATE_SUBSECTIONS выполняет сегментацию структуры. Первый параметр — частота, на которой будет производиться анализ, а, значит, до этого необходимо выполнить сегментацию, второй параметр — переменная с сохранённой структурой.

После выполнения сегментации с помощью команды GET_EXCITATION_VECTOR можно получить вектор воздействий.

Пример работы с автосегментацией приведён в листинге 5.21 и на рисунке 5.32.

Листинг 5.21 – Пример создания структуры с автосегментацией

```
SET_SUBSECTIONS_ON_WAVE 10
SET_SUBSECTIONS_MINIMAL 10
SET "f" 300.e+6

SET "dipole_" CREATE_SUBSECTIONS f dipole
SET "exc_v" GET_EXCITATION_VECTOR

SET "dipole_load_" CREATE_SUBSECTIONS f dipole_load
SET "exc_v_load" GET_EXCITATION_VECTOR

SET "dipole_mseg_" CREATE_SUBSECTIONS f dipole_mseg
SET "exc_v_mseg" GET_EXCITATION_VECTOR

DRAW_STRUCTURE dipole_
DRAW_STRUCTURE dipole_load_
DRAW_STRUCTURE dipole_mseg
```

После выполнения сегментации команда GET_SUBSECTIONS_NUMBER позволяет получить количество подсекций в сегментированной структуре. Число подсекций равно порядку используемой в дальнейшем матрицы импедансов и позвоялет оценить количество памяти и времени, которые необходимо затратить на вычисление токов в заданной структуре. В качестве параметра команда GET_SUBSECTIONS_NUMBER принимает переменную с сохранённой сегментированной структурой.

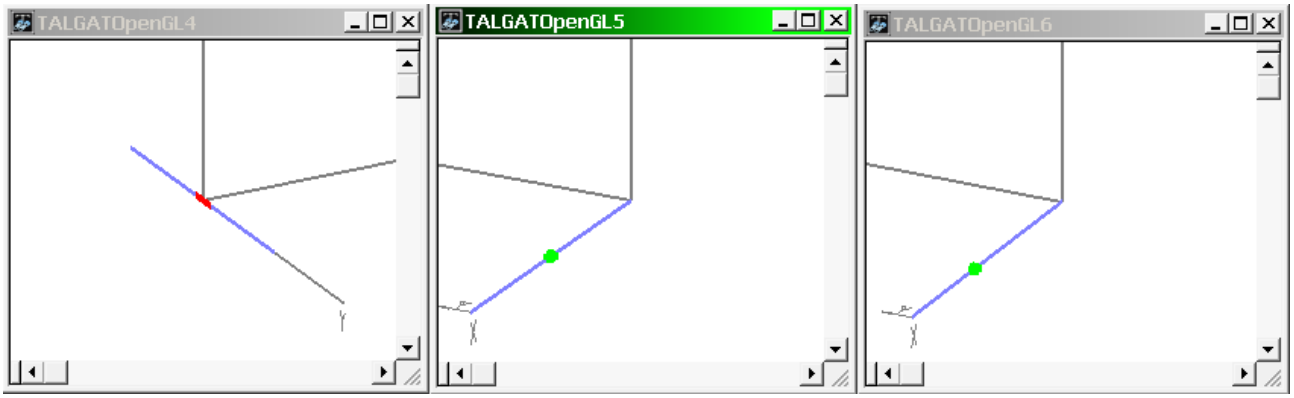


Рисунок 5.32 – Результат выполнения скрипта из листинга 5.21

5.1.3.5 Создание структуры с ручной сегментацией

При ручной сегментации с помощью команды SET_SUBSECTIONS задаётся число подсекций для каждого отрезка провода. После использования команды SET_SUBSECTIONS необходимо отключать ручную автосегментацию командой «SET_SUBSECTIONS -1».

Команда EXCITATION_SUBSECTION задаёт номер подсекции с источником питания. Первый параметр комплексный – значение воздействия источника питания, второй – номер подсекции с источником питания (индексация начинается с нуля).

Пример создания структуры с ручной сегментацией приведён в листинге 5.22.

Листинг 5.22 – Пример создания структуры с ручной сегментацией

```
CLEAR_STRUCTURE
RADIUS 0.02
   SET_SUBSECTIONS 3
   EXCITATION_SUBSECTION (1.,0.) 1
   BEGIN 0.75 0. 0.
   END 0.76 0. 0.
   CREATE_WIRE
EXCITATION (0.,0.)
   SET_SUBSECTIONS 15
   LOAD_SERIAL_SUBSECTION 7 1. 1.e-6 1.e-12
   BEGIN 0.76 0. 0.
   END 3. 0. 0.
CREATE_WIRE
   SET_SUBSECTIONS 5
   BEGIN 0. 0. 0.
   END 0.75 0. 0.
   CREATE_WIRE
SET "dipole_manual_segm" GET_STRUCTURE
SET_SUBSECTIONS -1
DRAW_STRUCTURE dipole_manual_segm
```

Структура, полученная в результате выполнения команд DRAW_STRUCTURE, будет выглядеть так же, как и в предыдущем примере (рисунок 5.32), однако, количество и размер подсекций будут отличаться.

5.1.3.6 Вычисление токов

Команда CALCULATE_IMPEDANCE_MATRIX используется для расчета матрицы импедансов. Первый параметр — частота (должна быть та же, что и при сегментации). Второй параметр — имя метода вычисления потенциального интеграла при расчете элементов матрицы импедансов:

- PIA126_127 — аналитический быстрый по Харрингтону;
- PIA129_135 — аналитический уточненный по Харрингтону;
- PIWerner — аналитический по Вернеру;
- PIUeddl — численное интегрирование методом Уэддля;
- PIBode — численное интегрирование методом Боде;
- PINewtonCotes — численное интегрирование методом Ньютона-Котеса (наибольшая точность).

Третий параметр — сегментированная структура.

Процедура одинакова для всех структур, поэтому произведем вычисления для первой структуры dipole_. Для вычисления других структур (без учета нагрузок) просто замените имена структур и векторов воздействий (например, dipole_ и exc_vec на dipole_load_ и exc_v_load).

Вычисление вектора токов проводится следующей командой: «SET "currents_v" LU_SOLVE LU_FACT impedance_m exc_v», то есть решается СЛАУ методом, основанном на LU-разложении.

Команда DRAW_CURRENTS отображает в графическом окне токи (цветами) в структуре. Первый параметр – сегментированная структура, второй параметр – вычисленный путем решения СЛАУ вектор токов, третий параметр может принимать два значения: 1 – толщина отрезков пропорциональна величине тока в данном отрезке, 0 – реальная толщина.

Пример вычисления токов приведён в листинге 5.23 и на рисунке 5.33.

Листинг 5.23 – Пример вычисления токов

```
SET "impedance_m" CALCULATE_IMPEDANCE_MATRIX f "PIA126_127" dipole_
SET "currents_v" LU_SOLVE LU_FACT impedance_m exc_v

SET "currents_v_iter" BICGSTABPRE impedance_m exc_v 0. 0 0 0.
ECHO NEW_LINE
```

```
ECHO LINE_TO_STRING * currents vector (LU_SOLVE) *
ECHO TO_STRING currents_v
ECHO NEW_LINE
ECHO LINE_TO_STRING *currents vector (BICGSTABPRE) *
ECHO TO_STRING currents_v_iter

DRAW_CURRENTS dipole_ currents_v 0
```

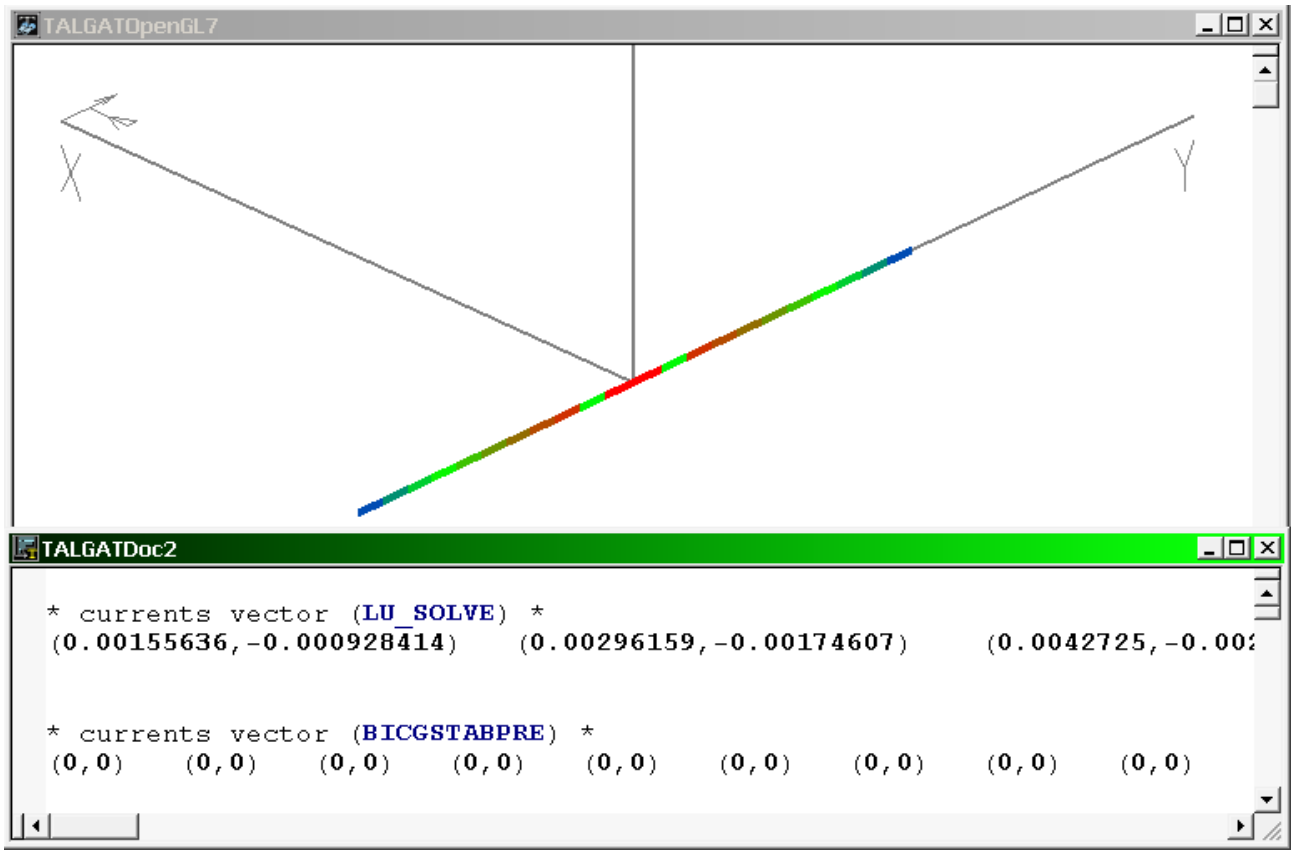


Рисунок 5.33 – Результат выполнения скрипта из листинга 5.23

5.1.3.7 Вычисление параметров

Процедура одинакова для всех структур, поэтому произведем вычисления для структуры dipole_. Для вычисления других структур (без учета нагрузок) просто замените имена структур и векторов воздействий (например, dipole_ и exc_vec на dipole_load_ и exc_v_load).

Обратите внимание: в случае одного источника можно использовать команду FOR_SINGLE_SOURCE, которая из переданного вектора воздействий выбирает элемент, соответствующий подсекции с источником питания. Первый параметр – сегментированная структура, второй – вектор воздействий.

Команды CALCULATE_ADMITTANCE и CALCULATE_IMPEDANCE вычисляют входной адмиттанс и импеданс структуры соответственно. Первый параметр – вектор токов, второй параметр – вектор воздействий.

Команда CALCULATE_REFLECTION_RATIO возвращает матрицу коэффициентов отражения. Первый параметр – импеданс, второй параметр – волновое сопротивление (Ом) воздействующего тракта.

Команда CALCULATE_VOLTAGE_STANDING_WAVE_RATIO возвращает коэффициент стоячей волны по напряжению. Параметр – коэффициент отражения.

Команда CALCULATE_TRANSMITTED_POWER_RATIO возвращает коэффициент передачи мощности. Параметр –коэффициент отражения.

Команда GET_FAR_ZONE_RADIUS возвращает радиус дальней зоны. Первый параметр – частота, на которой проводится анализ, второй параметр – сегментированная структура.

Команда CALCULATE_E_FAR_ZONE возвращает составляющие вектора напряженности E (В/м) в дальней зоне. Первый параметр – частота, второй параметр – вектор токов, третий параметр – сегментированная структура, четвертый параметр – радиус дальней зоны.

Пример вычисления параметров структур приведён в листинге 5.24 и на рисунок 5.34. Обратите внимание, что из-за ограниченной ширины страницы некоторые строки скрипта были перенесены. Однако в системе TUSUR.EMC эти строки не должны содержать переносов.

Листинг 5.24 – Пример вычисления параметров структур

```
ECHO NEW_LINE
ECHO LINE_TO_STRING * admittance *
ECHO FOR_SINGLE_SOURCE dipole_ CALCULATE_ADMITTANCE currents_v exc_v

ECHO NEW_LINE
ECHO LINE_TO_STRING * impedance *
SET "impedance_v" CALCULATE_IMPEDANCE currents_v exc_v
ECHO FOR_SINGLE_SOURCE dipole_ impedance_v

ECHO NEW_LINE
ECHO LINE_TO_STRING * reflection_ratio *
SET "reflection_ratio" FOR_SINGLE_SOURCE dipole_
CALCULATE_REFLECTION_RATIO impedance_v 50.
ECHO reflection_ratio

ECHO NEW_LINE
ECHO LINE_TO_STRING * VSWR *
ECHO CALCULATE_VOLTAGE_STANDING_WAVE_RATIO reflection_ratio
```

```
ECHO NEW_LINE
ECHO LINE_TO_STRING * TPR *
ECHO CALCULATE_TRANSMITTED_POWER_RATIO reflection_ratio

ECHO NEW_LINE
ECHO LINE_TO_STRING * E in far zone *
ECHO TO_STRING CALCULATE_E_FAR_ZONE f currents_v dipole_ 0. 0.
GET_FAR_ZONE_RADIUS f dipole_
```

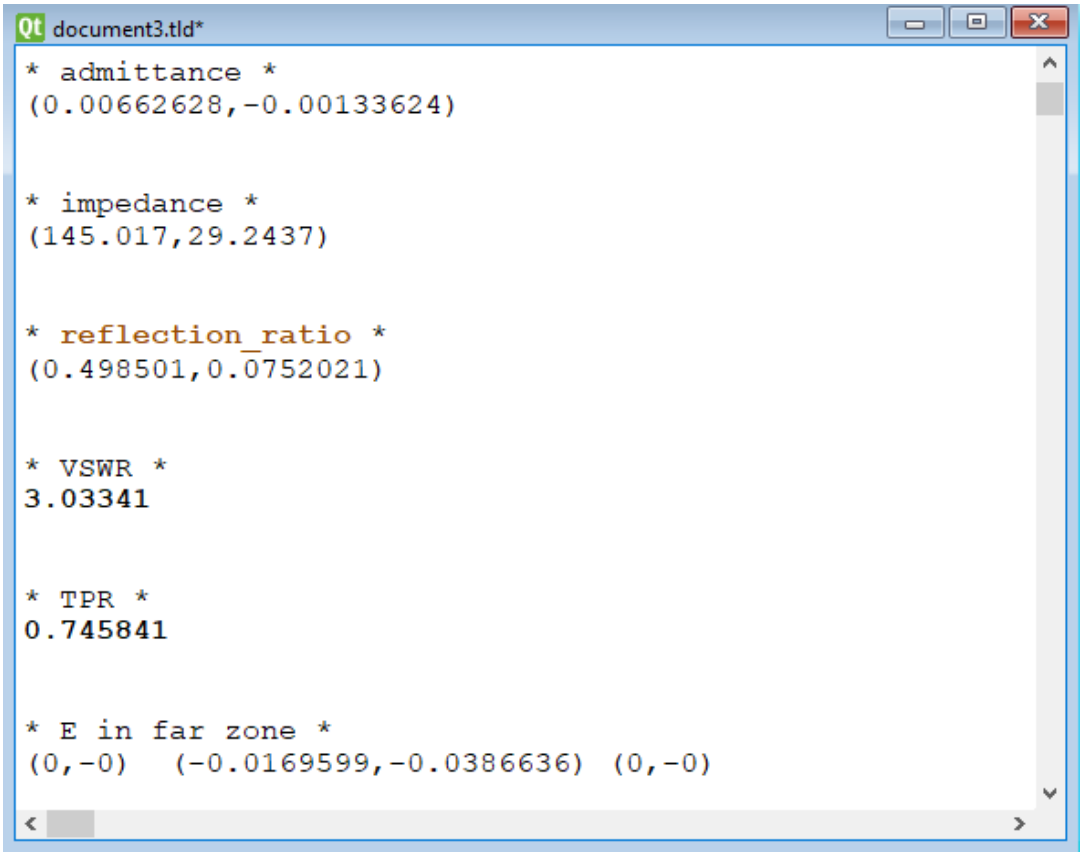


Рисунок 5.34 – Результат выполнения скрипта из листинга 5.24

5.1.3.8 Учет нагрузок

Электродинамический анализ с учетом нагрузок отличается только способом вычисления вектора токов. Прочие вычисления аналогичны.

Сначала необходимо с помощью команды CALCULATE_ADMITTANCE_MATRIX рассчитать матрицу адмиттансов на основе матрицы импедансов. Затем с помощью команды CALCULATE_CURRENTS_WITH_LOADS вычислить вектор токов с учётом нагрузок. Первый параметр – матрица адмиттансов, второй параметр – вектор воздействий, третий параметр – сегментированная структура, четвёртый параметр – метод учёта нагрузок (доступен только «"Altman"» — по Альтману), пятый параметр – частота.

Для примера вычислим вектор токов для второй структуры dipole_load_ (листинг 5.25 и рисунок 5.35). Обратите внимание, что из-за ограниченной ширины страницы некоторые строки скрипта были перенесены. Однако в системе TUSUR.EMC эти строки не должны содержать переносов.

Листинг 5.25 – Пример вычисления параметров структуры с учётом нагрузок

```
SET "impedance_m" CALCULATE_IMPEDANCE_MATRIX f "PIA126_127" dipole_load_
SET "admittance_m" CALCULATE_ADMITTANCE_MATRIX impedance_m
SET "currents_v_load" CALCULATE_CURRENTS_WITH_LOADS admittance_m exc_v_load
dipole_load_ "Altman" f

ECHO LINE_TO_STRING currents for structure with loads
ECHO TO_STRING currents_v_load
ECHO NEW_LINE

DRAW_CURRENTS dipole_load_ currents_v_load 0
```

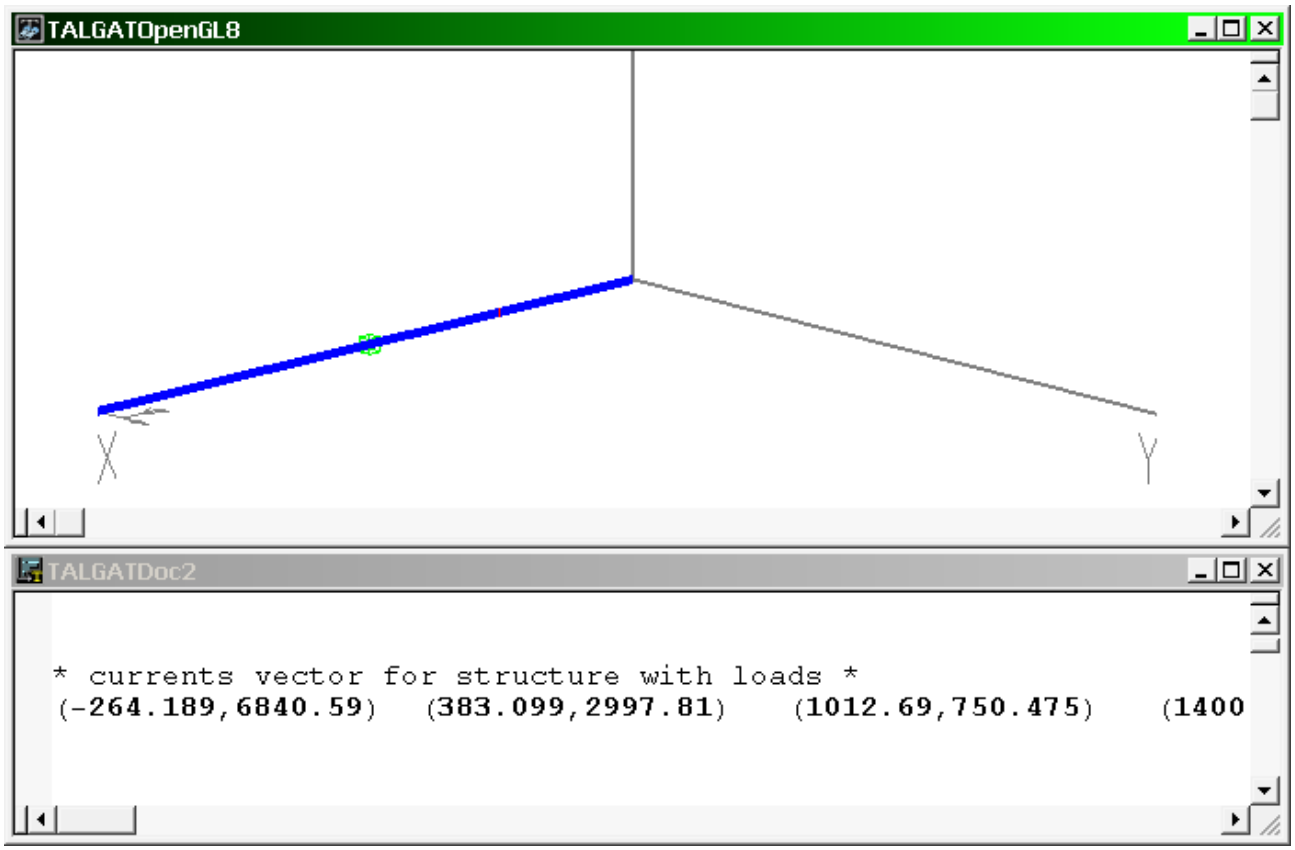


Рисунок 5.35 – Результат выполнения скрипта из листинга 5.25

5.1.3.9 Построение диаграмм

Команда CALCULATE_FAR_ZONE_DATA возвращает структуру данных, необходимую для построения диаграмм направленности. Первый параметр – частота, второй параметр – вектор токов, третий параметр – вектор воздействий, четвёртый параметр –

сегментированная структура, пятый – начальное значение teta, шестой – число шагов по teta, седьмой – шаг teta, восьмой – начальное значение fi, девятый – число шагов по fi, десятый – шаг fi. Обратите внимание: для получения более точных диаграмм направленности уменьшайте шаг по углу teta и по углу fi.

Команда FAR_ZONE_DATA_TO_MATRIX возвращает одну из матриц, содержащихся в структуре данных для построения ДН. Первый параметр – структура данных для построения ДН, второй параметр – данные для распечатки (доступные варианты: teta, fi, E, Eteta, Efi, G, Gteta, Gfi, где E означает напряженность поля, а G (Gain)– коэффициент направленного действия).

Команда DRAW_DN выводит диаграмму направленности в графическом окне визуального клиента. Первый параметр – сегментированная структура, второй параметр – структура данных для построения ДН, третий параметр – данные для построения (доступны: EFarZone, G), четвёртый параметр – построение по составляющим поля (доступны: by_sum, by_fi, by_teta).

Пример построения диаграммы приведён в листинге 5.26 и на рисунке 5.36. Обратите внимание, что из-за ограниченной ширины страницы некоторые строки скрипта были перенесены. Однако в системе TUSUR.EMC эти строки не должны содержать переносов.

Листинг 5.26 – Пример построения диаграммы

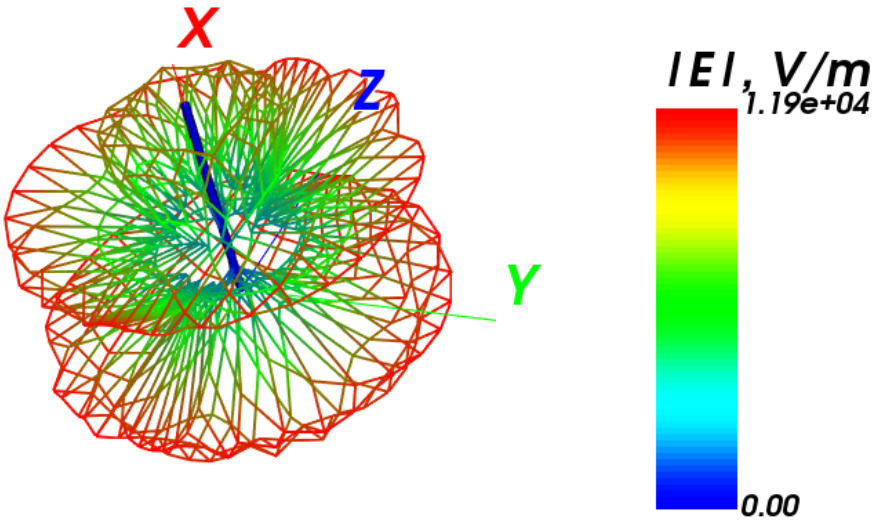
```
SET "far_zone_data" CALCULATE_FAR_ZONE_DATA f currents_v_load exc_v_load
dipole_load_ 0. 36 10. 0. 36 10.

ECHO FORMAT_STRING n*bsb* LINE_TO_STRING far zone data
ECHO TO_STRING far_zone_data

ECHO NEW_LINE
ECHO LINE_TO_STRING * Eteta matrix from far zone data *
ECHO TO_STRING FAR_ZONE_DATA_TO_MATRIX far_zone_data "Eteta"

DRAW_DN dipole_load_ far_zone_data "EFarZone" "by_sum"
```

При аппроксимации полусферы



document3.tld*

```
* far zone data *
EFarZoneData for frequency 3e+08
teta, rad       fi, rad |Eteta|, V/m    |Efi|, V/m      |E|, V/m        Gteta   Gfi
0       0       3863.68 0       3863.68 0.00696238      0       0.00696238
0       0.174533        3804.99 670.922 3863.68 0.00675244      0.000209941     0.00696238
0       0.349066        3630.68 1321.46 3863.68 0.00614794      0.000814444     0.00696238
0       0.523599        3346.05 1931.84 3863.68 0.00522179      0.0017406       0.00696238
0       0.698132        2959.75 2483.53 3863.68 0.00408569      0.00287669      0.00696238
0       0.872665        2483.53 2959.75 3863.68 0.00287669      0.00408569      0.00696238
0       1.0472  1931.84 3346.05 3863.68 0.0017406       0.00522179      0.00696238
0       1.22173 1321.46 3630.68 3863.68 0.000814444     0.00614794      0.00696238
0       1.39626 670.922 3804.99 3863.68 0.000209941     0.00675244      0.00696238
0       1.5708  2.36582e-13     3863.68 3863.68 2.61047e-35     0.00696238      0.00696238
0       1.74533 670.922 3804.99 3863.68 0.000209941     0.00675244      0.00696238
0       1.91986 1321.46 3630.68 3863.68 0.000814444     0.00614794      0.00696238
```

Рисунок 5.36 – Результат выполнения скрипта из листинга 5.26

5.1.4 Модуль вычисления временного и частотного откликов RESPONSE

Данный модуль предназначен для вычисления временного и частотного отклика. Сначала задается схема линии передач, затем происходит вычисление отклика одной из следующих команд:

3. F_RESPONSE – вычисление частотного отклика, в результате в переменную fs записывается матрица-строка частот, в переменные Vf1, Vf2 и так далее (для каждого узла в схеме) – частотные характеристики в виде матриц-строк. При этом для источников используются значения, заданные командой SIMULATION_SOURCES_HARMONICS, по умолчанию – 1 В (что дает нам АЧХ).

4. F_RESPONSE_I – то же самое, что F_RESPONSE, плюс дополнительно считаются токи в частотной области и записываются в виде матриц-строк в переменные If1, If2 и так далее.

5. T_RESPONSE – вычисление временного отклика: путем обратного преобразования Фурье из матриц-строк Vf1, Vf2, ... вычисляются напряжения и записываются в переменные V1, V2, ... в виде матриц-строк. При этом для источников используются значения, заданные командами SIMULATION_SOURCES_* (см. подробное описание ниже). Вычисляются временные точки, соответствующие частотам fs, и записываются в переменную ts. Кроме того, для удобства пользователя все напряжения для всех узлов записываются в строки одной матрицы и сохраняются в переменную mV, которую потом можно сохранить на диск одной командой (MATRIX_SAVE mV “myfile.mat”) и позже загрузить обратно (SET “mV” MATRIX_LOAD “myfile.mat”).

6. T_RESPONSE_I – то же самое, что T_RESPONSE, плюс дополнительно считаются токи во временной области и записываются в виде матриц-строк в переменные I1, I2 и так далее. Кроме того, для удобства пользователя все токи для всех узлов записываются в строки одной матрицы и сохраняются в переменную mI.

Для вычисления временного отклика необходимо к данному модулю RESPONSE загрузить модули UTIL, MATRIX и GRAFH (см. листинг 5.27).

Листинг 5.27 – Подключение модулей, необходимых для вычисления отклика

```
INCLUDE "UTIL"
INCLUDE "RESPONSE"
INCLUDE "MATRIX"
INCLUDE "GRAPH"
```

Для обновления параметров схемы при повторном вычислении, после подключения модулей, необходимо добавить команду CLEAR_SCHEME.

5.1.4.1 Параметры для вычисления переходных процессов

Для задания параметров используется команда TRANSIENT_ANALYSIS_SETUP. Данной командой задается временной шаг и число отсчетов на период повторения импульсов для алгоритмов БПФ (см. листинг 5.28).

Листинг 5.28 – Задание временного шага и числа отсчетов на период повторения импульсов

```
TRANSIENT_ANALYSIS_SETUP "step_time" 0.05E-9
или
TRANSIENT_ANALYSIS_SETUP "n_edge_count" 20
Первый аргумент – параметры "step_time" (временной шаг) или "n_edge_count" (число
отчетов на фронт). Второй аргумент - значение параметра.

TRANSIENT_ANALYSIS_SETUP "count_degree" 12
```

Первый аргумент – параметр "count_degree", который участвует в следующей формуле: 2count_degree. Второй аргумент - значение параметра.

5.1.4.2 Погонные параметры линий передачи

Для задания матриц погонных параметров используются команды CREATE_REAL_MATRIX и SET_MATRIX_VALUE. Кроме того, вместо ручного ввода матриц можно использовать матрицы, полученные в результате вычисления матриц C, C0, L, L0 средствами модуля MOM2D (см. листинг 5.29). Пример ручного ввода матриц погонных параметров приведен в листинге 5.29. Порядок матриц погонных параметров определяется числом проводников, не считая опорного.

Листинг 5.29 – Ручной ввод матриц погонных параметров линий передачи

```
SET "L" CREATE_REAL_MATRIX 2 2
SET "B" CREATE_REAL_MATRIX 2 2
SET "R" CREATE_REAL_MATRIX 2 2
SET "G" CREATE_REAL_MATRIX 2 2
SET "L" SET_MATRIX_VALUE L 0 0 494.6e-9
SET "L" SET_MATRIX_VALUE L 0 1 63.3e-9
SET "L" SET_MATRIX_VALUE L 1 0 63.3e-9
SET "L" SET_MATRIX_VALUE L 1 1 494.6e-9
SET "C" SET_MATRIX_VALUE C 0 0 62.8e-12
SET "C" SET_MATRIX_VALUE C 0 1 -4.9e-12
SET "C" SET_MATRIX_VALUE C 1 0 -4.9e-12
SET "C" SET_MATRIX_VALUE C 1 1 62.8e-12
SET "R" SET_MATRIX_VALUE R 0 0 0.1
SET "R" SET_MATRIX_VALUE R 0 1 0.02
SET "R" SET_MATRIX_VALUE R 1 0 0.02
SET "R" SET_MATRIX_VALUE R 1 1 0.1
SET "G" SET_MATRIX_VALUE G 0 0 0.1
SET "G" SET_MATRIX_VALUE G 0 1 -0.01
SET "G" SET_MATRIX_VALUE G 1 0 -0.01
SET "G" SET_MATRIX_VALUE G 1 1 0.1
```

5.1.4.3 Построение схемы

Для построения схемы используются следующие команды:

- RESISTOR – резистор;
- CAPACITOR – емкость;
- INDUCTANCE – индуктивность;
- SOURCE – источник воздействия;
- TRANSMISSION_LINE – линия передачи.

5.1.4.4 Команды RESISTOR, CAPACITOR, INDUCTANCE

Пример задания элементов схемы с использованием этих команд приведён в листинге 5.30.

Листинг 5.30 – Задание команд для резистора, емкости и индуктивности

```
RESISTOR "R1" 1 2 50.
CAPACITOR "C1" 1 2 1.e-12
INDUCTANCE "L1" 1 2 1.e-9
```

Первый аргумент – имя конкретного параметра. Второй и третий аргументы – номера узлов, между которыми расположен элемент, а четвертый – значение параметра.

5.1.4.5 Команда SOURCE

Пример задания источника воздействия показан в листинге 5.31.

Листинг 5.31 – Задание источника воздействия

```
SOURCE "V1" 0 1
```

Первый аргумент – имя конкретного параметра. Второй и третий аргументы – номера узлов, между которыми расположен источник. Для задания типа источника и его параметров используются команды:

7. Импульсный сигнал: SIMULATION_SOURCES_VPULSE Vin Vpv tTD tRT tFT tD Period, где Vin – постоянная составляющая, В; Vpv – максимальное значение напряжения, В; tTD – время задержки, с; tRT – длительность переднего фронта, с; tFT – длительность заднего фронта, с; tD – длительность вершины импульса, с; Period – период повторения импульсов, с.

8. Синусоидальный сигнал: SIMULATION_SOURCES_VSIN Vin Vpv Fsin tTD THETA Phase, где Vin – постоянная составляющая, В; Vpv – максимальное значение амплитуды, В; Fsin – частота сигнала, Гц; tTD – время задержки, с; THETA - коэффициент затухания, 1/с; Phase - Фаза, град.

9. Гауссов импульс: SIMULATION_SOURCES_VGAUSS Vpv fc r bw tpr bwr, где Vpv – максимальное значение амплитуды, В; fc – несущая частота генерируемого сигнала, Гц; bw – относительная (т.е. нормированная к несущей частоте) ширина спектра, %; tpr – максимальное уменьшение огибающей относительно ее пикового уровня, дБ; bwr – уменьшение спектральной функции относительно ее пикового значения, дБ.

10. Экспоненциальный импульс: SIMULATION_SOURCES_VEXP Vin Vpv tRD tRT tFT, где Vin – постоянная составляющая, В; Vpv – максимальное значение амплитуды, В; tRD – время задержки, с; tRT – длительность переднего фронта, с; tFT – длительность заднего фронта, с.

5.1.4.6 Команда TRANSMISSION_LINE

Для задания линий передачи в схему, используются следующие команды:

TRANSMISSION_LINE "t11" N in1 out1 … inN outN, где N – количество проводников, а inN – номер узла, с которым соединен проводник N в начале линии и outN – номер узла, с которым соединен проводник N в конце линии. Дополнительно для задания параметров линии передачи используется команда:

TRANSMISSION_LINE_PARAMETERS L C R G length, где C – матрица погонных коэффициентов электростатической индукции, L – матрица погонных коэффициентов электромагнитной индукции, G – матрица погонных проводимостей, R – матрица погонных сопротивлений, length – длина отрезка линии.

Пример создания схемы приведен в листинге 5.32

Листинг 5.32 – Команды для построения схемы в TUSUR.EMC

```
RESISTOR "R1" 1 2 50
RESISTOR "R2" 0 3 100
RESISTOR "R3" 0 4 100
RESISTOR "R4" 0 5 100

SOURCE "V1" 0 1
SIMULATION_SOURCES_VPULSE 0. 1. 0. 1.5e-9 1.5e-9 6.0e-9 0.

TRANSMISSION_LINE "tl1" GET_MATRIX_ROWS L 2 4 3 5
TRANSMISSION_LINE_PARAMETERS L B R G 0.3048
```

При создании элемента линия передачи (transmission line) доступен новый параметр – количество сегментов. В предыдущих версиях системы этот параметр настраивался в окне *DynaVis*, при запуске интерпретации кода и был одинаковым для всех линий передачи схемы.

При работе со скриптовым вводом установка данного параметра осуществляется через шестой аргумент новой команды TRANSMISSION_LINE_PARAMETERS6 (в предыдущих версиях TRANSMISSION_LINE_PARAMETERS). Также сохранена поддержка работы скриптов, созданных в предыдущей версии системы.

5.1.4.7 Вывод схемы

Для вывода созданной схемы в графическое окно используется команда DRAW_SCHEME. Пример отображения схемы в TUSUR.EMC приведен на рисунке 5.37.

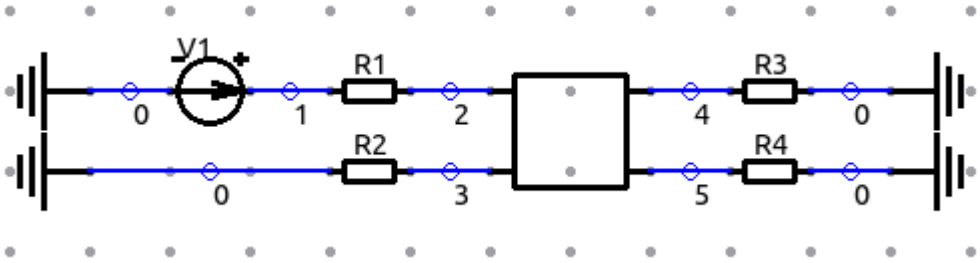


Рисунок 5.37 – Схема, построенная в TUSUR.EMC

5.1.4.8 Вычисление отклика

После ввода параметров схемы для вычисления напряжений задается команда T_RESPONSE, а для токов – T_RESPONSE_I. Аргументом для обеих команд является частота, на которой вычислялись матрицы потерь R, G. Пример использования этой команды приведен в листинге 5.33

Листинг 5.33 – Вычисление отклика

```
SET "f0" 1.0000e+9
T_RESPONSE f0 (или T_RESPONSE_I f0)
```

5.1.4.9 Вывод полученных результатов

Для вывода результатов вычисления в графическое окно (рисунок 5.38) используются команды, приведенные в листинге 5.34

Листинг 5.34 – Команды для вывода результатов вычисления напряжений в графическое окно

```
ADD_XY_DATA_c ts V1 (или I2) COMPLEX_PLOT_REAL
SET_PLOT_COLOR 0. 0. 0.
SET_PLOT_LABEL LINE_TO_STRING V1 (или I2)
ADD_XY_DATA_c ts V2 (или I2) COMPLEX_PLOT_REAL
SET_PLOT_COLOR 0. 0. 0.
SET_PLOT_LABEL LINE_TO_STRING V2 (или I2)
ADD_XY_DATA_c ts V3 (или I3) COMPLEX_PLOT_REAL
SET_PLOT_COLOR 0. 0. 1.
SET_PLOT_LABEL LINE_TO_STRING V3 (или I3)
ADD_XY_DATA_c ts V4 (или I4) COMPLEX_PLOT_REAL
SET_PLOT_COLOR 1. 0. 0.
SET_PLOT_LABEL LINE_TO_STRING V4 (или I4)
```

```
ADD_XY_DATA_c ts V5 (или I5) COMPLEX_PLOT_REAL
SET_PLOT_COLOR 0. 1. 0.
SET_PLOT_LABEL LINE_TO_STRING V5 (или I5)
SET_PLOT_TITLE LINE_TO_STRING time response
SET_X_TITLE LINE_TO_STRING t
SET_Y_TITLE LINE_TO_STRING V (или I)
SET_PLOT_RANGE 0. -0.2 2.e-8 0.7
PLOT_XY
```

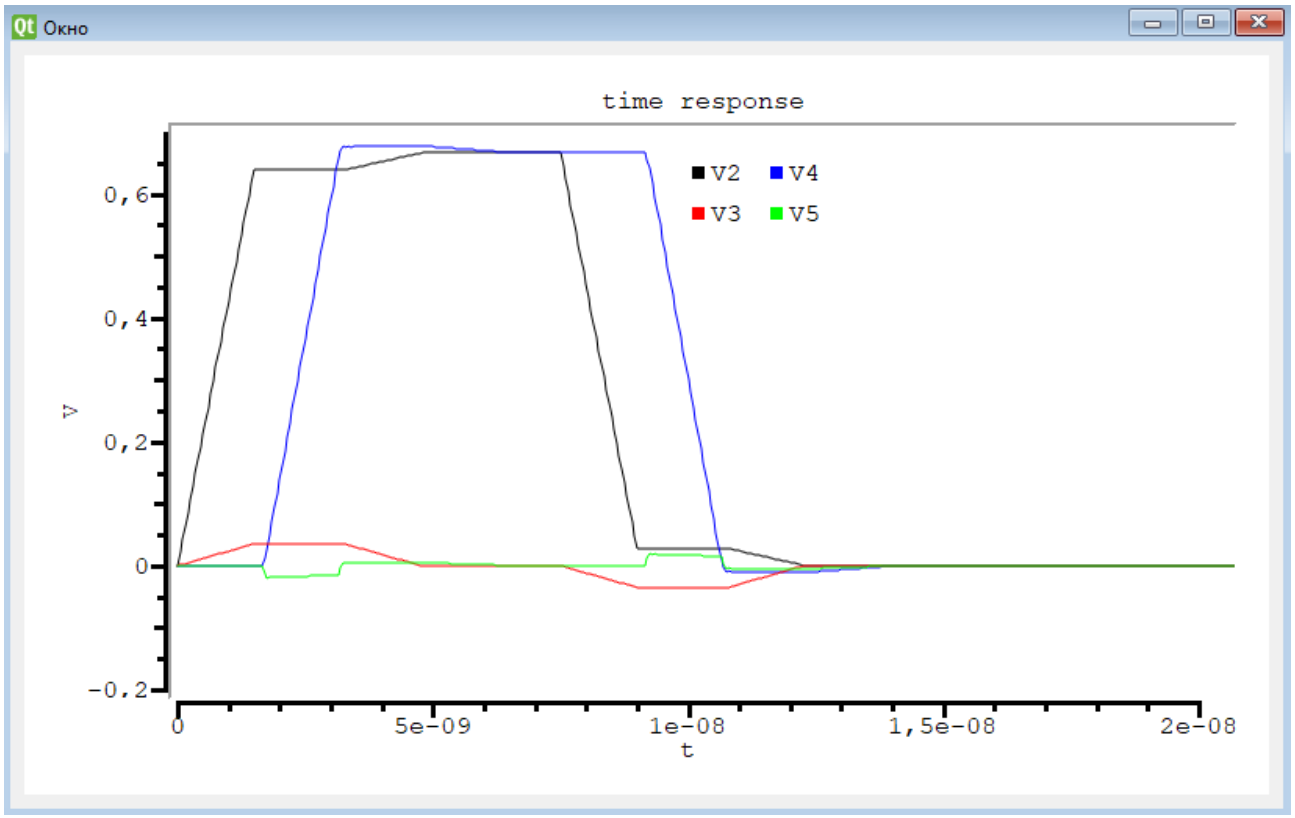


Рисунок 5.38 – Графическое окно с результатами вычислений

5.1.4.10 Графический редактор

Графический редактор системы TUSUR.EMC используется для построения произвольных принципиальных схем из отрезков регулярных линий передачи и эмуляции нагрузок в виде схем из элементов с сосредоточенными параметрами (далее редактор схем). Окно редактора принципиальных схем представлено на рисунке 5.39. Меню редактора состоит из следующих элементов:

 – создание нового окна для построения принципиальной схемы;

 – соединительная линия (соединяет элементы схемы);

 – резистор;

 – конденсатор;

 – индуктивность;

 – источник напряжения;

 – источник тока;

 – отрезок регулярной линии передачи;

 – сигнальная или схемная земля;

 – маркер (устанавливается в узлах схемы для вывода результатов вычислений)

 – запуск вычисления отклика.

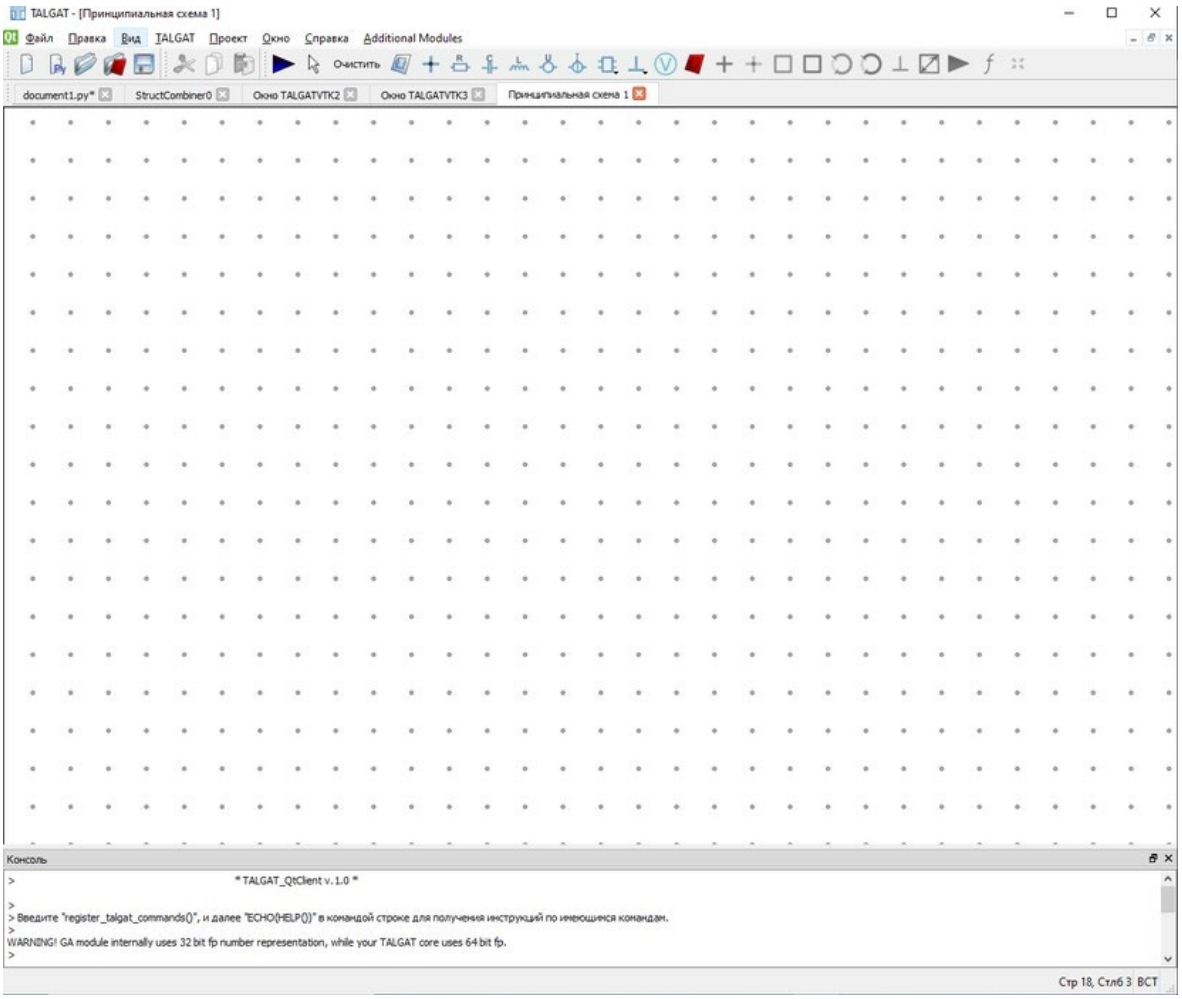


Рисунок 5.39 – Окно редактора принципиальных схем

При построении схемы пользователь выбирает элементы схемы в меню и размещает их на поле окна редактора. Элементы соединяются между собой с помощью элемента «соединительная линия». При размещении элемента на поле ему автоматически присваивается имя. В местах соединения выводов элементов образуются узлы, которым присваиваются соответствующие номера. При этом сигнальной земле соответствует номер узла 0. Для таких элементов как резистор, конденсатор, индуктивность, источник тока, источник напряжения, отрезок линии передачи создано дополнительное окно «Параметры». С помощью него задаются соответствующие параметры элементов. Примеры ввода

параметров для резистора, источника воздействия и отрезка линии передачи представлены на рисунке 5.40.

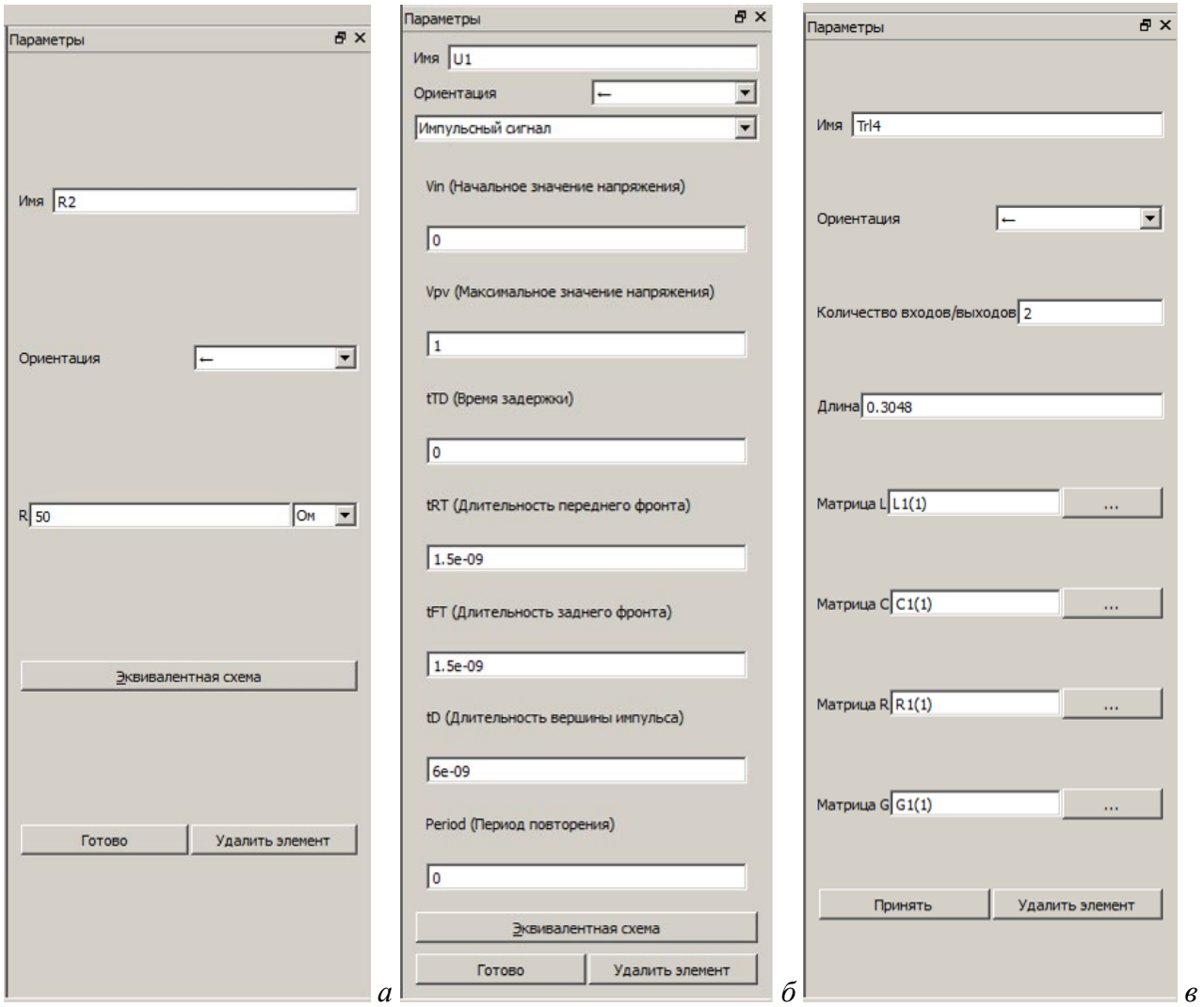


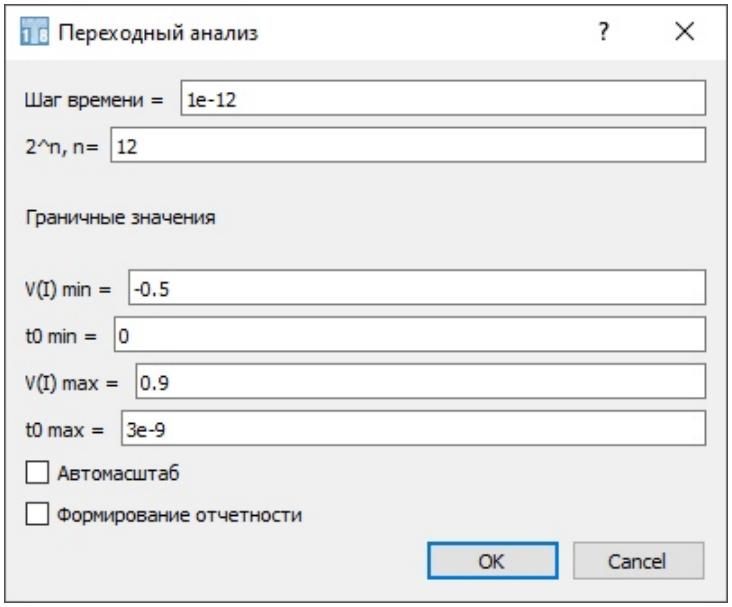
Рисунок 5.40 – Окно ввода параметров для резистора (*а*), импульсного сигнала (*б*) и отрезка линии передачи (*в*)

Для вычисления откликов с помощью редактора принципиальных схем используется кнопка «Запуск вычислений отклика». При ее нажатии выполняется запуск окна выбора типа переходного анализа, в котором пользователь может выбрать временной или частотный анализ заданной топологии. После выбора типа переходного анализа происходит запуск окна его настроек.

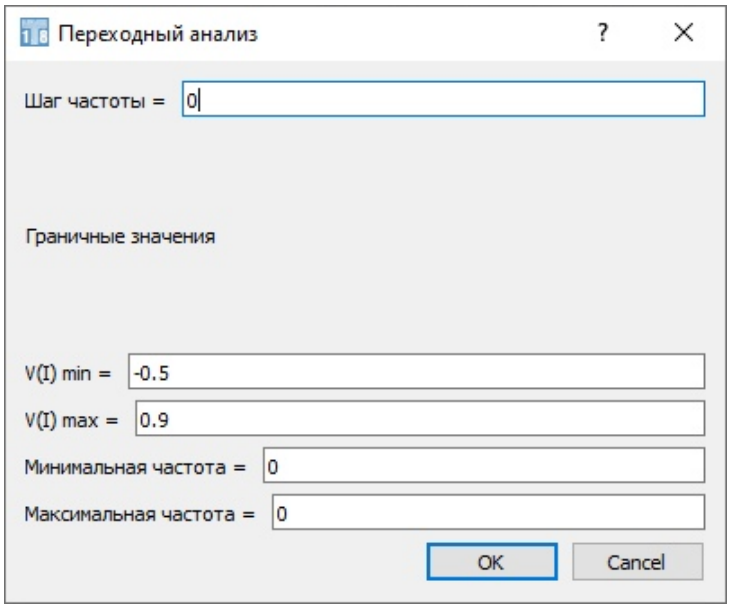
Интерфейс окна настроек временного анализа представлен на рисунке 5.41*а*. Пользователю доступно задание шага времени («Step_time») и числа отсчетов по времени (2^{count_degree}). Предусмотрены настройки графического отображения откликов: $V(I)_{min}$, $V(I)_{max}$ – минимальные и максимальные значения напряжения (тока) по оси ординат; t_{0min}, t_{0max} –

минимальные и максимальные значения по времени (ось абсцисс). Также доступна функция автоматического масштабирования графика.

Интерфейс окна настроек частотного анализа представлен на рисунке 5.41*б*. В окне доступно задание диапазона частот (используется для вычисления и отображения отклика), шага по частоте, минимального и максимального значений напряжения (тока) при отображении отклика. Если параметры, отвечающие за настройки частотного диапазона, не будут заданы (равны нулю), то система автоматически рассчитает частотный отклик на основании параметров источника сигнала, заданного в исследуемой принципиальной схеме.



а

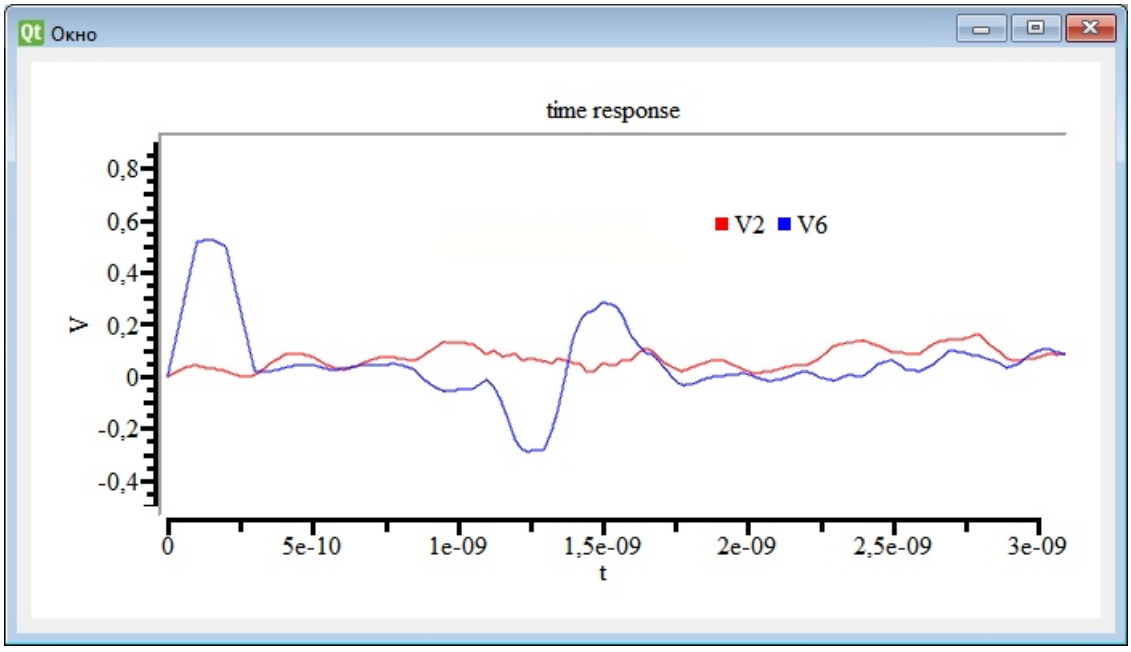


б

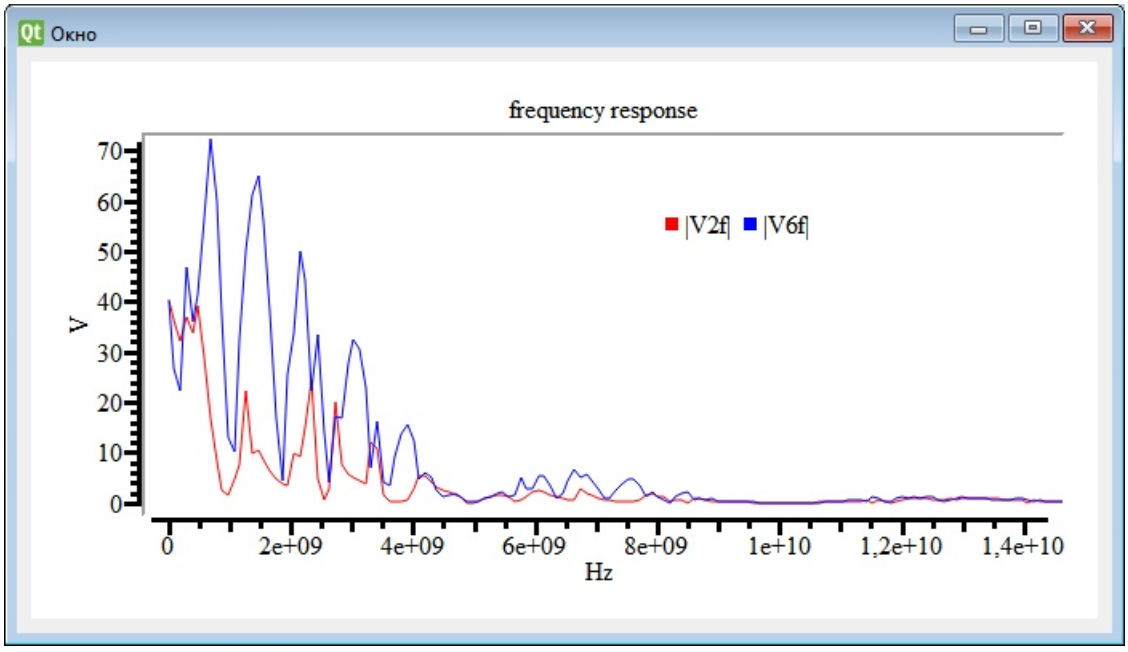
Рисунок 5.41 – Окна настроек временного (*а*) и частотного (*б*) анализа

После задания требуемых параметров генерируется код, который компилируется автоматически, и запускает окно с графическим отображением отклика (пример отображения

временного и частотного откликов представлен на рисунке 5.42). В окне с графическим отображением отклика доступна функция масштабирования с помощью колеса мыши или выделением прямоугольной области. Также имеются функции перемещения по графику (путем зажатия левой кнопки мыши и перемещения курсора) и изменения положения легенды. При наведении на рассчитанный отклик указателя мыши в его местоположении отображаются значение напряжения (тока) и время, соответствующее данному значению.



а



б

Рисунок 5.42 – Пример отображения временного (*а*) и частотного (*б*) откликов

5.2 Модули оптимизации

5.2.1 Модуль генетических алгоритмов GA

5.2.1.1 Общие сведения

Предполагается, что пользователь знаком с генетическими алгоритмами (ГА), принципами их работы и терминологией.

Сначала необходимо загрузить модуль GA (листинг 5.35).

Листинг 5.35 – Подготовка к использованию модуля GA

```
INCLUDE "UTIL"
INCLUDE "GA"
```

5.2.1.2 Простой пример оптимизации

Команда GA_MIN находит минимум функции. Параметры:

11. размер популяции;
12. число поколений;
13. коэффициент мутаций;
14. коэффициент кроссовера;
15. число аргументов функции качества num_vars;
16. от 6 до 5+2·num_vars — пары значений нижнего и верхнего пределов изменения аргументов функции качества;
17. последний параметр – функция качества.

При этом внутри функции качества существуют команды GA_PARAM_1, GA_PARAM_2 и так далее до GA_PARAM_ с числом аргументов функции качества, возвращающие параметры функции качества, для которых необходимо рассчитать ее значение.

Команда GET_BEST_GA_RESULT возвращает найденное ГА оптимальное значение функции качества.

Команда GET_BEST_GA_PARAMETER возвращает значения аргументов функции качества, при котором ее значение равно оптимальному. Параметр – индекс аргумента функции качества (индексация начинается с нуля и соответствует порядку, в котором аргументы идут при вызове команды GA_MIN).

В листинге 5.36 показан пример поиска минимума функции F(x,y) = x 2 + y 2 , а на рисунке 5.43 – результаты его выполнения. Сама функция качества создаётся динамически командой CREATE_KEYWORD (см. разд. 4.2.2.6).

Листинг 5.36 – Пример поиска минимума функции

```
CREATE_KEYWORD "quality_function"
   SET "x" GA_PARAM_1
   SET "y" GA_PARAM_2
   SET "qf_result" PLUS MUL x x MUL y y
END_CREATE_KEYWORD qf_result

GA_MIN 30 5 0.1 0.5 2 -10. 10. -10. 10. "quality_function"

ECHO FORMAT_STRING std GA_optim_result GET_BEST_GA_RESULT
ECHO FORMAT_STRING std optimized_x GET_BEST_GA_PARAMETER 0
ECHO FORMAT_STRING std optimized_y GET_BEST_GA_PARAMETER 1
```

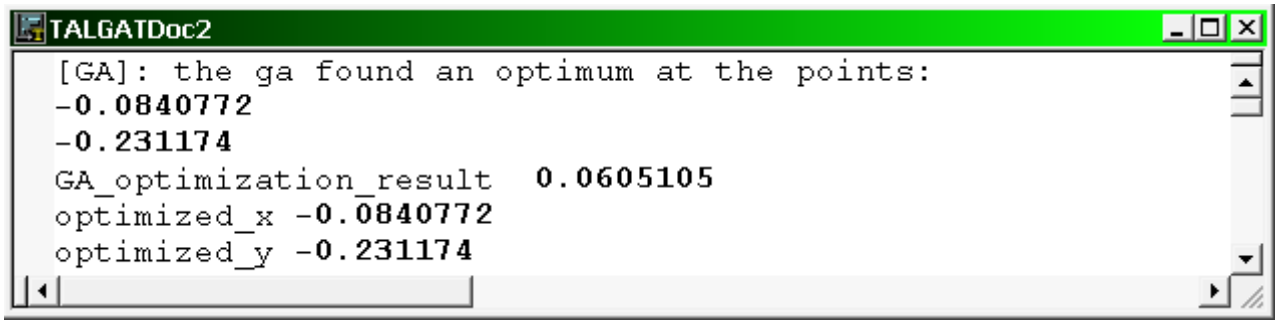


Рисунок 5.43 – Результаты выполнения скрипта из листинга 5.36

Команда GA_MAX выполняет поиск максимума функции качества. Параметры аналогичны команде GA_MIN. При поиске максимума той же функции F(x,y) = x 2 + y 2 , ГА выдает в качестве результата значения границ диапазона оптимизации параметров функции — (-10,-10) или (10,10), что говорит о том, что функция не имеет максимума.

5.2.1.3 Измерение времени оптимизации

Пример измерения времени оптимизации приведён в листинге 5.37, а результаты его выполнения – на рисунке 5.44. Команда REPORT_TIMER в данном случае выведет время оптимизации в секундах. Обратите внимание, что из-за ограниченной ширины страницы некоторые строки скрипта были перенесены. Однако в системе TUSUR.EMC эти строки не должны содержать переносов.

Листинг 5.37 – Пример измерения времени оптимизации

```
REPORT_TIMER GA_MAX 30 5 0.1 0.5 2 -10. 10. -10. 10. "quality_function"
```

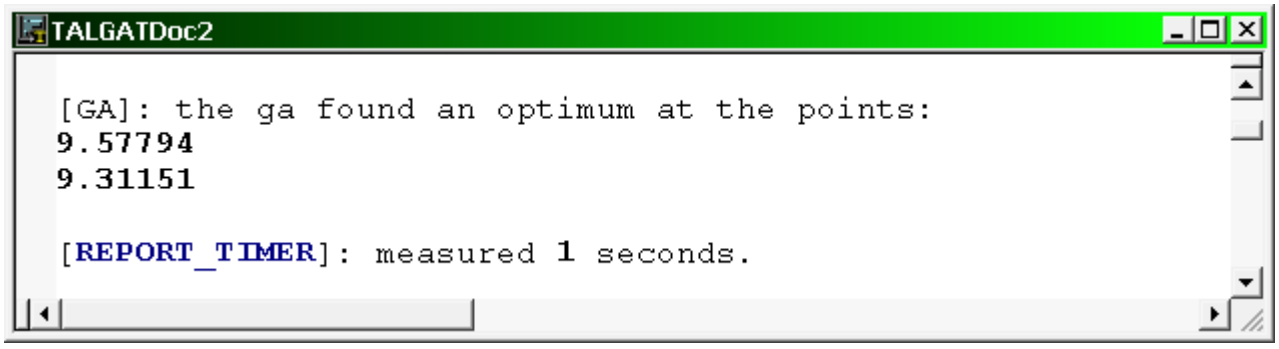


Рисунок 5.44 – Результаты выполнения скрипта из листинга 5.37

5.2.1.4 Дополнительные команды

Команда SET_NBITS_PER_NUMBER устанавливает для ГА количество битов на один ген в геноме (по умолчанию 16). Соответственно, команда GET_NBITS_PER_NUMBER возвращает текущее количество битов на один ген в геноме.

5.2.1.5 Оптимизация двумерной конфигурации

Задача: минимизировать разницу (Kc-Kl) для обращенной микрополосковой линии (рисунок 5.45) путем изменения параметра hd2 (толщины второго слоя диэлектрика). Исходные данные: число линий — 2, d = 1, w = d, t = w · 0.1, s = d, hd1 = w · 0.5, hd2 = w · 1, er1 = 2, er2 = 5, er3 = 1.

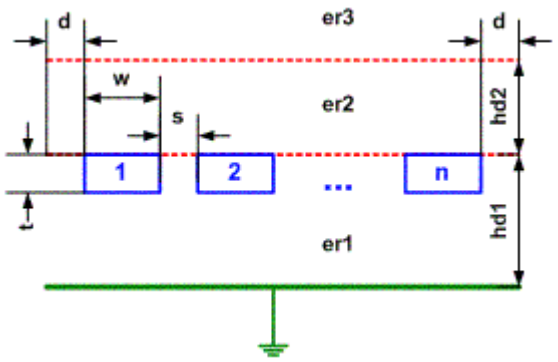
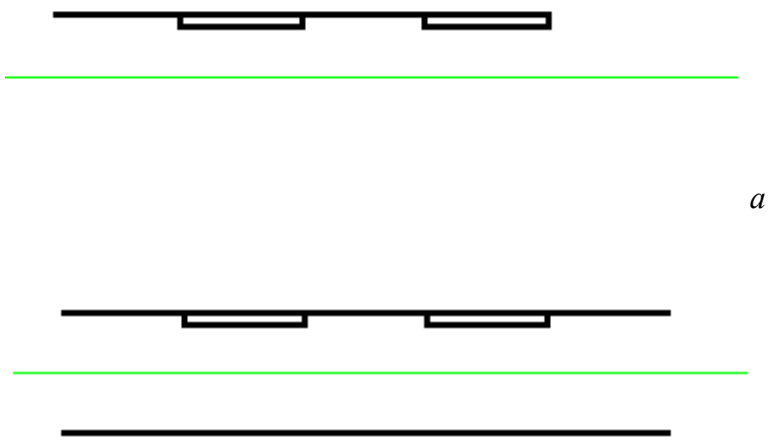


Рисунок 5.45 – Обращенная микрополосковая линия

В листинге 5.38 приведён скрипт, инициализирующий динамическую команду add_cnd, которая создаёт проводник и часть диэлектрика под ним (рисунок 5.46, а).



а

б

Рисунок 5.46 – Пошаговое создание обращённой микрополосковой линии: (*а*) результат выполнения двух вызовов add_cnd; (*б*) результат однократного выполнения qf

Листинг 5.38 – Задание исходных параметров и динамической команды для оптимизации двумерной конфигурации

```
INCLUDE "MOM2D"
INCLUDE "MATRIX"

SET "num_of_lines" 2

SET "d" 1.
SET "w" d
SET "t" MUL w 0.1
SET "s" d
SET "hd1" MUL w 0.5
SET "hd2" MUL w 1.

SET "er1" 2.
SET "er2" 5.
SET "er3" 1.

SET "subint" 8

SET "si_diel" subint
SET "si_diel_btw" DIV subint 4
SET "si_cndw" DIV subint 4
SET "si_cndt" DIV subint 8

SET_INFINITE_GROUND 1

CREATE_KEYWORD "add_cnd"

DIELECTRIC
   SET_SUBINTERVALS si_diel_btw
   SET_ER_PLUS er1
   SET_ER_MINUS er2
   LINE MINUS x_begin d hd1 x_begin hd1

CONDUCTOR
   SET_E R_PLUS er2
   SET_SUBINTERVALS si_cndw
   LINE x_begin hd1 PLUS x_begin w hd1
   SET_ER_PLUS er1
   SET_SUBINTERVALS si_cndt
   LINETO PLUS x_begin w MINUS hd1 t
   SET_SUBINTERVALS si_cndw
   LINETO x_begin MINUS hd1 t
   SET_SUBINTERVALS si_cndt
   LINETO x_begin hd1

SET_VARIABLE "x_begin" PLUS x_begin PLUS w
 s

END_CREATE_KEYWORD
```

Скрипт из листинга 5.39 инициализирует динамическую функцию качества qf, которая, циклически вызывая add_cnd (число вызовов равно числу линий), полностью

достраивает конфигурацию, анализирует структуру и вычисляет разницу Kc-Kl – эта величина и возвращается функцией качества. Затем проводится минимизация с помощью ГА. Для того, чтобы нарисовать оптимальную конфигурацию, в функции качества постоянно сохраняются новые конфигурации с (Kc-Kl) меньшим, чем начальное значение. Обратите внимание, что из-за ограниченной ширины страницы некоторые строки скрипта были перенесены. Однако в системе TUSUR.EMC эти строки не должны содержать переносов. В результате оптимизации получили, что оптимальное соотношение hd2/w равно 0.0024882, при этом разница Kc-Kl минимальна и составляет 0.0154497 (рисунок 5.47).

Листинг 5.39 – Задание функции качества и оптимизация двумерной конфигурации с помощью ГА

```
CREATE_KEYWORD "qf"

SET "hd2" MUL w GA_PARAM_1
SET "hd2" PLUS hd1 hd2

SET "x_begin" d

CYCLE num_of_lines add_cnd

DIELECTRIC
   SET_SUBINTERVALS si_diel_btw
   SET_ER_PLUS er1
   SET_ER_MINUS er2
   LINE MINUS x_begin s hd1 PLUS
   MINUS x_begin s d hd1
   SET_ER_PLUS er2
   SET_ER_MINUS er3
   SET_SUBINTERVALS si_diel
   LINE 0. hd2 PLUS MINUS x_begin s d hd2

SET_VARIABLE "conf" GET_CONFIGURATI
ON_2D
SET_VARIABLE "smn" SMN_C conf
SET_VARIABLE "cm" CALCULATE_C smn
conf
SET_VARIABLE "smn_l0" SMN_L0 conf
SET_VARIABLE "cm_l0" CALCULATE_L0
smn_l0 conf
SET "kc" DIV MINUS 0. GET_MATRIX_
VALUE
cm 0 1 GET_MATRIX_VALUE cm 0 0
SET "kl" DIV GET_MATRIX_VALUE
cm_l0 0 1 GET_MATRIX_VALUE cm_l0 0
0
IF LESS MINUS kc kl save_kc_kl
THEN SET "save_conf" conf
THEN SET "save_kc_kl" MINUS kc kl

END_CREATE_KEYWORD MINUS kc kl

SET "save_kc_kl" 1.e+6

SET "time" REPORT_TIMER
GA_MIN 10 10 0.1 0.6 1.e-6 1. "qf"

ECHO FORMAT_STRING sdtsd hd2/w=
GET_BEST_GA_PARAMETER 0 (kc-kl)=
GET_BEST_GA_RESULT

ECHO FORMAT_STRING dtdtd time
GET_BEST_GA_PARAMETER 0
GET_BEST_GA_RESULT

DRAW_CONFIGURATION save_conf
```

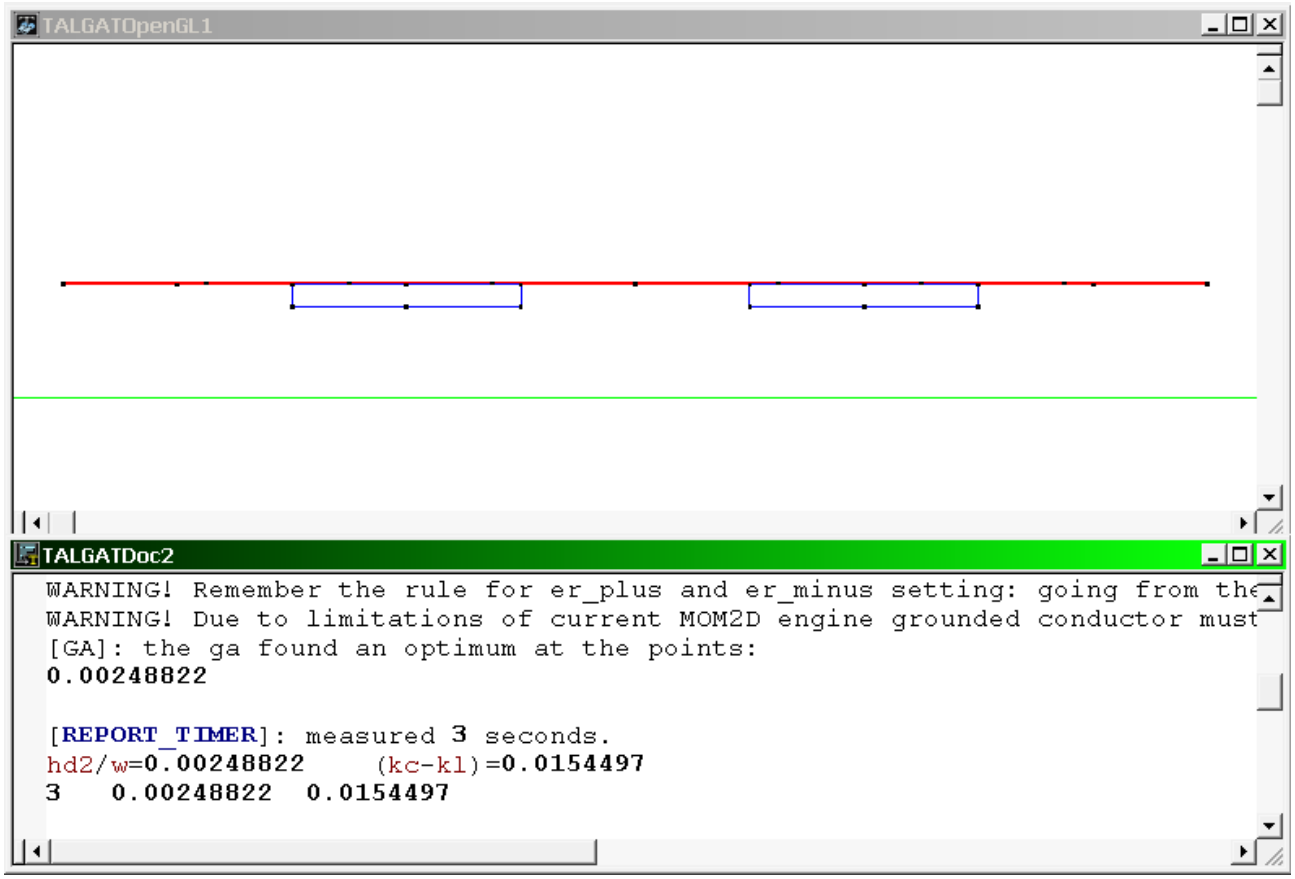


Рисунок 5.47 – Результат выполнения скрипта из листингов 5.38 и 5.39

5.2.1.6 Структурная оптимизация

Задача: минимизировать z-составляющую напряженности электрического поля в точке (центр источника_x, 100, центр источника_z) путем изменения количества пассивных диполей в структуре. Исходные данные: начальное число диполей по оси X – 5, по оси Z – 3, число диполей с источником питания по оси X – 2, по оси Z – 1, высота проводов – 5, частота 30 МГц, радиус дальней зоны – 100.

В листинге 5.40 создаются две динамических команды. Первая — do_source, создает на основе заданных параметров диполь с источником питания. Вторая – do_dipole, создает на основе заданных параметров пассивный диполь при условии, что параметр функции качества ГА с соответствующим номером больше 0. Обратите внимание, что из-за ограниченной ширины страницы некоторые строки скрипта были перенесены. Однако в системе TUSUR.EMC эти строки не должны содержать переносов.

Листинг 5.40 – Создание динамических команд do_source и do_dipole

```
INCLUDE "MOMW"

SET "num_dipoles_by_x" 5
SET "where_calc_E_x" x
SET "where_calc_E_z" PLUS z MINUS
DIV wire_height 2. DIV wire_height 100.
```

```
SET "num_dipoles_by_z" 3
SET "source_dipole_x" 2
SET "source_dipole_z" 1

SET "wire_height" 5.
SET "btw_wire_x" 5.
SET "btw_wire_z" 5.
SET "num_subsections" 10
SET "btw_wire_z" PLUS btw_wire_z
wire_height

SET "PI_method" PIA126_127
SET "f" 30.e+6
SET "far_zone_radius" 100.

SET_SUBSECTIONS_ON_WAVE 1
SET_SUBSECTIONS_MINIMAL
num_subsections
SET "source_dipole_subsections" PLUS
MUL num_subsections 2 1

CLEAR_STRUCTURE

RADIUS 5.e-3

CREATE_KEYWORD "do_source"

SET_SUBSECTIONS num_subsections

SET "x" MUL TO_DOUBLE
source_dipole_x btw_wire_x
SET "z" MUL TO_DOUBLE
source_dipole_z btw_wire_z
BEGIN x 0. z
END x 0. PLUS z MINUS DIV wire_height 2.
DIV wire_height 100.
CREATE_WIRE
BEGIN x 0. PLUS z MINUS DIV wire_height 2.
DIV wire_height 100.
END x 0. PLUS z PLUS DIV wire_height 2.
DIV wire_height 100.
EXCITATION (1.,0)
CREATE_WIRE
EXCITATION (0.,0.)

SET_SUBSECTIONS num_subsections
BEGIN x 0. PLUS z PLUS DIV wire_height
2.
DIV wire_height 100.
END x 0. PLUS z wire_height
CREATE_WIRE

END_CREATE_KEYWORD

CREATE_KEYWORD "do_dipole"

SET "call_create" 1
BEGIN x 0. z
END x 0. PLUS z wire_height
IF EQU source_dipole_z num_z
THEN IF EQU source_dipole_x num_x
THEN SET "call_create" 0
IF LESS GET_VARIABLE PLUS
GA_PARAM_ TO_STRING w_num 0.
THEN SET "call_create" 0
SET "w_num" PLUS w_num 1

IF EQU call_create 1
THEN CREATE_WIRE

SET "x" PLUS x btw_wire_x
SET "num_x" PLUS num_x 1
IF EQU num_x num_dipoles_by_x
THEN SET "num_x" 0
THEN SET "num_z" PLUS num_z 1
THEN SET "x" 0.
THEN SET "z" PLUS z btw_wire_z

END_CREATE_KEYWORD
```

В листинге 5.41 создается функция качества qf. В ней происходит вызов в цикле динамической команды do_dipole, которая на основе значений параметров функции качества GA_PARAM_n создает или не создает пассивные диполи с номерами n. Команда GA_MIN_STRUCTURAL оптимизирует структуру. Параметры:

18. размер популяции;
19. число поколений;

20. коэффициент мутаций;
21. коэффициент кроссовера;
22. число аргументов num_vars;
23. функция качества.

Обратите внимание, что из-за ограниченной ширины страницы некоторые строки скрипта были перенесены. Однако в системе TUSUR.EMC эти строки не должны содержать переносов. Результаты приведены на рисунке 5.48.

Листинг 5.41 – Задание функции качества и структурная оптимизация

```
CREATE_KEYWORD "qf"

CLEAR_STRUCTURE

SET "num_x" 0
SET "num_z" 0
SET "x" 0.
SET "z" 0.
SET "w_num" 1
CYCLE MUL num_dipoles_by_x
num_dipoles_by_z do_dipole
do_source

SET "structure" GET_STRUCTURE
SET "subsec" CREATE_SUBSECTIONS f
structure
SET "exc_vec" GET_EXCITATION_VECTOR
SET "imp_m"
CALCULATE_IMPEDANCE_MATRIX
f PI_method subsec

SET "currents" LU_SOLVE LU_FACT imp_m
exc_vec
SET "e" ABS GET_MATRIX_VALUE
CALCULATE_E_FAR_ZONE f currents subsec
where_calc_E_x far_zone_radius
where_calc_E_z 0 2
IF LESS e save_e
THEN SET "save_e" e
THEN ECHO FORMAT_STRING
sbd minimized save_e
THEN SET "save_subsec" subsec
THEN SET "save_currents" currents

END_CREATE_KEYWORD e

SET "pop_size" 5
SET "gen_num" 3

REPORT_TIMER
GA_MIN_STRUCTURAL pop_size
gen_num 0.6 MUL num_dipoles_by_x
0.1 num_dipoles_by_z "qf"
ECHO GET_BEST_GA_RESULT

DRAW_CURRENTS save_subsec
save_currents 0
```

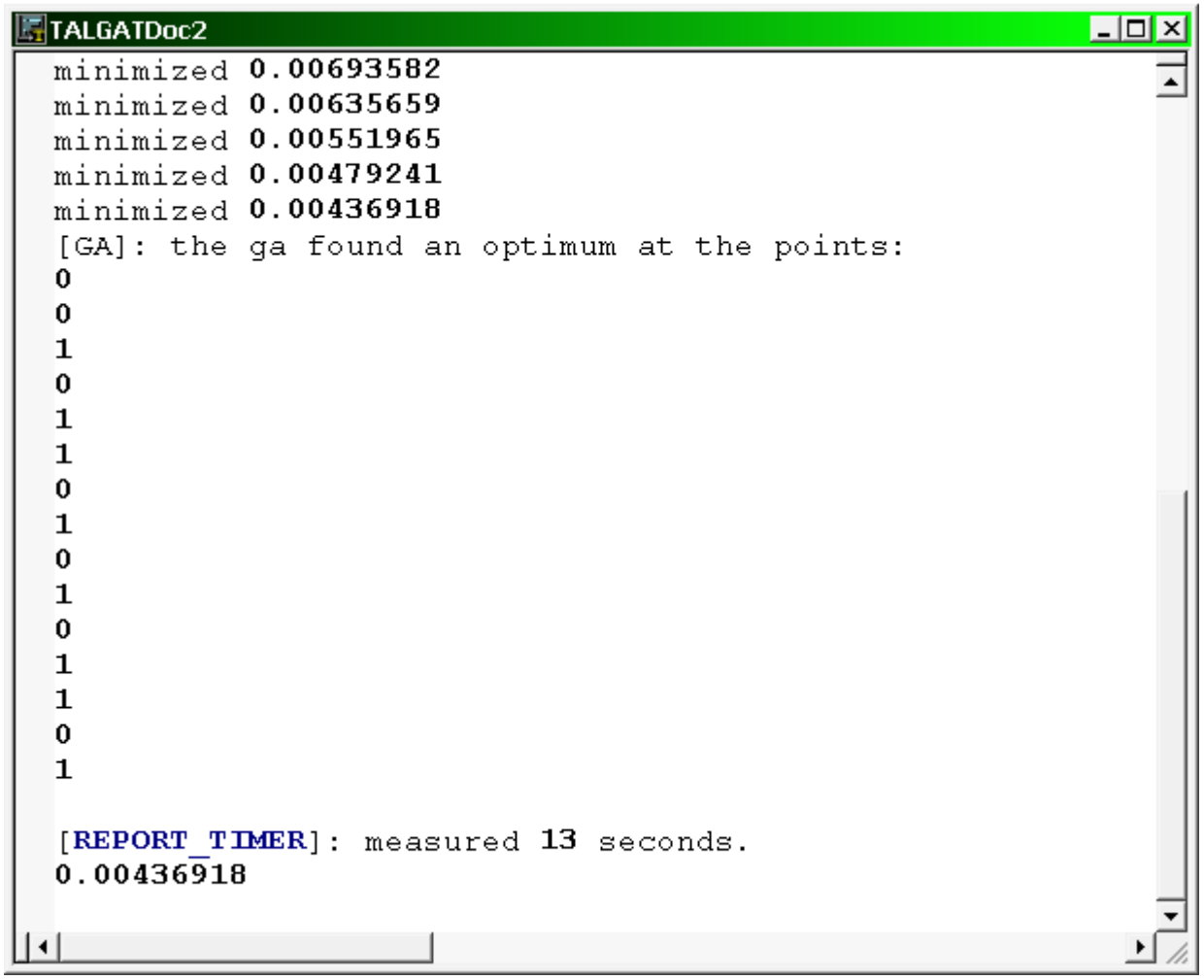


Рисунок 5.48 – Результат выполнения скрипта из листингов 5.40 и 5.41

В результате оптимизации получены следующие результаты: минимальное значение z-составляющей напряженности электрического поля в заданной точке составляет 0.00436918 при расположении диполей, приведённом на рисунке 5.49.

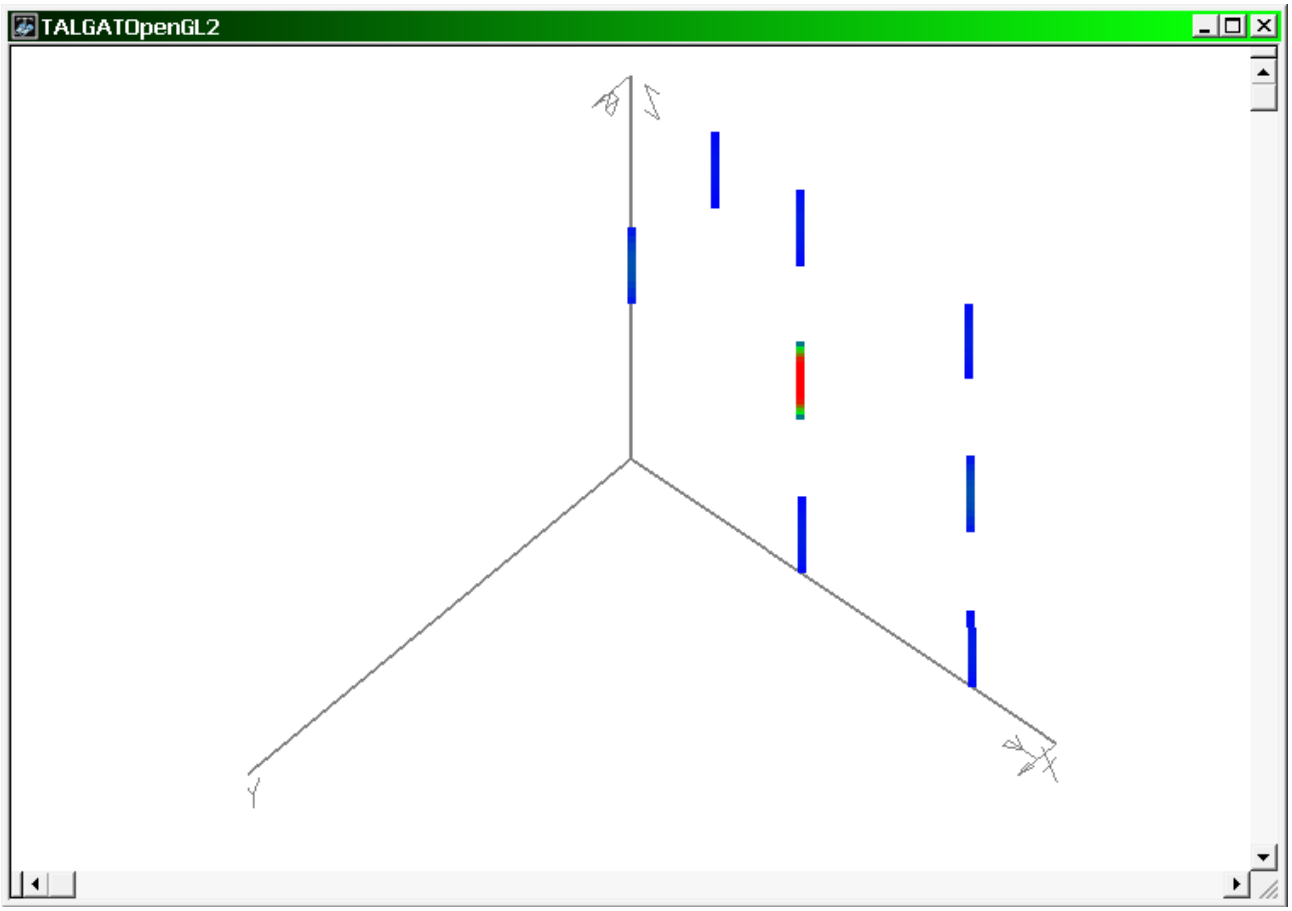


Рисунок 5.49 – Полученная в результате оптимизации структура

5.2.2 Модуль эволюционных стратегий ES

5.2.2.1 Оптимизация эволюционными стратегиями

После загрузки в ПО TUSUR.EMC модуля ЭС пользователю становится доступна команда ES_MIN, при вызове которой выполняется алгоритм (μ / μ I , λ) - σ - самоадаптационной ЭС. Команда ES_MIN имеет следующие параметры:

24. размер популяции родителей μ (рекомендуется μ = λ /3);
25. размер популяции потомков λ ;
26. количество поколений nGens ;
27. размерность пространства поиска N (равно количеству параметров целевой функции);
28. имя динамической команды, которая реализует целевую функцию, принимает N параметров (GA_PARAM_1, GA_PARAM_2, и так далее до GA_PARAM_N) и возвращает вещественное число (число с плавающей точкой).

Целевая функция строится на скриптовом языке по тем же правилам, что и для модуля ГА, поэтому модуль ЭС можно использовать для оптимизации всех целевых функций, реализованных ранее для ГА. При этом нет необходимости вносить какие-либо изменения в сами целевые функции, достаточно изменить строку с командой, запускающей оптимизацию.

Для задания дополнительных настроек оптимизации предназначена команда ES_OPTS, которая принимает следующие параметры:

29. параметр самообучения τ (рекомендуется $\tau = 1/N$, если ввести $\tau = 0$, то команда ES_OPTS автоматически вычислит τ по этой формуле);
30. начальное значение величины мутации σ (рекомендуется $\sigma = 0.3$);
31. начальная точка поиска (зависит от целевой функции).

Для получения результатов оптимизации используются команды GET_BEST_ES_RESULT, GET_BEST_ES_PARAMETER и GET_BEST_ES_PARAMETER, которые имеют то же назначение и список параметров, что и аналогичные команды GET_BEST_GA_RESULT,
GET_BEST_GA_PARAMETER, GET_BEST_GA_PARAMETER из модуля GA.

Пример оптимизации ЭС целевой функции приведен в листинге 5.42.

Листинг 5.42 – Пример оптимизации ЭС целевой функции

```
INCLUDE "ES"
INCLUDE "UTIL"
INCLUDE "MATRIX"
INCLUDE "INFIX"

SET "nGens" 10
SET "aN" 10
SET "y" CREATE_REAL_MATRIX 1 aN // simple sphere fitness function F(y) = norm(y)
CREATE_KEYWORD "Sphere"
SET "y" SET_MATRIX_ROW y 0 GA_PARAM_1 GA_PARAM_2 GA_PARAM_3
GA_PARAM_4 GA_PARAM_5 GA_PARAM_6 GA_PARAM_7
GA_PARAM_8 GA_PARAM_9 GA_PARAM_10
SET "sum" 0.
FOR "i" 0 MINUS aN 1 1 SET "sum" PLUS sum MUL GET_MATRIX_VALUE y 0 i
GET_MATRIX_VALUE y 0 i
SET_INFIX_VARIABLE "sum" sum
END_CREATE_KEYWORD INFIX sqrt(sum)
ES_OPTS 0 1 10.
ECHO TIMER ES_MIN 3 10 nGens aN "Sphere"
ECHO GET_BEST_ES_RESULT
ECHO GET_BEST_ES_PARAMETER 0
ECHO GET_BEST_ES_PARAMETER MINUS aN 1
```

5.2.2.2 Алгоритм неявного фильтрования

Для оптимизации с помощью алгоритма неявного фильтрования в модуле ES реализована команда IF_MIN. Команда IF_MIN имеет следующие параметры:

32. количество элементов в массиве шкал nScales (рекомендуется nScales=10);
33. максимальное количество итераций nIter (по смыслу аналогично количеству поколений nGens);
34. размерность пространства поиска N (равно количеству параметров целевой функции);
35. имя динамической команды, которая реализует целевую функцию.

Для задания дополнительных настроек оптимизации предназначена команда IF_OPTS, которая принимает следующие параметры:

36. вид разностной аппроксимации градиента (0 – центральный, рекомендуемый; 1 – форвардный);
37. начальная точка поиска (зависит от целевой функции).

Пример оптимизации с помощью алгоритма неявного фильтрования приведен в листинге 5.43.

Листинг 5.43 – Пример оптимизации с помощью алгоритма неявного фильтрования

```
IF_OPTS 0 10.
ECHO TIMER IF_MIN 4 50 aN "Sphere"
ECHO GET_BEST_IF_RESULT
ECHO GET_BEST_IF_PARAMETER 0
ECHO GET_BEST_IF_PARAMETER MINUS aN 1
```

5.3 Модули утилит

5.3.1 Модуль команд общего назначения UTIL

5.3.1.1 Общие сведения

Модуль UTIL предназначен для использования команд общего назначения. Перед началом работы с модулем UTIL необходимо загрузить его (листинг 5.44).

Листинг 5.44 – Загрузка модуля UTIL

```
INCLUDE "UTIL"
```

5.3.1.2 Математические команды

После загрузки модуля UTIL пользователю становятся доступны следующие математические команды.

Команда EQU сравнивает два аргумента: если они равны, возвращает 1, иначе – 0.

Команда GREATER сравнивает два аргумента: если первый больше, то возвращает 1, если меньше – 0, если оба аргумента равны – 0.

Команда LESS сравнивает два аргумента: если первый меньше, то возвращает 1, если больше – 0, если оба аргумента равны – 0.

Команды PLUS, MUL и DIV производят операции сложения, умножения и деления двух своих параметров.

Команда ABS возвращает модуль своего параметра.

Важные замечания:

Все математические команды модуля UTIL перегружены для поддержки всех типов данных. В качестве параметров данные команды могут принимать следующие стандартные типы данных: пустой (NULL), long, double, string, complex. Кроме того, система TUSUR.EMC позволяет включить в этот список типы данных, которые реализованы в других модулях системы. Таким образом, после загрузки модуля MATRIX данные команды работают также с типами данных real_matrix, complex_matrix.

Заметим, что это касается лишь сочетаний параметров, имеющих смысл. Так, не будет работать «MUL строка1 строка2».

Разные типы параметров генерируют исключение. Для сравнения разнотипных параметров используйте команды преобразования типов (разд. 5.3.1.3), например: «EQU 10. TO_DOUBLE 10».

Более сложные математические функции реализованы в модуле инфиксной записи выражений INFIX (разд. 5.3.3).

Пример использования математических команд показан в листинге 5.45. Результат выполнения данного скрипта приведен на рисунке 5.50.

Листинг 5.45 – Пример использования математических команд

```
ECHO EQU text text
ECHO GREATER 10. 11.
ECHO LESS 11. 12.
ECHO PLUS this_is_ a_test
ECHO MUL (1.,1.) (2.,2.)
ECHO DIV (1.,1.) (2.,2.)
ECHO ABS (1.,1.)
```

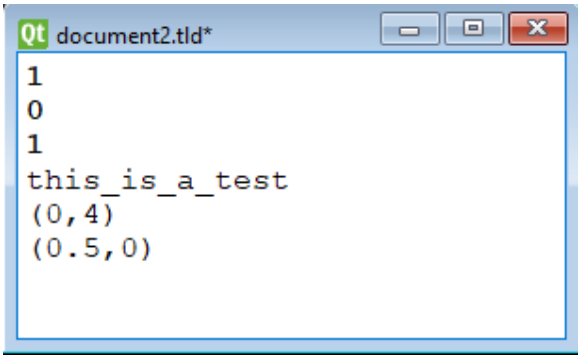


Рисунок 5.50 – Результат выполнения скрипта из листинга 5.45

5.3.1.3 Преобразования типов

Можно использовать четыре команды приведения типов: TO_STRING, TO_LONG, TO_DOUBLE и TO_COMPLEX, выполняющих, соответственно, приведение к строке, к целому числу, к числу с плавающей точкой и к комплексному числу. Также модуль UTIL реализует две команды REAL и IMAG для выделения вещественной и мнимой частей комплексного числа.

Важные замечания:

- Хотя из разд. 4.2.2.1 следует, что использовать в качестве параметра, тип которого должен быть, например, double, параметр типа long нельзя, это ограничение можно ЧАСТИЧНО обойти с помощью команд преобразования типов.
- Кроме того, с помощью команды TO_STRING осуществляется форматированный вывод всех типов данных. Она так же перегружена для поддержки всех типов данных, как и математические команды. Использование данной команды для форматированного вывода матриц описано в разд. 5.3.2.3.
- Неверные преобразования типов генерируют исключение (рисунок 5.51), например: «TO_LONG not_a_number», «TO_DOUBLE (1.,1.)».

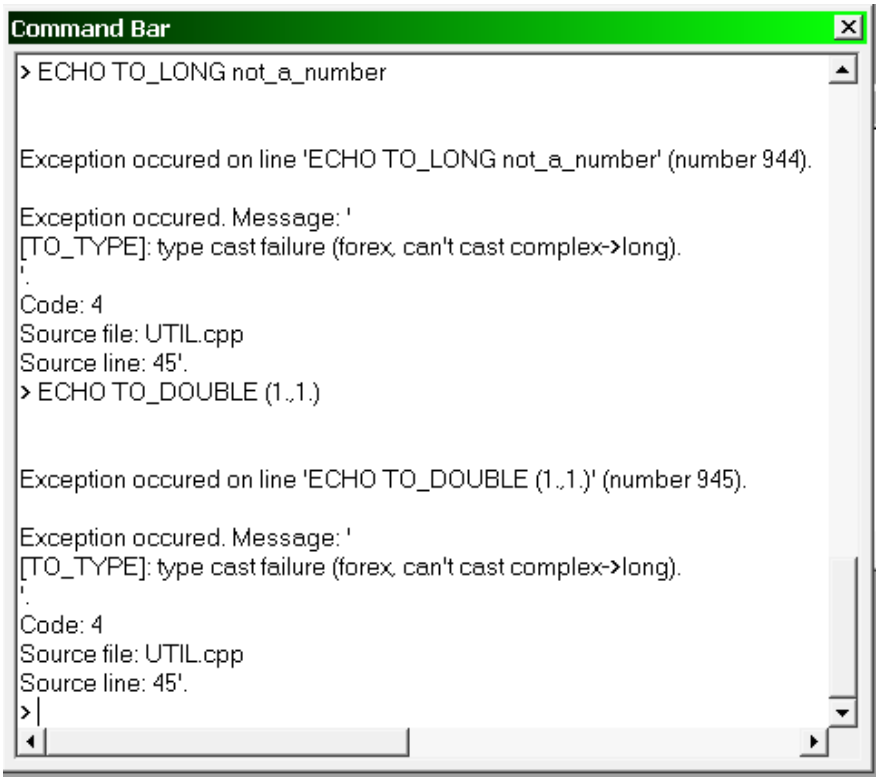


Рисунок 5.51 – Исключения в результате выполнения неверных преобразований типов

Пример использования команд преобразования типов показан в листинге 5.46, а результат его выполнения – на рисунке 5.52.

Листинг 5.46 – Пример использования команд преобразования типов

```
ECHO TO_STRING (1.,1.)
ECHO TO_COMPLEX 10
ECHO TO_LONG 1.
ECHO TO_DOUBLE 1
ECHO REAL (1,2)
ECHO IMAG (1,2)
```

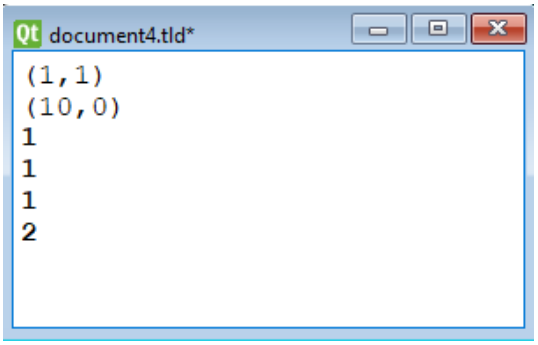


Рисунок 5.52 – Результат выполнения скрипта из листинга 5.46

Кроме того, команды REAL и IMAG могут также принимать в качестве параметра комплексные матрицы, возвращая в виде результата действительную матрицу, соответственно, с реальной или мнимой частью элементов из исходной матрицы.

5.3.1.4 Условные команды

Для проверки условий можно использовать конструкцию «IF выражение THEN команда». Если выражение истинно, то инструкция после команды THEN выполнится. Пример использования условных команд показан в листинге 5.47, а результат его выполнения – на рисунке 5.53.

Листинг 5.47 – Пример использования условных команд

```
IF LESS 1 2
THEN ECHO 1_is_less_than_2
IF EQU 1 2
THEN ECHO its_impossible
```

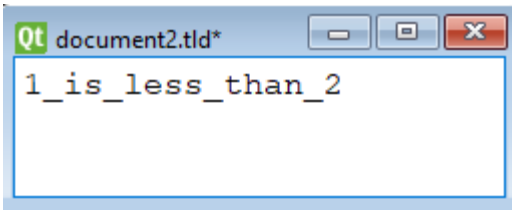


Рисунок 5.53 – Результат выполнения скрипта из листинга 5.47

Для более сложных случаев следует использовать конструкцию «IF выражение THEN команда ELSE команда». С помощью команды ELSE можно упростить листинг на рисунке 5.53 следующим образом (листинг 5.48).

Листинг 5.48 – Пример использования команды ELSE

```
IF LESS 1 2
THEN ECHO 1_is_less_than_2
ELSE ECHO its_impossible
```

5.3.1.5 Файловый ввод-вывод

Открытие файла для записи осуществляется командой OPEN_FOR_WRITE, параметром которой является имя файла. Вывод в файл осуществляется командой WRITE, в качестве параметра передаётся выражение, которое нужно вывести. Если же использовать команду WRITE с параметром NEW_LINE, то произойдёт перевод каретки на новую строку. Команда CLOSE осуществляет закрытие файла.

Открытие файла для считывания осуществляется командой OPEN_FOR_READ. Чтение осуществляется командой READ, которая возвращает прочитанную строку. Закрытие также производится командой CLOSE.

Важные замечания:

- Одновременно могут быть открыты один файл для чтения и один файл для записи.
- При достижении конца файла команда READ возвращает строку «[EOF]».

Пример использования файлового ввода-вывода показан в листинге 5.49, а результат его выполнения – на рисунках 5.54 и 5.55.

Листинг 5.49 – Пример использования файлового ввода-вывода

```
OPEN_FOR_WRITE test.txt
WRITE file_io_test
WRITE NEW_LINE
WRITE (1,2)
CLOSE
OPEN_FOR_READ test.txt
ECHO READ
ECHO READ
IF EQU READ [EOF]
THEN ECHO end_of_file_is_detected
CLOSE
```

Рисунок 5.54 – Результат выполнения скрипта из листинга 5.49

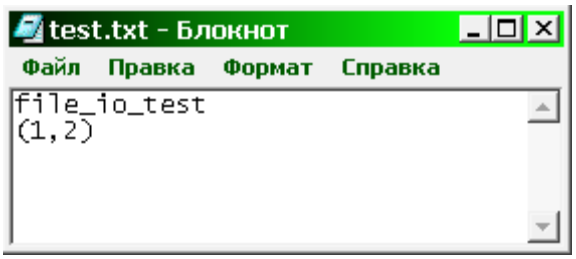


Рисунок 5.55 – Файл, созданный в результате выполнения скрипта из листинга 5.49

5.3.1.6 Проверка версий

Команда ECHO, с параметром GET_CORE_VERSION, возвращает номер версии ядра системы.

Команда CHECK_CORE_VERSION с параметром 110000 проверяет, используется ли ядро версии 1.10xxx. Если номер версии отличается от заданного параметром команды CHECK_CORE_VERSION, то выводится соответствующее сообщение.

5.3.1.7 Команды циклов

Команда CYCLE вычисляет или выполняет свой параметр заданное количество раз.

Команда FOR имеет пять параметров и работает так:

38. создает переменную с именем в виде первого параметра команды FOR – счетчик;
39. присваивает счетчику второй параметр – начальное значение;
40. вычисляет пятый параметр – тело цикла, если в качестве пятого параметра передается имя динамической команды – вызывает динамическую команду;
41. прибавляет к счетчику четвертый параметр – значение шага;
42. если счетчик меньше, либо равен третьему параметру – конечному значению, переходит на шаг 3.

Обратите внимание, что имя переменной заключено в кавычки, а второй, третий и четвертый параметры имеют тип double. Пример использования команд цикла показан на листинге 5.50, а результат его выполнения – на рисунке 5.56

Листинг 5.50 – Пример использования команд цикла

```
CYCLE 3 ECHO cycle
FOR "num" 1. 3. 1. ECHO num
```

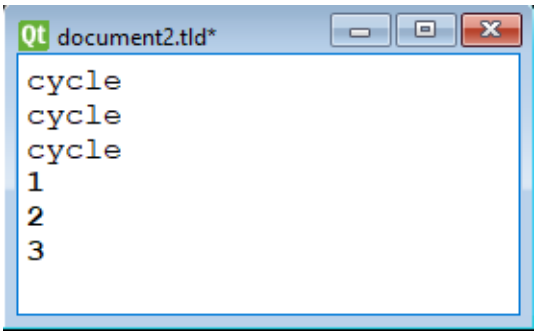


Рисунок 5.56 – Результат выполнения скрипта из листинга 5.50

Команду FOR можно также вызывать с параметрами типа long, например, FOR "num" 1 3 1 ECHO num.

Для построения графика функции в диапазоне параметров удобно использовать команду FOR_RANGE. Команда FOR_RANGE создает матрицу, в первую строку которой записывает значения счетчика, а во вторую – результаты выполнения тела цикла (пятый параметр команды FOR_RANGE). На листинге 5.51 – Пример использования команды FOR_RANGE приведен пример построения графика функции mulFloat (param) =2 · param (описание команд для построения графиков смотрите в разделе о модуле построения графиков GRAPH).

Листинг 5.51 – Пример использования команды FOR_RANGE

```
CREATE_KEYWORD "mulFloat"
   SET "retval" MUL param 2.0
END_CREATE_KEYWORD retval
SET "m1" FOR_RANGE "param" 1 10 1 mulFloat
INCLUDE "GRAPH"
ADD_XY_DATA_r GET_ROW m1 0 GET_ROW m1 1
PLOT_XY
```

5.3.1.8 Форматированный вывод

Команда ECHO с параметром NEW_LINE возвращает символ новой строки.

Команда SET_FORMAT устанавливает режим вывода чисел типа double. Возможные параметры: [AUTO], [SCIENTIFIC], [FIXED]. По умолчанию стоит режим [AUTO].

Команда GET_FORMAT возвращает значение режима вывода чисел типа double.

Команда SET_PRECISION устанавливает количество знаков после запятой, которые выводятся в режиме [FIXED] (по умолчанию их 6).

Команда GET_PRECISION возвращает количество знаков после запятой в режиме [FIXED].

Команда LINE_TO_STRING переводит все символы строки в тип string, включая пробелы и табуляцию.

Команда FORMAT_STRING - аналог функции printf() в C. Первый параметр - строка с символами форматирования в любом количестве и последовательности:

- t – табуляция;
- b – пробел;
- n - новая строка;
- l – long;
- d – double;
- s – string;
- c – complex;
- \ – не считать следующий символ флагом (например, \c – получаем символ c).

Пример использования форматированного вывода показан в листинге 5.52, а результат его выполнения – на рисунке 5.57.

Листинг 5.52 – Пример использования форматированного вывода

```
ECHO NEW_LINE
ECHO two_new_line_symbols_before_and_one_after
ECHO NEW_LINE
```

```
SET_FORMAT [SCIENTIFIC]
ECHO 0.1
ECHO GET_FORMAT

SET_FORMAT [FIXED]
ECHO GET_PRECISION
ECHO 0.1
SET_PRECISION 10
ECHO 0.1

SET_FORMAT [AUTO]
ECHO 0.1
ECHO this is a line with spaces and tabs
ECHO LINE_TO_STRING line with spaces and tabs
ECHO FORMAT_STRING nts:tdn Number 10.
```

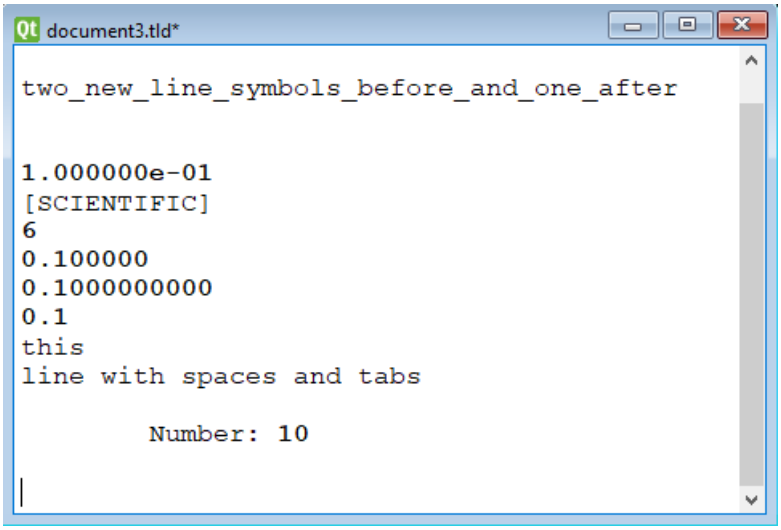


Рисунок 5.57 – Результат выполнения скрипта из листинга 5.52

5.3.1.9 Измерение времени

Команда TIMER возвращает время в секундах, которое было затрачено для вычисления выражения, указанного в качестве параметра. Используется в виде «ECHO TIMER выражение».

Команда REPORT_TIMER аналогична «ECHO TIMER», но выводит результат в отформатированном виде (листинг 5.53 и рисунок 5.58).

Листинг 5.53 – Пример использования команд измерения времени

```
ECHO TIMER CYCLE 50000 ABS (100,100)
REPORT_TIMER CYCLE 50000 ABS (100,100)
```

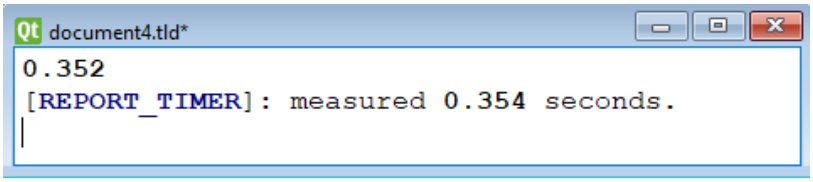


Рисунок 5.58 – Результат выполнения скрипта из листинга 5.53

5.3.1.10 Другие команды

Команда GET_DIRECTORY возвращает в виде строки папку, в которой расположены исполняемые файлы системы. Например, «C:\Program Files\MyCatalog\».

5.3.2 Модуль работы с матрицами MATRIX

5.3.2.1 Общие сведения

Модуль MATRIX предназначен для работы с матрицами. Перед началом работы с модулем его необходимо загрузить «INCLUDE "MATRIX"», также необходимо загрузить модуль утилит «INCLUDE "UTIL"».

Матрицы в системе TUSUR.EMC относятся к типу данных matrix. При этом матрицы из чисел типа double номинально обозначаются как тип real_matrix, матрицы из чисел типа complex - как тип complex_matrix. Если команда требует параметр типа matrix, то ей можно передавать параметры типа real_matrix и complex_matrix. Если команда явно требует параметр типа real_matrix, то ей нельзя передавать параметр типа complex_matrix, и наоборот.

Индексация столбцов и строк матриц начинается с нуля. Векторы в системе представляются матрицами с одной строкой.

В данном сборнике методических материалов приведены основные сведения о модуле MATRIX. Для получения дополнительной информации по командам для решения СЛАУ обратитесь к методическим указаниям по использованию методов решения СЛАУ.

5.3.2.2 Работа с матрицами

Команда CREATE_REAL_MATRIX служит для создания матриц из чисел double, а команда CREATE_COMPLEX_MATRIX создаёт матрицы из чисел complex. Число аргументов обеих команд два, первый — число строк, второй — число столбцов матрицы.

Команды GET_MATRIX_ROWS и GET_MATRIX_COLS возвращают число строк и столбцов матрицы, указанной в качестве аргумента.

Команда GET_MATRIX_SIZE выводит в форматированном виде (строка) размер матрицы, указанной в качестве аргумента.

Команда SET_MATRIX_VALUE имеет четыре аргумента: матрица, индекс строки, индекс столбца и присваиваемое значение. Пример использования данных команд приведён в листинге 5.54 и на рисунке 5.59.

Листинг 5.54 – Создание и инициализация матриц

```
SET "real_m" CREATE_REAL_MATRIX 2 2
SET "complex_m" CREATE_COMPLEX_MATRIX 2 2
ECHO GET_MATRIX_ROWS real_m
ECHO GET_MATRIX_COLS real_m
ECHO GET_MATRIX_ROWS complex_m
ECHO GET_MATRIX_COLS complex_m
ECHO GET_MATRIX_SIZE real_m
ECHO GET_MATRIX_SIZE complex_m
SET "real_m" SET_MATRIX_VALUE real_m 0 0 1.
SET "real_m" SET_MATRIX_VALUE real_m 0 1 2.
SET "real_m" SET_MATRIX_VALUE real_m 1 0 3.
SET "real_m" SET_MATRIX_VALUE real_m 1 1 4.
SET "complex_m" SET_MATRIX_VALUE complex_m 0 0 (1,1)
SET "complex_m" SET_MATRIX_VALUE complex_m 0 1 (2,2)
SET "complex_m" SET_MATRIX_VALUE complex_m 1 0 (3,3)
SET "complex_m" SET_MATRIX_VALUE complex_m 1 1 (4,4)
```

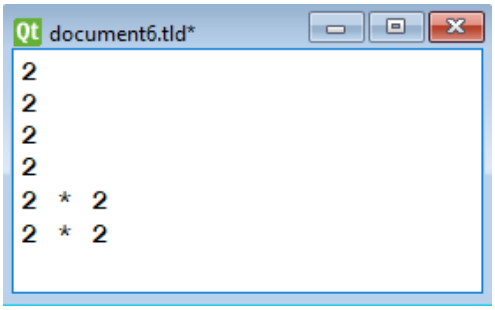


Рисунок 5.59 – Результат выполнения скрипта из листинга 5.54

Обратите внимание, что элементам матрицы типа real_matrix мы присваиваем значения типа double, а элементам матрицы типа complex_matrix — значения типа complex.

5.3.2.3 Форматированный вывод матриц

С помощью команды TO_STRING из модуля UTIL осуществляется форматированный вывод всех типов данных, в том числе матриц. Обратите внимание: команда TO_STRING работает с матрицами только при загруженном модуле MATRIX.

В некоторых случаях требуется вывести элементы матрицы без форматирования, в одну строку (как вектор). В этом случае используйте команду

SET_PRINT_MATRIX_AS_VECTOR с параметром 1. После использования всегда выключайте данную опцию (та же команда с параметром 0).

Пример форматированного вывода матриц показан в листинге 5.55 и на рисунке 5.60.

Листинг 5.55 – Форматированный вывод матриц

```
ECHO LINE_TO_STRING our real matrix:
ECHO TO_STRING real_m
ECHO LINE_TO_STRING our complex matrix:
ECHO TO_STRING complex_m
ET_PRINT_MATRIX_AS_VECTOR 1
ECHO TO_STRING real_m
ECHO TO_STRING complex_m
SET_PRINT_MATRIX_AS_VECTOR 0
```

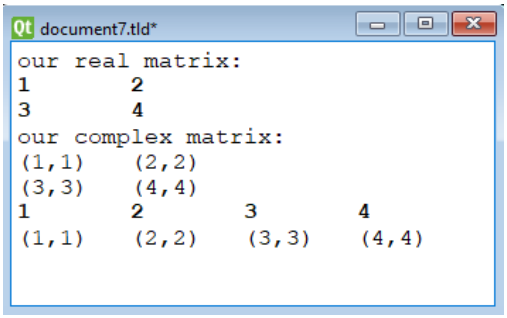


Рисунок 5.60 – Результат выполнения скрипта из листинга 5.55

5.3.2.4 Решение СЛАУ

Хотя справка по командам для решения СЛАУ говорит, что данные команды принимают матрицы любого типа, в данной версии системы команды для решения СЛАУ реализованы только для матриц типа complex_matrix.

Инициализируем вектор (однострочную матрицу) свободных членов right_m и сравним разные методы решения СЛАУ.

43. Метод LU-факторизации. Сначала факторизуем левую матрицу СЛАУ командой LU_FACT, в качестве аргумента указываем матрицу complex_m из предыдущего примера. Затем находим решение командой LU_SOLVE. Параметры команды LU_SOLVE:

44. факторизованная левая матрица СЛАУ;

45. вектор (однострочная матрица) свободных членов.

46. Метод BICGSTABPRE. Решение находится командой BICGSTABPRE. Если один из указанных параметров равен нулю, то используется значение по умолчанию. Список параметров команды BICGSTABPRE (в скобках указано значение по умолчанию):

47. левая матрица СЛАУ;

48. вектор (однострочная матрица) свободных членов;

49. double, точность вычислений (1.e-8);
50. long, максимальное количество итераций (400);
51. long, тип предобусловливания: NONE - 0, SGS - 1, ILU - 2 (NONE);
52. double, допуск обнуления при предобусловливании ILU (1.e-3).

Пример использования команд для решения СЛАУ приведён в листинге 5.56 (левая матрица СЛАУ объявлена в листинге 5.54), а результат его выполнения – на рисунке 5.61.

Листинг 5.56 – Три метода решения СЛАУ

```
SET "right_m" CREATE_COMPLEX_MATRIX 1 2
SET "right_m" SET_MATRIX_VALUE right_m 0 0 (1,0)
SET "right_m" SET_MATRIX_VALUE right_m 0 1 (2,0)
SET "lu_fact" LU_FACT complex_m
SET "lu_solve" LU_SOLVE lu_fact right_m

SET "bicgstab_solve" BICGSTAB complex_m right_m 0.1.e-6 10 ILU0nr 1.e-1
ECHO TO_STRING lu_solve
ECHO TO_STRING bicgstab_solve
```

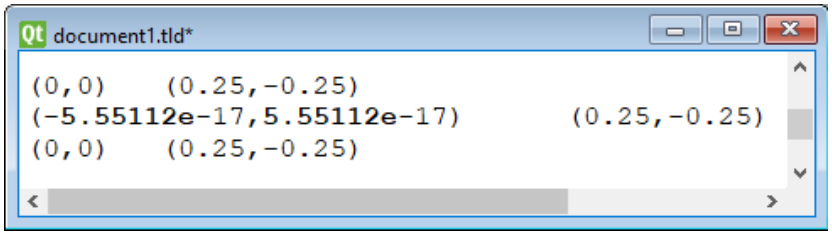


Рисунок 5.61 – Результат выполнения скрипта из листинга 5.56

5.3.2.5 Вырезание строк и столбцов матриц

Команда REMOVE_MATRIX_ROW удаляет строку матрицы, указанной в качестве первого параметра. Индекс строки — второй параметр. Аналогично удаляются и столбцы при помощи команды REMOVE_MATRIX_COL. Пример приведён в листинге 5.57 и на рисунке 5.62.

Листинг 5.57 – Пример использования команд вырезания

```
ECHO LINE_TO_STRING remove first row from real_m and second column from complex_m
SET "cut_real_m" REMOVE_MATRIX_ROW real_m 0
SET "cut_complex_m" REMOVE_MATRIX_COL complex_m 1
ECHO TO_STRING cut_real_m
ECHO TO_STRING cut_complex_m
```

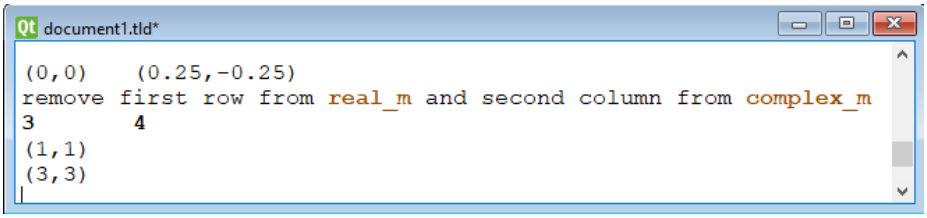


Рисунок 5.62 – Результат выполнения скрипта из листинга 5.57

5.3.3 Модуль инфиксной записи выражений INFIX

5.3.3.1 Общие сведения

Модуль INFIX предназначен для удобного задания математических выражений. Он позволяет пользователям системы TUSUR.EMC находить значения математических выражений в привычной инфиксной форме (a+b)/2; создает байт-код каждого выполняемого выражения (так что многочисленное выполнение одного и того же выражения значительно быстрее, чем с помощью интерпретатора системы); работает только со значениями типа double (так что плавающая точка в целых числах не требуется).

Перед началом работы с модулем INFIX его нужно загрузить (листинг 5.58).

Листинг 5.58 – Подготовка к работе с модулем INFIX

```
INCLUDE "UTIL"
INCLUDE "INFIX"
```

5.3.3.2 Короткий пример использования инфиксных выражений

Команда INFIX обрабатывает свой строковый параметр (инфиксное выражение) и возвращает его значение. Пример использования команды INFIX приведён в листинге 5.59 и на рисунке 5.63.

Листинг 5.59 – Пример использования команды INFIX

```
ECHO INFIX 2+2*2
ECHO INFIX (1+1)^10
ECHO INFIX cos(pi/2)+ ln(5*e)
```

Рисунок 5.63 – Результат выполнения скрипта из листинга 5.59 не считает последнюю строку

5.3.3.3 Подробный пример использования инфиксных выражений в системе

Команда SET_INFIX_VARIABLE передаёт переменную в модуль INFIX. Первый параметр — имя переменной, второй параметр — значение.

Команда SET_INFIX вводит строковое выражение для вычисления. Команда RUN_INFIX вычисляет выражение, введённое командой SET_INFIX, при этом модуль INFIX генерирует байт-код и исполняет его. Можно использовать более короткий вариант записи, например: «SET_VARIABLE "my_double_variable" INFIX sqrt(my_infix_double)».

Команда GET_INFIX_VARIABLE возвращает значение инфиксной переменной. Параметр — имя переменной.

Команда GET_INFIX возвращает строковое представление последнего инфиксного выражения, введённого командами INFIX или SET_INFIX.

Пример использования инфиксных команд приведён в листинге 5.60 и на рисунке 5.64.

Листинг 5.60 – Пример использования инфиксных команд

```
SET_VARIABLE "my_double" 4.5
SET_INFIX_VARIABLE my_infix_double my_double
SET_INFIX_VARIABLE my_infix_double 4.5
SET_INFIX sqrt(my_infix_double*2)
ECHO RUN_INFIX
SET_VARIABLE "my_double" RUN_INFIX
SET_VARIABLE "my_double" INFIX sqrt(my_infix_double*2)
ECHO GET_INFIX_VARIABLE my_infix_double
ECHO GET_INFIX
```

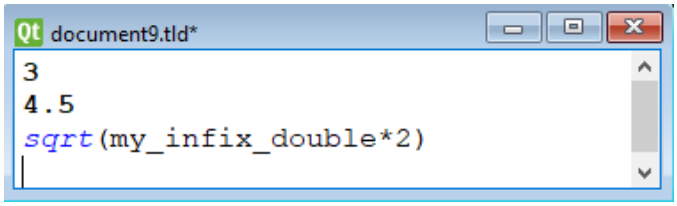


Рисунок 5.64 – Результат выполнения скрипта из листинга 5.60

5.3.3.4 Список поддерживаемых системой инфиксных функций и операторов

Список поддерживаемых инфиксных функций приведён в таблице 5.2. Обратите внимание, что тригонометрические функции работают с радианами.

Таблица 5.2 – Поддерживаемые функции

Функция	Описание
abs	модуль
acos	арккосинус
acosh	гиперболический арккосинус
asin	арксинус
asinh	гиперболический арксинус
atanh	гиперболический арктангенс
atan	арктангенс
cos	косинус
cosh	гиперболический косинус
exp	экспонента
ld	логарифм по основанию 2
lg	логарифм по основанию 10
ln	логарифм по основанию e (2.71828...)
log	логарифм по основанию 10
rint	округление до ближайшего целого
sign	Функция знака { -1, x<0 1, x>0
sin	синус
sinh	гиперболический синус
sqrt	квадратный корень
tan	тангенс
tanh	гиперболический тангенс

В таблице 5.3 приведены поддерживаемые инфиксные операторы. Операторы с большим приоритетом выполняются в первую очередь. В модуле INFIX также заданы константы pi и e.

Таблица 5.3 – Поддерживаемые операторы

Оператор	Действие	Приоритет
<=	меньше либо равно	1
>=	больше либо равно	1
!=	не равно	1
==	равно	1
>	больше	1
+	сложение	2
-	вычитание	2
*	умножение	3
/	деление	3
^	степень	4

5.3.4 Модуль построения графиков GRAPH

5.3.4.1 Общие сведения

Модуль предназначен для построения и отображения графиков в системе TUSUR.EMC. Перед началом работы с модулем его необходимо загрузить «INCLUDE "QtClient.exe""GRAPH"»

5.3.4.2 Структура задания команд

53. Подключить модуль GRAPH.

54. Создать линию или линии с помощью команд ADD_XY_DATA_c и ADD_XY_DATA_r.

55. После создания линии можно описать ее параметры (надпись, цвет, стиль, ширину) при этом параметры описываются строго после создания линии, к которой эти параметры должны принадлежать, в произвольном порядке. Если параметры не будут заданы, то линия будет иметь стандартные свойства.

56. Отобразить график PLOT_XY.

Таблица 5.4 – Команды модуля GRAPH

Команда	Описание команды	Описание параметров
ADD_XY_DATA_r x y	Вещественные данные для построения графика	x, y - массив данных типа REAL
ADD_XY_DATA_c x y z	Комплексные данные для построения графика	x, y - массив данных типа COMPLEX, z - способ отображения комплексного числа на графике: может принимать одно из значений из таблицы 5.5.
PLOT_XY	Построение графика	

Таблица 5.5 – Команды, определяющие способ отображения комплексного числа

Параметр	Использование массива x	Использование массива y
COMPLEX_PLOT_REAL	Действительная часть	Действительная часть
COMPLEX_PLOT_IMAG	Мнимая часть	Мнимая часть
COMPLEX_PLOT_ABS	Модуль	Модуль
COMPLEX_PLOT_REAL_IMAG	Действительная часть	Мнимая часть
COMPLEX_PLOT_IMAG_REAL	Мнимая часть	Действительная часть
COMPLEX_PLOT_ARG	Действительная часть	Аргумент (градусы)

5.3.4.3 Пример создания линий

Координаты линии задаются с помощью команд ADD_XY_DATA_c и ADD_XY_DATA_r. Пример создания линий приведён в листинге 5.61.

Описание параметров графика

Листинг 5.61 – Пример создания линии с помощью ADD_XY_DATA_r

```
INCLUDE "QtClient.exe""GRAPH"
INCLUDE "MATRIX"
//Пример создания линии с использованием команды ADD_XY_DATA_r :
// Создадим входные матрицы x и y. При построении используется первая
//строка матрицы.
SET "x" CREATE_REAL_MATRIX 1 4
SET "y" CREATE_REAL_MATRIX 1 4
SET "x" SET_MATRIX_VALUE x 0 0 1. //значения по x = {1,2,3,4}
SET "x" SET_MATRIX_VALUE x 0 1 2.
SET "x" SET_MATRIX_VALUE x 0 2 3.
SET "x" SET_MATRIX_VALUE x 0 3 4.
SET "y" SET_MATRIX_VALUE y 0 0 1.5//значения по y = {1.5, 0.5, 3.5, 2.5}
SET "y" SET_MATRIX_VALUE y 0 1 0.5
```

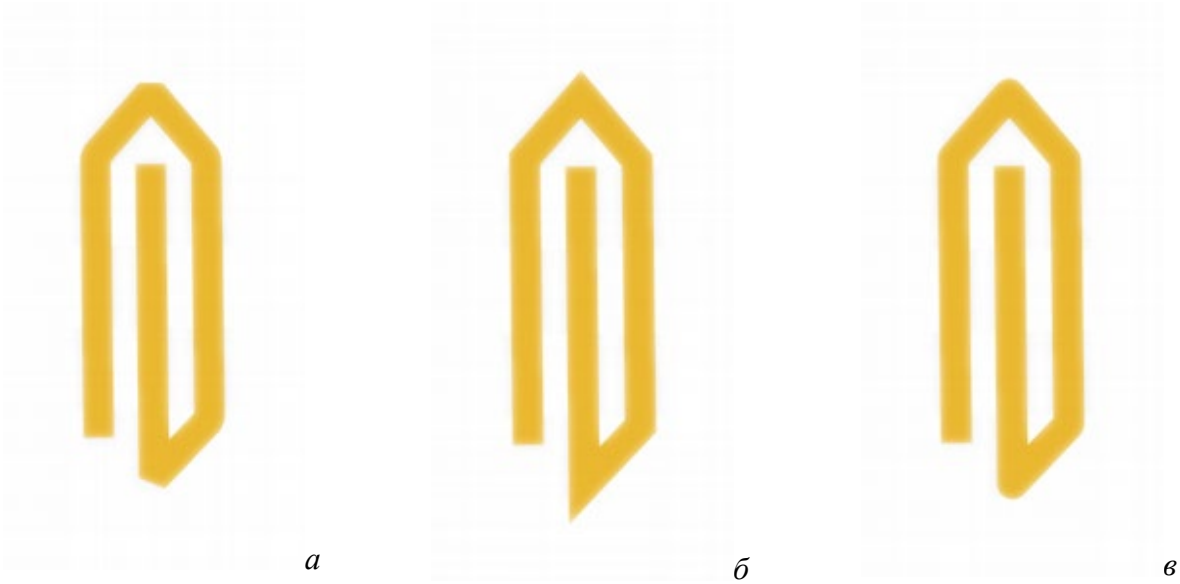
```
SET "y" SET_MATRIX_VALUE y 0 2 3.5
SET "y" SET_MATRIX_VALUE y 0 3 2.5
ECHO TO_STRING x
ECHO TO_STRING y
// Создадим первую линию
ADD_XY_DATA_r x y
//Пример создания линии с использованием команды ADD_XY_DATA_c :
// Создание комплексных входных данных x и y. При построении
//используется первая строка матрицы.
SET "x" CREATE_COMPLEX_MATRIX 1 4
SET "y" CREATE_COMPLEX_MATRIX 1 4
SET "x" SET_MATRIX_VALUE x 0 0 (0.,0.)
SET "x" SET_MATRIX_VALUE x 0 1 (1.,1.)
SET "x" SET_MATRIX_VALUE x 0 2 (3.,3.)
SET "x" SET_MATRIX_VALUE x 0 3 (4.5,4.5)
SET "y" SET_MATRIX_VALUE y 0 0 (0,1.5)
SET "y" SET_MATRIX_VALUE y 0 1 (0.5,0.5)
SET "y" SET_MATRIX_VALUE y 0 2 (2.5,3.5)
SET "y" SET_MATRIX_VALUE y 0 3 (3.5,2.5)
ECHO TO_STRING x
ECHO TO_STRING y
// Третий параметр - способ построения комплексного числа:
//COMPLEX_PLOT_REAL - строить действительную часть
//COMPLEX_PLOT_IMAG - строить мнимую часть
//COMPLEX_PLOT_ABS - строит модуль
ADD_XY_DATA_c x y COMPLEX_PLOT_REAL // Создадим вторую линию.
```

Параметры графика задаются командами, описаными в таблице 5.6.

Таблица 5.6 – Параметры линий

Команда	Описание команды	Описание параметров
SET_PLOT_JOIN_STYLE x	Стиль соединений линии	x принимает значения от 0 до 2: 0 – Bevel соединения (рисунок 5.65а), 1 – Miter соединения (рисунок 5.65б), 2 – Round соединения (рисунок 5.65а)
SET_PLOT_CAP_STYLE x	Стиль окончания линии	x принимает значения от 0 до 2: 0 – квадратное окончание (рисунок 5.66а), 1 – плоское (рисунок 5.66б), 2 – округленное (рисунок 5.66в)
SET_PLOT_STYLE x	Стиль линии	x принимает значения от 0 до 5: 0 – сплошная линия, 5.67а 1 – пунктирная, 5.67б 2 – точечная, 5.67в 3 – точечно-пунктирная 5.67г, 4 – пунктир-точка-точка 5.67д 5 – сплошная пунктирная 5.67е.
SET_PLOT_BACKGROUND r g b	Цвет фона	Параметры r, g и b принимают значения типа double от 0 до 1
SET_PLOT_LINE_WIDTH x	Ширина линии	
SET_PLOT_LOGY x, SET_PLOT_LOGX x	Логарифмическая шкала	x - максимальное значение по шкале (должна быть кратна 10)
SET_PLOT_RANGE xmin ymin xmax ymax	Минимальные и максимсальные значение на шкале	xmin - минимальное значение на шкале X, ymin - минимальное значение на шкале Y, xmax - максимальное значение на шкале X, ymax - максимальное значение на шкале Y
SET_Y_TITLE some_title	Надпись на оси Y	some_title - надпись типа STRING

SET_X_TITLE some_title	Надпись на оси X	some_title - надпись типа STRING
SET_PLOT_TITLE some_title	Надпись над графиком	some_title - надпись типа STRING
SET_PLOT_LABEL some_label	Надпись для линии	some_title - надпись типа STRING
SET_PLOT_COLOR r g b	Цвет линии	Параметры r, g и b принимают значения типа double от 0 до 1



а *б* *в*

Рисунок 5.65 – Результат выполнения команды SET_PLOT_JOIN_STYLE

а *б* *в*

Рисунок 5.66 – Результат выполнения команды SET_PLOT_CAP_STYLE

а *б* *в* *г* *д* *е*

Рисунок 5.67 – Результат выполнения команды SET_PLOT_CAP_STYLE

6 СООБЩЕНИЯ ОПЕРАТОРУ

Текстовые управляющие команды вводятся в командной панели в нижней части главного окна клиента, после чего нажатием ENTER вызывается обработка команды и вывод результатов. На рисунке 6.1 пользователь ввел команду «КОМАНДА» и нажал ENTER.

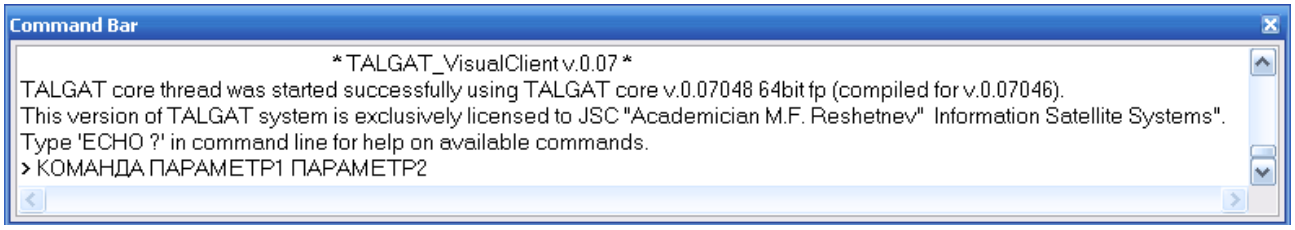


Рисунок 6.1 – Ввод текстовой управляющей команды с двумя параметрами

Клиент передает запрос ядру системы, однако, поскольку данная команда не существует, происходит генерация исключения, о чем клиент сообщает пользователю в специальном окне (рисунок 6.2).

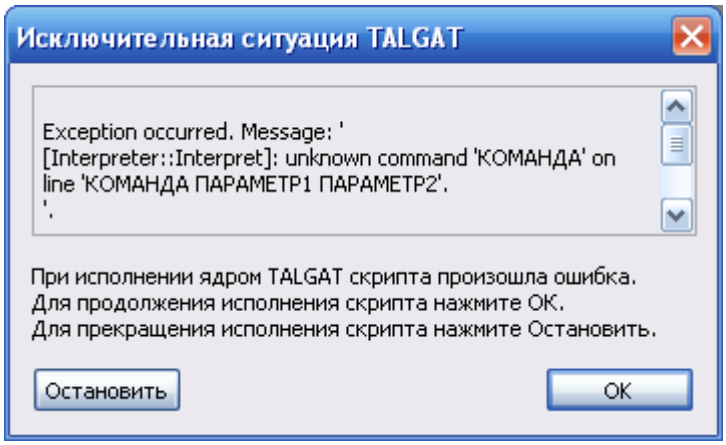


Рисунок 6.2 – Окно с информацией об исключении

Окно исключительной ситуации содержит текстовое описание ошибки; текст строки, которая вызвала ошибку; кнопку ОК для продолжения работы; кнопку «Остановить», которая позволяет предотвратить повторное исполнение команды, если она включена в цикл. Информация об исключении также дублируется в командной панели системы (рисунок 6.3).

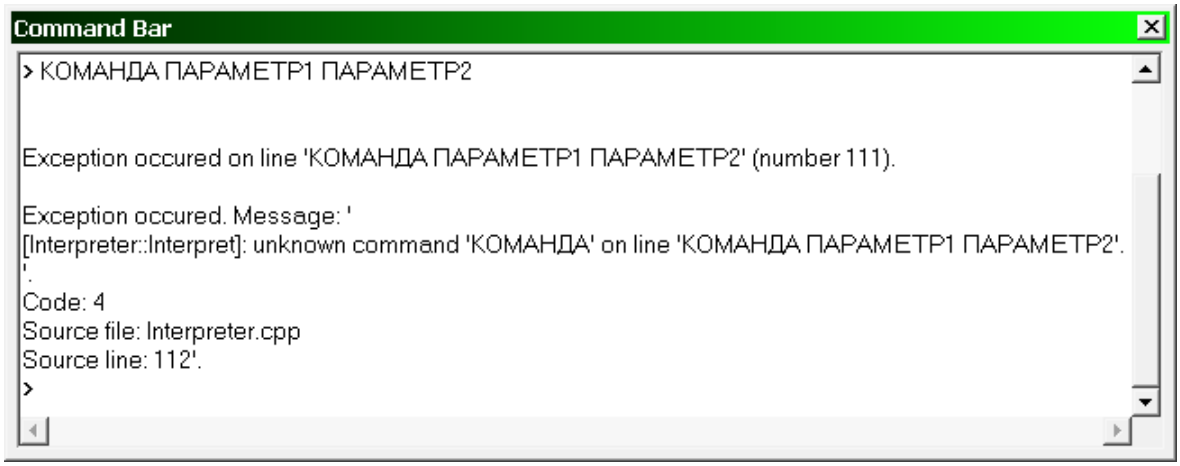


Рисунок 6.3 – Результат исполнения несуществующей команды