ТРУДЫ
ВСЕРОССИЙСКОЙ НАУЧНОЙ КОНФЕРЕНЦИИ
«ПРОЕКТИРОВАНИЕ НАУЧНЫХ И ИНЖЕНЕРНЫХ ПРИЛОЖЕНИЙ В СРЕДЕ MATLAB»

Часть 6. MATLAB в Интернете и образовании

Под общ. ред. В.Е. Кондрашова

СОДЕРЖАНИЕ

УДК 681.3

МЕТОДЫ И МОДЕЛИ ЗАЩИТЫ И СОПРОВОЖДЕНИЯ ОИС В СЕТИ INTERNET/INTRANET НА ОСНОВЕ СРЕДСТВ MATLAB

Ботуз С.П.
Федеральный институт промышленной собственности, г. Москва
e-mail: bsp_serg@pol.ru

В качестве введения в специализированный курс «Распределенная технология графо-аналитической (ГА) защиты и сопровождения объектов интеллектуальной собственности (ОИС) в сети Internet/Intranet (ГВС)» представляется целесообразным осуществить обзор следующих тем:

1. Основы технологии графо-аналитической защиты и сопровождения ОИС в сети Internet/Intranet на базе инструментальных средств MATLAB;
2. Технология и процедуры формирования прав доступа к ОИС в сети Internet/Intranet;

2.1. Технология формирования библиотеки ГА профилей и системной политики экспертизы ОИС;

3. Стратегии графо-аналитической защиты и сопровождения ОИС в сети Internet/Intranet;
4. Конструктивная геометрия графо-аналитических бинарных полей в инструментальной среде MATLAB;

4.1. Основное описание ГА базиса;

4.2. Вспомогательное описание ГА базиса;

5. Синтез конструктивных элементов графо-аналитических бинарных полей;

5.1. Первая задача синтеза ГА бинарных полей;

5.2. Вторая задача синтеза ГА бинарных полей;

5.3. Частные случаи синтеза ГА бинарного поля;

5.4. Конструктивные определения и некоторые топологические свойства графо-аналитических бинарных полей;

6. Графо-аналитические оценки устойчивости нелинейных динамических процессов;

6.1. Условия визуализации многомерных графо-

аналитических оценок систем ЛПР-ОИС-ГВС на экране монитора.

Более подробно рассмотрим основные объектно-ориентированные модули программного комплекса (ПК) загрузочных модулей (BotLab), предназначенного для обработки и формирования графо-аналитических примитивов ОИС в инструментальной среде MATLAB.

В частности, рассмотрим особенности функционирования следующих автономных объектно-ориентированных модулей программного комплекса BotLab:

Base-5 // Синтез графических объектов;

Scanner-8 //Идентификация процессов сканирования ГА объектов ИС;

Rastr-3 // Формирование растровых изображений ГА профиля ОИС;

Gap-3 // Синтез ГА профиля ОИС на основе IP с помощью класса InetAddress.

В результате показано, что автономность базовых модулей **Base-5** и **Scanner-8** имеет функциональный характер. Основные объектные составляющие данных модулей описаны в [1].

Основные объектные составляющие данных модулей описаны в [1] (§§6.2 – 6.5). Поэтому в данном сообщении соответствующие составляющие объектно-ориентированного стиля программирования (абстрагирование, инкапсуляция, модульность, иерархия и др.) не повторяются. При всем этом следует отметить, что все абстракции объектного программирования обладают как статическими, так и динамическими свойствами. Например, если рассматривать любой файл как объект, то он требует определенного объема памяти, имеет имя и содержание. Данные свойства являются статическими.

Однако конкретные значения данных свойств изменяются в процессе использования файла (объекта).

В этой связи для придания модулям автономности осуществлены сопоставительные исследования различных языковых интерпретаций разработанных в [1] алгоритмов (см. §§6.2 – 6.5 в [1]), а именно, рассматривались Java, C++ и Fortran реализации в интегрированной среде MATLAB.

В результате установлено, что наилучшими характеристиками по быстродействию в процессе обработки и формирования графо-аналитических примитивов ОИС обладают загрузочные модули консервативной версии Fortran, откомпилированные в оптимизационном компиляторе Microsoft Fortran Power Station - 4, который основан на стандарте Фортрана-90.

Здесь следует отметить, что в настоящее время стандарт

Фортрана-90 реализован практически для всех компьютеров от персональных ЭВМ до больших вычислительных систем.

На основе данного стандарта разработан язык HPF (High Performance Fortran), который предназначен для оптимизации параллельных вычислительных процессов для многопроцессорных вычислительных комплексов. Так, полученные загрузочные модули на ПЭВМ Pentium-III 800/128MB P-133/QFB 10.2GB/AGP 32MB в оптимизационном компиляторе Microsoft Fortran Power Station – 4 из приводимых в докладе примеров исходных текстов синтеза ГА объектов по критерию быстродействия обработки ГА примитивов информационных полей ОИС, состоящих из 100 ГА примитивов в сравнении с Java-реализацией (Java 1.2), составляет не менее 6-10 раз, а в сравнении с C++ (Borland C++ 5) – в 2-3 раза.

При увеличении числа обрабатываемых ГА примитивов ОИС выигрыш по отношению к Java-реализации приближается к экспоненте, а в случае C++ – эта зависимость носит возрастающий характер, пропорциональный числу обрабатываемых ГА примитивов.

Кроме этого, как показывает практика организации параллельных вычислений на базе интегрированных средств MATLAB, распределенную (параллельную) обработку данных проще всего организовать на базе использования консервативного подмножества языка Fortran. Это не исключает использования более современных его модификаций, например Digital Fortran или Compaq Visial Fortran 6.1a в инструментальной среде MATLAB.

Объектно-ориентированный модуль Base-5 ПК Botlab предназначен для генерации (кодирования, дешифрации, ... , «распознавания») проблемно-ориентированных графических объектов в заданной предметной области, исходя из выделяемых вычислительных ресурсов в сети Internet/Intranet.

В данной реализации состав и структура программного модуля Base-5 в интегрированной среде MATLAB 6 не требует от пользователя каких либо глубоких априорных знаний, связанных с процессом генерации ГА профиля ОИС. Обеспечено это благодаря тому, что данный процесс осуществляется на основе использования базовых графических примитивов, которые высвечиваются на экране монитора. Пользователю данного модуля достаточно выбрать любой из предлагаемого множества или сформировать свой набор ГА парадигм. Этот набор формируется в автоматизированном режиме на основе использования процедуры стохастической кусочно-линейной аппроксимации.

Каждый этап работы с данным объектно-ориентированным модулем, его состав и длительность, определяются ЛПР путем

запоминания состояния информационного поля (ИП) монитора. Каждому такому состоянию присваивается дата и уникальное имя, а все содержимое данного ИП кодируется в виде множества кодовых строчек.

Каждая кодовая строка содержит:

<Тн><имя ГА примитиваXXн, УнXXк, УкXTк>,

где Тн и Тк – время начала и конца, формирования ГА примитива, координаты которого на экране монитора - <Хн, Ун> и <Хк, Ук>, соответственно начало и конец.

Причем, если в качестве такого графического примитива будет выбрано какое-либо состояние ИП, то для данного текущего ИП формируется полный кодовый массив данных. Такая стратегия формирования ГА примитивов позволяет упростить процесс накопления данных (знаний, информативных признаков и т.п.), так как каждый ГА объект (каждая его кодовая страничка или ГА примитив) содержит всю предысторию его создания не только статики, но и динамики. Соответственно итоговый ОИС − ГА образ ИП есть некоторая сумма всех действий, которые были совершены при его создании.

Структура объектно-ориентированного модуля Base-5 не требует каких либо дополнительных комментариев, кроме тех, которые приведены непосредственно в его листинге.

Однако следует отметить, что заменив размерность массивов ппс, хп, ххп, yn, yyn, xk, xkk, yk, xkk, yk, yyk и перекомпилировав программу, можно учесть (например, снизить) требования к объему ОЗУ. В данной реализации, когда в одном ИП может быть 600 ГА объектов, загрузочный модуль после компиляции в оптимизированном компиляторе Microsoft Fortran Power Station - 4 не превышает 340Кбайт.

С помощью данного модуля сгенерировано в настоящее время более 1000 входных файлов с ГА объектами в таких предметных областях, как системы управления и оптическая обработка данных по следующим рубрикам МПК: G01D9/00, G05B11/16, G011B7/00 и др.

Объектно-ориентированный модуль Scanner-8 ПК Botlab предназначен для параметрической идентификации процессов сканирования ГА объектов, отображаемых на ИП. При этом сам алгоритм сканирования того или иного объекта может быть синтезирован ЛПР непосредственно в процессе кодирования, дешифрации, ..., распознавания» проблемно-ориентированных графических объектов, например, на основе использования ЛПР в качестве источника программных (задающих) воздействий.

Состав и структура объектно-ориентированного модуля

Scanner-8 основан на организации и поддержке активного взаимодействия с ЛПР, идентифицируя его не только как некоторую биомеханическую систему, осуществляющую позиционирование, например, указателя курсора «мыши» в заданном наборе ИП, но и как некоторый когнитивный инструмент или набор стереотипов для исследования когнитивных способностей ЛПР. В последнем из вышеперечисленных режимов функционирования соответствующая обратная интерактивная связь приобретает двойственный характер, а именно, ЛПР выступает как некоторая механическая подсистема, наделенная когнитивными характеристиками. Сам процесс формирования траектории сканирования и последовательность просмотра ИП определяется в динамике или задается ЛПР.

Объектно-ориентированный модуль Scanner-8 (и его более ранние версии) был использован в процессе решения различных задач (научных [1] см. § 3.6, практических [1] см. § 6.2 и в учебных процессах [1] см. § 6.5).

Описанные здесь модули могут эффективно выполнять возложенные на них функции, связанные с обработкой и формированием графо-аналитических примитивов ОИС (изобретений, полезных моделей, промышленных образцов и т.п.) не только интегрированной среде MATLAB 4 и выше, но и в различных операционных средах: Windows 95/98 и др.

Литература

1. *Дейтл Г.* Введение в операционные системы: Т.1. М.: Мир, 1987. 359 с.
2. Секреты создания интрасетей. СПб.: Питер, 1998. 592 с.
3. *Фролов А.В., Фролов Г.В.* Программирование для Windows NT. Ч.2. М.: Диалог – МИФИ, 1997. 271 с.
4. *Медведев В.С., Потемкин В.Г.* Нейронные сети. М.: Диалог – МИФИ, 2002.
5. *Ботуз С.П.* Автоматизация исследования, разработки и патентования позиционных систем программного управления. М.: Наука, Физматлит, 1999.
6. *Ботуз С.П.* Методы и модели экспертизы объектов интеллектуальной собственноти в сети Internet. М.: Наука, Физматлит, (в печати).

УДК 681.3:681.42

СОЗДАНИЕ WEB-СТРАНИЦ С ДОСТУПОМ К MATLAB

Иглин С.П.
Национальный технический университет "ХПИ", г. Харьков, Украина
e-mail: iglin@kpi.kharkov.ua

1. Введение

Для обучения, в частности, дистанционного, значительный интерес представляют интерактивные учебники. Они наряду с текстом и иллюстрациями содержат активные элементы, позволяющие пользователю совершать те или иные действия, активизирующие учебный процесс. Так, при изучении математических или технических дисциплин студенту было бы полезно по мере освоения материала сразу проверить свои знания на практике, решив какую-либо задачу. Такую возможность обеспечивает, например, ППП "Notebook" [1], который предоставляет пользователю механизм связи между MATLAB и документом MS Word.

В Internet-приложениях более удобным представляется создание web-страниц, обладающих такими же свойствами. Созданные в формате HTML, эти страницы будут доступны для просмотра обычными браузерами. В то же время на них должны быть элементы, аналогичные зонам ввода и вывода пакета "Notebook", а также реализован сценарий взаимосвязи с MATLAB. Рассмотрим подробнее, какие элементы должны быть на такой интерактивной странице.

1. На странице должна быть кнопка для запуска MATLAB, или, как альтернативный вариант, страница должна автоматически запускать MATLAB при своей загрузке. Видимо, первый вариант предпочтительнее, так как запуск MATLAB занимает некоторое время, что замедляет работу при путешествии по перекрёстным ссылкам.
2. Должна быть область ввода, куда пользователь будет вводить команды MATLAB для последующего счёта.
3. Должна быть также область вывода, на которую web-страница

будет направлять результаты, полученные из MATLAB.

4. Должна быть кнопка и (или) комбинация горячих клавиш, при нажатии на которую будет реализован механизм связи между web-страницей и MATLAB: данные из области ввода будут переданы в MATLAB, а полученные результаты возвращены в область вывода web-страницы.
5. Нужно реализовать механизм сохранения изменений в областях ввода и вывода.
6. Необходимо предусмотреть возможность восстановления исходного состояния областей ввода и вывода.

Пункты 5 и 6 – не обязательные, но они взаимосвязаны: если есть п. 5, то необходим и 6. Если создаваемая интерактивная книга предназначена для работы на локальном компьютере, то эти пункты необходимы. Если же книга будет установлена на сервере, то, видимо, можно обойтись и без них.

В процессе работы web-страница должна динамически изменять область вывода. Из современных браузеров только Internet Explorer поддерживает возможность динамического изменения отдельных тэгов [2]. К сожалению, ни Netscape Communicator, ни Opera такими свойствами не обладают. Поэтому созданные страницы будут корректно взаимодействовать с MATLAB только при использовании Internet Explorer. В других браузерах они будут выглядеть как обычные web-страницы.

Необходимость использовать именно Internet Explorer облегчает задачу разработчика, так как Internet Explorer поддерживает два языка написания сценариев: VBScript и JavaScript, и разработчик может выбрать любой из них. В этом докладе для примеров используется VBScript.

2. Области ввода и вывода

В теле web-страницы область ввода задаётся как многострочный редактор **<textarea>**, а область вывода – как выделенная область ** **:

```
<textarea rows=1 cols=80 wrap=off name=Inp1
  onClick=SetHeight(Inp1) onKeyPress=SetHeight(Inp1)>
</textarea>
<div class=out id=Out1>
</div>
```

Процедура **SetHeight**, которая вызывается при любой работе с мышкой или клавиатурой, устанавливает высоту области ввода в

соответствии с фактическим числом строк в ней. Вот как она выглядит:

```
sub SetHeight(MyInp) ' устанавливает число строк
  Dim ArrOfStr, CountRows
  ArrOfStr = Split(MyInp.Value, vbCr, -1, 1) ' в массив
  CountRows = Ubound(ArrOfStr)+1 ' нужное число строк
  if CountRows>1 then
    MyInp.rows = CountRows ' установили
  end if
end sub
```

Стиль областей ввода и вывода описан так:

```
textarea {color="green"; background-color="#EEEEEE";
  font-family="courier new cyr"; font-size="11pt"}
div.out {color="blue"; background-color="#EEEEEE";
  font-family="courier new cyr"; font-size="11pt"}
```

По аналогии с ППП "Notebook" текст в области ввода будет вводиться шрифтом Courier New зелёного цвета, а в области вывода – этим же шрифтом, но синего цвета. Цвет областей выбран светло-серый, чтобы они выделялись на общем белом фоне страницы.

Сами области ввода и вывода в теле страницы – пустые: информация в них вставляется при загрузке страницы. Такой механизм удобен для сохранения и восстановления областей (см. далее §5). Процедуры для заполнения областей ввода и вывода используют ActiveX элемент, обеспечивающий доступ к файловой системе компьютера. Вот их шаблон:

```
Dim fso
Set fso=CreateObject("Scripting.FileSystemObject")
sub LoadInp(MyInp) ' загружает область ввода из файла
  Dim MyFile
  Set MyFile = fso.OpenTextFile("имя_файла",1)
  MyInp.Value=MyFile.ReadAll
  MyFile.close
  call SetHeight(MyInp)
end sub
sub LoadOut(MyOut) ' загружает область вывода из файла
  Dim MyFile
  Set MyFile = fso.OpenTextFile("имя_файла",1)
```

```
  MyOut.innerHTML=MyFile.ReadAll
  MyFile.close
end sub
```

По этому шаблону загружаются реальные области ввода и вывода. Вместо символической записи "имя_файла" нужно подставить реальное имя текстового файла, в котором записана соответствующая область.

Если создаваемая страница предназначена для размещения на сервере, и возможность сохранения изменений не предусматривается, то исходное содержимое областей ввода и вывода можно задать в теле страницы, например, так:

```
<textarea rows=1 cols=80 wrap=off name=Inp1
  onClick=SetHeight(Inp1) onKeyPress=SetHeight(Inp1)>
clear all % очистили область
syms x % описали символическую переменную
y=sin(x); % вычислили символическую функцию
fprintf('Функция: y=%s\n',char(y)); % напечатали
x1=subs(x,linspace(0,pi)); % задали массив x
y1=subs(y,x,x1); % посчитали значения функции
plot(x1,y1,'b') % рисуем
set(get(gcf,'CurrentAxes'),...
  'FontName','Times New Roman Cyr','FontSize',12)
title('Пример 2D графика') % заголовок
xlabel('\itx') % метка оси OX
ylabel('\ity') % метка оси OY
da=daspect; % получили текущий масштаб
da(1:2)=min(da(1:2)); % выравняли масштаб
daspect(da); % установили новый масштаб
xlim([0 pi]); % пределы по оси OX
</textarea>
<div class=out id=Out1>
<pre>Функция: y=sin(x)</pre>
<center><img border=1 src="ris1.emf"></img></center><br>
</div>
```

Здесь предполагается, что в файле с именем "ris1.emf" находится рисунок, построенный системой MATLAB. Текстовая часть заключена в тэги **<pre>**, чтобы она выглядела в точности так, как в командном окне MATLAB.

3. Подключение MATLAB к странице

В нужном месте тела страницы (например, вначале) нужно расположить кнопку, нажав на которую, пользователь запустит MATLAB. Для кнопки можно использовать тэги `<input>` или `<button>`. Она может выглядеть, например, так:

```
<button id=LoadMATLAB language=VBScript
 title="Подключить MATLAB" onClick=LoadMATLAB()>
 <img src="matlab.gif"></img>
</button>
```

Предполагается, что в файле с именем `"matlab.gif"` находится картинка, помещаемая на кнопку, скажем, анимация с вращающимся логотипом MATLAB. Процедура `LoadMATLAB`, вызываемая при нажатии на эту кнопку, такая:

```
Dim MATLAB
Set MATLAB=Nothing
sub LoadMATLAB() ' загружает MATLAB
Set MATLAB = CreateObject("MATLAB.Application")
 if MATLAB is Nothing then
   alert "Невозможно подключить MATLAB"
 end if
end sub
```

Если на локальном компьютере пользователя MATLAB не установлен, выдаётся сообщение об этом.

4. Связь между web-страницей и MATLAB

Непосредственно после областей ввода и вывода (см. §2) помещаем кнопку с надписью "Посчитать":

```
<input type=button value="Посчитать"
 language=VBScript onClick=CalcMATLAB(Inp1,Out1)>
```

Процедура **CalcMATLAB** должна обеспечивать взаимосвязь между web-страницей и MATLAB. Это – основная процедура, самая сложная и длинная. Опишем её поэтапно. Вначале (после описаний) мы проверяем, доступен ли вообще MATLAB:

```
sub CalcMATLAB(MyInp,MyOut)
```

```
Dim MyV, MyC, Ub, ib, MyR, MyA, CFig, iFig, MyOF, MyID
if MATLAB is Nothing then
alert "Счёт невозможен, так как MATLAB не подключен."
exit sub
end if
```

Теперь осуществляем текстовый ввод-вывод. Если в тексте нет символов кириллицы (вторая половина таблицы ASCII, коды со 128 по 255), то этот фрагмент выглядит очень просто и привлекательно:

```
MyOut.innerHTML = "" ' очистили область вывода
MyV=MyInp.Value 'взяли все символы из области ввода
MyR=MATLAB.Execute(MyV) 'выполнили все команды ввода
MyOut.innerHTML = "<pre>" & MyR & "</pre>" ' вывод
```

К сожалению, этот фрагмент не работает с кириллицей. Символы кириллицы в нём неправильно передаются и в MATLAB, и из него. Для преодоления этих трудностей мы используем команду MATLAB `eval` и файл дневника `diary`. Чтобы правильно передать в MATLAB команды с символами кириллицы, применим следующий приём. Все символы нужной команды преобразуем в байты (обычные числа), составим из них вектор-строку, а в MATLAB пошлём команду, которая преобразует этот вектор снова в символы, а затем выполнит полученную команду. Обратную задачу – правильно передать символы кириллицы из MATLAB в область вывода, мы решим с помощью дневника. Мы запишем сессию в дневник, а потом загрузим содержимое дневника в область вывода.

Итак, первая команда, которую нужно выполнить – это открытие дневника:

```
MyID = "diary.txt" ' имя файла дневника
MyV = "diary(" & chr(39) & MyID & chr(39) & ")"
Ub = len(MyV) ' число символов в команде открытия
MyC = "eval(char([" ' начинаем формировать строку
for ib=1 to Ub ' преобразовываем символ в байт
   MyC = MyC & Cstr(Asc(Mid(MyV,ib,1))) & " "
next
MyC = MyC & "]));" ' закончили создание команды
MATLAB.Execute(MyC) ' открыли дневник
```

Теперь по этому же принципу осуществляем передачу содержимого области ввода в MATLAB. Текстовые результаты при этом запишутся в дневник.

```
MyV = MyInp.Value ' все символы из области ввода
Ub = len(MyV) ' число символов в области ввода
MyC = "eval(char([" ' начинаем формировать строку
for ib=1 to Ub ' преобразовываем символ в байт
  MyC = MyC & Cstr(Asc(Mid(MyV,ib,1))) & " "
next
MyC = MyC & "]));" ' закончили создание команды
MATLAB.Execute(MyC) ' выполнили команды
MATLAB.Execute("diary off") ' закрыли дневник
```

В дневнике записана вся текстовая информация, которую нужно поместить в область вывода. Копируем её оттуда:

```
Set MyV = fso.GetFile(MyID) ' нашли файл дневника
if (MyV.size>0) then ' есть информация в файле
 Set MyV = fso.OpenTextFile(MyID,1) ' открыли дневник
 MyOut.innerHTML = "<pre>" & MyV.ReadAll & "</pre>"
 MyV.Close ' закрыли файл дневника
else
  MyOut.innerHTML = "<pre></pre><br>" ' пусто
end if
fso.DeleteFile(MyID) ' удалили дневник
```

На этом вывод текстовой информации заканчивается. Переходим к помещению всех построенных графических фигур в область вывода. Вначале проверим, созданы ли вообще какие-нибудь фигуры, и, если да, то сколько их. Для этого используем команду **findobj**.

```
MATLAB.Execute("MyHandlesFromMATLABToHTML=findobj;
")
      ' нашли указатели на графические объекты
      ' оставляем только целые - они соответствую фигурам
      MATLAB.Execute("MyHandlesFromMATLABToHTML=
        MyHandlesFromMATLABToHTML(find((
        MyHandlesFromMATLABToHTML-
        floor(MyHandlesFromMATLABToHTML))==0));")
      ' находим max номер - это количество созданных фигур
```

```
        MyR                                                    =
MATLAB.Execute("MyHandlesFromMATLABToHTML=
          max(MyHandlesFromMATLABToHTML)")
         MyA = split(MyR,"=",-1,1) ' разделяем по знаку равенства
         CFig = CInt(MyA(1)) ' количество графических объектов
         MATLAB.Execute("clear
MyHandlesFromMATLABToHTML")
         ' удалили созданную в MATLABe переменную
```

В процессе работы мы создали в MATLAB новую переменную с очень длинным именем: **MyHandlesFromMATLABToHTML**. Будем предполагать, что переменной с таким именем ранее в рабочей области MATLAB не было.

Теперь мы знаем, сколько фигур построено. Записываем каждую из них в файл и загружаем этой файл на web-страницу, как обычный рисунок:

```
        for iFig=1 to CFig ' цикл по всем фигурам
         ' формируем команду для записи из MATLABa в файл
          MyA = "имя_файла_фигуры.emf" ' имя файла фигуры
          MyV = "MyComFromMATLABToHTML=sprintf(" &
            chr(39) & "print -dmeta " & chr(39) & chr(39)
            & "%s" & chr(39) & chr(39) & " -f%d" & chr(39) &
            "," & chr(39) & MyA & chr(39) & ");"
          Ub = len(MyV) ' число символов
          MyC = "eval(char(["
          for ib=1 to Ub
            MyC = MyC & Cstr(Asc(Mid(MyV,ib,1))) & " "
          next
          MyC = MyC & "]));" ' закончили создание команды
          MATLAB.Execute(MyC) ' выполняем команду

MATLAB.Execute("eval(MyComFromMATLABToHTML)")
          ' строка для добавления в зону вывода
          MyR = "<center><img border=1
            src=""имя_файла_фигуры.emf""></img></center>"
         MyOut.innerHTML= MyOut.innerHTML & MyR & "<br>"
          if iFig<CFig then
            MyOut.innerHTML = MyOut.innerHTML & "<p>"
          end if
          MyC = "delete(" & Cstr(iFig) & ")"
          MATLAB.Execute(MyC) ' Удаляем фигуру
```

```
        next
        MATLAB.Execute("clear
MyComFromMATLABToHTML")
    end sub
```

Для каждой из созданных фигур мы формируем команду записи её в файл, выполняем эту команду средствами MATLAB, затем вставляем полученный рисунок на web-страницу, и после этого удаляем фигуру. Созданную вспомогательную переменную `MyComFromMATLABToHTML` также удаляем. Таким образом, все созданные фигуры появляются в области вывода.

5. Сохранение и восстановление областей ввода и вывода

Если создаваемая интерактивная книга предназначена для работы на локальном компьютере, целесообразно создать механизм сохранения изменений в областях ввода-вывода и восстановления исходного их состояния. Сохранить области ввода и вывода можно так:

```
sub SaveInp(MyInp) ' запоминает область ввода в файле
  Dim MyFile
  Set MyFile = fso.OpenTextFile("имя_файла",2,True)
  MyFile.Write(MyInp.Value)
  MyFile.close
end sub
sub SaveOut(MyOut) ' запоминает область вывода в файле
  Dim MyFile
  Set MyFile = fso.OpenTextFile("имя_файла",2,True)
  MyFile.Write(MyOut.innerHTML)
  MyFile.close
end sub
```

Вызов этих процедур может быть осуществлён внутри процедуры **CalcMATLAB** (см. §4) в самом конце.

Для восстановления исходного состояния областей ввода-вывода нужно иметь файлы с исходными вариантами. Тогда процедура восстановления сводится к перезаписи файлов с исходными вариантами в файлы с текущими, и выводе содержимого текущих файлов на web-страницу. Для восстановления поместим на страницу кнопку:

```
<input type=button value="Посчитать"
```

```
 language=VBScript onClick=Restore(Inp1,Out1)>
```

Процедура обработки нажатия на эту кнопку:

```
sub Restore(MyInp,MyOut)
  Dim source, dest
  source = "исходный_файл_области_ввода"
  dest = "текущий_файл_области_ввода"
  fso.CopyFile source, dest, True
  call LoadInp(MyInp)
  source = "исходный_файл_области_вывода"
  dest = "текущий_файл_области_вывода"
  fso.CopyFile source, dest, True
  if (fso.FileExists("исходные_файлы_фигур.emf")) then
    source = "маска_исходных_файлов фигур.emf"
    dest = "каталог_для_фигур"
    fso.CopyFile source, dest, True
  end if
  call LoadOut(MyOut)
end sub
```

Эта процедура переписывает области ввода и вывода из исходных файлов в текущие, и при необходимости переписывает исходные фигуры. Затем области ввода и вывода на web-странице обновляются. Процедуры **LoadInp** и **LoadOut** описаны в §2.

6. Выводы и замечания

В докладе рассмотрены приёмы создания web-страниц с доступом к MATLAB, которые обладают свойствами М-книг. Пользователь, загрузив такую страницу, получает удобный интерфейс работы с MATLAB. Он может ввести свои команды в область ввода, послать их на счёт в MATLAB, и просмотреть полученные результаты в области вывода web-страницы. Рассмотрена также возможность сохранения изменений и восстановления исходного состояния областей ввода и вывода.

По описанной методике автором созданы два методических пособия: "Вариационное исчисление" и "Обработка массива данных". Их можно свободно скопировать с персональной страницы автора в Internet:

http://users.kpi.kharkov.ua/apm/pers/iglin.html.

Созданные с помощью этих методов web-страницы будут корректно взаимодействовать с MATLAB при их загрузке только в

Internet Explorer, так как другие браузеры не поддерживают в должной мере динамические эффекты. Желательно также использовать MATLAB 5.x, так как в 6-й версии реализован другой механизм работы с дневником, и из-за этого возникают проблемы с передачей символов кириллицы. Некоторые варианты MATLAB 6 вообще не воспринимают букву "я" (ASCII-код 255). Возможно, этот байт использован разработчиками системы MATLAB для каких-то внутренних целей.

Литература

1. *Потёмкин В.Г.* Система инженерных и научных расчётов MATLAB 5.x. В 2-х томах. М.: Диалог-МИФИ, 1999. 366+304 с.
2. *Хольцнер С.* Dynamic HTML: руководство разработчика. Киев: Издательская группа BHV, 2000. 400 с.

УДК 53.072

РАЗРАБОТКА УЧЕБНЫХ МОДЕЛИРУЮЩИХ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ MATLAB WEB SERVER

Котельников И.А.[1], Матвеенко А.Н.[2], Черкасский В.С.[2]
[1]*Институт ядерной физики им. Будкера,* [2]*Новосибирский государственный университет, г. Новосибирск*
e-mail: kotelnikov@phys.nsu.ru, cherk@phys.nsu.ru

Учебные программы, моделирующие физические явления, которые позволяют представить результаты расчетов в виде графических и анимационных изображений, являются действенным инструментом обучения в арсенале преподавателей естественно-научных дисциплин. Не подменяя живое общение преподавателя и студента, они дополняют традиционное изложение предмета в виде набора простых моделей, которые можно объяснить “на пальцах”, средствами визуализации теоретических построений. Особую ценность представляют программы двойного назначения, которые можно использовать как для обучения студентов на практических занятиях и лекционных демонстрациях, так и для самостоятельных занятий обучаемых через интернет при предварительном ознакомлении с материалом или для закрепления полученных знаний.

Выбор языка программирования

Для разработки учебных программ мы использовали язык программирования MATLAB, который является мощным интерпретирующим языком сверхвысокого уровня. Использование подобного языка-интерпретатора позволяет достаточно быстро разрабатывать и отлаживать приложения, а благодаря наличию средств создания графического интерфейса создавать удобные в использовании продукты без особых усилий по собственно программированию. MATLAB WEB Server позволяет использовать разработанные для локальной работы программы с минимальными изменениями для предоставления доступа к ним через Интернет.

Закрытые (откомпилированные) и открытые моделирующие программы

При разработке учебных и/или демонстрационных программ первый вопрос, который приходится решать – разрабатываемая программа должна быть открытой или закрытой. Закрытой программой мы называем такую программу, при использовании которой возможно задание различных параметров, определяющих моделируемый процесс, а сам алгоритм производимых вычислений и программа являются неизменными и, как правило, недоступными. Открытой же программой мы называем такую программу, которую может видоизменять пользователь для получения тех или иных ответов. Если говорить об обучении предметной области (физике, математике, химии), то предпочтительным является тщательно разработанная закрытая программа с удобным графическим интерфейсом, использование которой не требует знания языка программирования. Достаточно научиться пользоваться такой программой. Такая же программа является предпочтительной при проведении лекционных демонстраций. Если же целью обучения является обучение моделированию, то предпочтительной является открытая программа, текст которой доступен пользователю и он может сам видоизменять ее, решая новые задачи.

Поскольку целью нашей работы была отработка технологии создания моделирующих (обучающих) программ и/или демонстрационных программ по физике (в данном конкретном случае – по электродинамике), то нами был выбран вариант разработки закрытой программы, текст которой недоступен пользователю.

Исполнение программ на стороне клиента и стороне сервера

Второй вопрос состоит в том, где проводить собственно расчеты – на стороне клиента или на стороне сервера. Расчеты на стороне клиента сужают спектр возможных применений, ограничивая платформы и требуя наличия у клиента соответствующей системы (MATLAB, MathCad, Mathematica) или передачи клиенту соответствующего exe-файла, что является, вообще говоря, нарушением идеологии Интернет особенно в связи с развитием вирусов, и воспринимается большинством пользователей неоднозначно. В результате учета этих обстоятельств нами было принято решение вычисления проводить на стороне сервера, а клиенту передавать результаты в виде статических или анимированных изображений.

Совмещение HTML и MATLAB

Система MATLAB версии 6.1 (или 5.3) работала на сервере под управлением операционной системы Windows 2000 (или NT), на которой был установлен web-сервер IIS (или Apache). Использовалась также конфигурация, где Web-сервер и MATLAB Web Server были установлены на разных компьютерах в пределах единой локальной сети.

В отличие от предоставляемых фирмой MathWork образцов, где все задачи вместе со своими html и m-файлами располагаются в одном каталоге, нами была разработана типовая система каталогов с соответствующими правами доступа. Каждая задача размещалась в отдельном каталоге, внутри которого размещался каталог для m-файлов (исполняемых файлов MATLAB) и, возможно, каталоги для хранения временных файлов (графических или анимационных, сгенерированных программой), файлов справки с описанием задачи т.д. Такая структура представляется нам более целесообразной

Для ввода параметров задачи и вывода результатов расчетов были разработаны html-шаблоны, обеспечившие дополнительную интерактивность по сравнению с типовыми шаблонами, поставляемыми в составе программного обеспечения MATLAB Web Server.

Проблемы вывода динамических изображений и пути их решения

Большинство представляющих интерес демонстраций и, соответственно, программ, обеспечивающие получение этих демонстраций, содержат динамические изображения в качестве результата расчета. В настоящее время в системе MATLAB существует два способа получения анимированных изображений - анимация "на лету" с помощью изменения свойств соответствующих графических объектов, и анимация, получаемая с помощью создания соответствующей матрицы с последующим ее выводом с помощью процедуры **movie**. Ни один из этих способов в чистом виде не годится для использования в Интернете. Последовательная пересылка и загрузка последовательности html-страниц для получения анимации, конечно, реализовать невозможно. Пересылка же матрицы для функции movie и ее последующее воспроизведение на стороне клиента требует наличия у клиента установленной системы MATLAB, от чего мы отказались с самого начала. Поэтому нами было принято решение генерировать на стороне сервера средствами MATLAB стандартный

для Интернета анимированный графический файл. Система MATLAB до версии 6.0 таких средств в качестве встроенных функций не содержала. Поэтому после проведенного поиска мы остановились на программе mpgwrite, взятой нами из архива MathWorks, и пакета программ makemovie, разработанного A.Weigman (wiegmann@math.lbl.gov).

Программа **mpgwrite** использует матрицу, подготовленную с помощью стандартных средств MATLAB для использования с функцией **movie.** Но дело в том, что функция **getframe**, используемая для этих целей, не работает на стороне сервера и команда **M(j)=getframe** возвращает пустую матрицу. Пришлось создавать матрицу **M** с помощью функции **print.**

```
print -djpeg -r72 skin.jpg
OutImage = imread('skin.jpg');%
frame(j1).cdata=OutImage;
frame(j1).colormap=[];
```

После чего использовать матрицу **frame** в качестве аргумента функции **mpgwrite** для создания **mpg-**файла**.**

При использовании функции **makemovie** в пакете поставки имеется функция **makeframe**, которая создает и записывает на диск промежуточные файлы с мгновенными изображениями. В этом пакете имеются варианты создания **gif-**файлов и **mpg-**файлов, но вариант с созданием **mpg-**файлов не работает, поэтому нами в настоящее время используются оба пакета один - для создания **gif-**файлов и второй - **mpg-**файлов. Необходимость предоставлять пользователю возможность выбора типа генерируемого файла связана с тем, что просмотр этих файлов осуществляется разными средствами по разному. **Gif-**файл просматривается на экране средствами броузера и полученная анимация прокручивается заранее заданное число раз. **Mpg-**файл просматривается с помощью проигрывателя Windows Media **(**на компьютерах, на которых установлена операционная система Windows) и этим просмотром можно управлять (останавливать, повторять, просматривать по кадрам)

Опыт разработки и эксплуатации моделирующих программ в Интернете

По описанной технологии к настоящему времени разработаны 6 задач по электродинамике: “Диаграмма направленности антенн”, “Токи на поверхности кубического резонатора”, “Диаграмма

направленности излучения релятивистской частицы", "Движение релятивистской частицы в поле сильной электромагнитной волны", "Нестационарный скин-эффект" и "Скин-эффект внутри проводящего полого цилиндра". С этими задачами можно познакомиться на сайтах http://matlab.tutornet.ru, http://matlab.fija.nsu.ru, http://phys.nsu.ru:8000.

При эксплуатации задач выяснилась следующая проблема - аварийный останов программы (деление на ноль, несовпадение размерности массивов и т.д.), т.е. те ошибки, которые при локальном исполнении приводят к останову исполнения задачи и переходу в режим командного окна, приводит к зависанию программы **matweb.exe** и необходимости его перезапуска. Кроме того, использованная система генерации графических файлов с записью на диск промежуточных картинок является достаточно сложной и относительно медленной. Выходом может быть разработка функции, генерирующей графические анимированные файлы в памяти. Возможно, появившаяся в версии 6.0 функция **avifile** поможет решить эту проблему.

В настоящее время все разработанные задачи могут работать также в локальном варианте, если на компьютера пользователя установлен MATLAB. При работе в пределах Новосибирского научного центра время получения результата расчета составляет для разных задач от 30 секунд до 2 минут.

УДК 004

ОБ ОСВОЕНИИ ПАКЕТА MATLAB С ИСПОЛЬЗОВАНИЕМ ПРАКТИЧЕСКИХ ЗАДАЧ

Кручинин П.А.
Московский государственный университет им. М.В.Ломоносова, г. Москва
e-mail: kruch@mech.math.msu.su

На механико-математическом факультете МГУ для студентов-механиков младших курсов преподавателями кафедры прикладной механики и управления читается факультативный спецкурс «Компьютерный анализ механических систем». Целями этого курса являются – ознакомление студентов с задачами, решаемыми на кафедре и привлечение их интереса к интегрированным пакетам, используемым для научных исследований.

Одним из таких пакетов является MATLAB. Для знакомства с этим пакетом отведен семестровый курс лекционно-практических занятий с зачетом. На этих занятиях студенты знакомятся основами MATLAB'а, решая практические задачи на компьютере. Особенность чтения подобного курса на мех-мате МГУ заключается в том, что слушатели хорошо освоили основы фундаментальных математических дисциплин, таких как математический анализ, линейная алгебра, аналитическая геометрия и др., неплохо представляют основы программирования. Однако знакомство с основными курсами прикладных дисциплин, таких как теория дифференциальных уравнений, методы вычислений, теория вероятности, теоретическая механика и др. для них только начинается. Это обстоятельство предъявляет дополнительные требования к простоте изложения содержательных механических задач.

Первые три лекции курса посвящены знакомству студентов с основными командами и функциями пакета MATLAB. Дальнейшие занятия проводятся по следующему сценарию. Формулируются новые для слушателей функции MATLAB'а. Излагается содержание несложной механической задачи и алгоритм ее решения. Слушателям предлагается написать программу решающую эту задачу. Перечислим предлагаемые задачи и разделы MATLAB, используемые при их решении.

1. Технологии ядерного сглаживания дискретной последовательности случайных чисел на примере показаний гравиметра. Эта задача используется для закрепления пройденного ранее материала: написание функций и циклов, использование поэлементных векторных операций, функций **eval** и **feval**.

Важной частью алгоритмов авиационной гравиметрии [1] является решение задачи сглаживания показаний гравиметрической аппаратуры. В связи с наличием ошибок дискретизации температурных помех и прочих возмущений соотношение сигнал-шум в этой задаче достигает значений порядка 10^3. Характерный вид показаний гравиметра приведен на рис. 1.

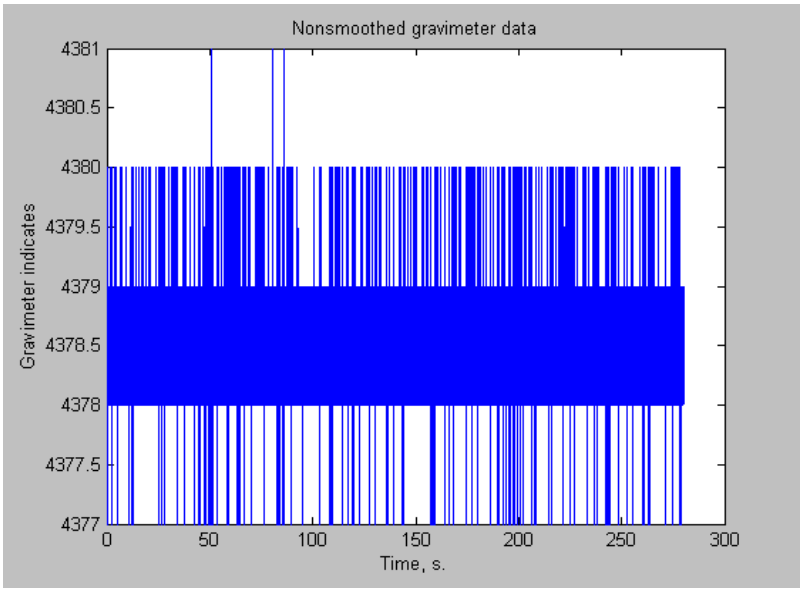


Рис. 1. Показания гравиметра.

Одна из эффективных процедур фильтрации высокочастотных шумов в таком сигнале связана с использованием процедур ядерного сглаживания [2]. В соответствии с этим методом новое сглаженное значение показаний для момента времени t вычисляется как среднее арифметическое $2N$ взвешенных значений взятых на интервале времени $[t-N*\Delta t,\ t-N*\Delta t]$ (Δt – величина такта съёма измерений). Выбор весовых коэффициентов можно сделать с использованием нескольких функций из TOOLBOX'а *ident* пакета MATLAB. Программа осуществляющая эти вычисления может иметь вид:

```
function [s] = winsms(h,nwin,winname)
% Window smoothing
%
% [s]=winsms(h,nwin,winname)
```

```
%
% h – матрица данных,
%       подлежащих сглаживанию
% nwin – ширина окна.
% winname – имя функции расчета
%             весовых коэффициентов
%
% s – сглаженные данные

if nargin < 3,
   winname='hanning';
end

wh=feval(winname,nwin);
wh=wh/sum(wh);
[N,k]=size(h);
s=zeros(N-nwin+1,k);
for j=1:k,
   for i=1:(N-nwin+1),
     s(i,j)= sum(h(i:(i+nwin-1),j).*wh);
   end
end
```

Результат вычислений для окон Хэмминга (функция *hamming*) и фон Ханна (функция *hanning*) приведены на рис.2 и 3.

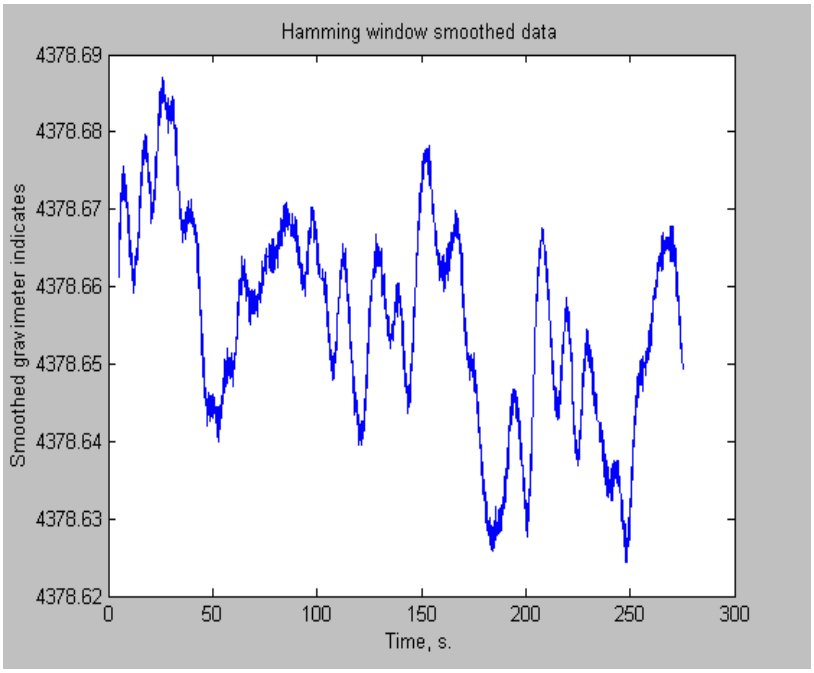


Рис. 2. Результат сглаживания показаний гравиметра окном Хэмминга.

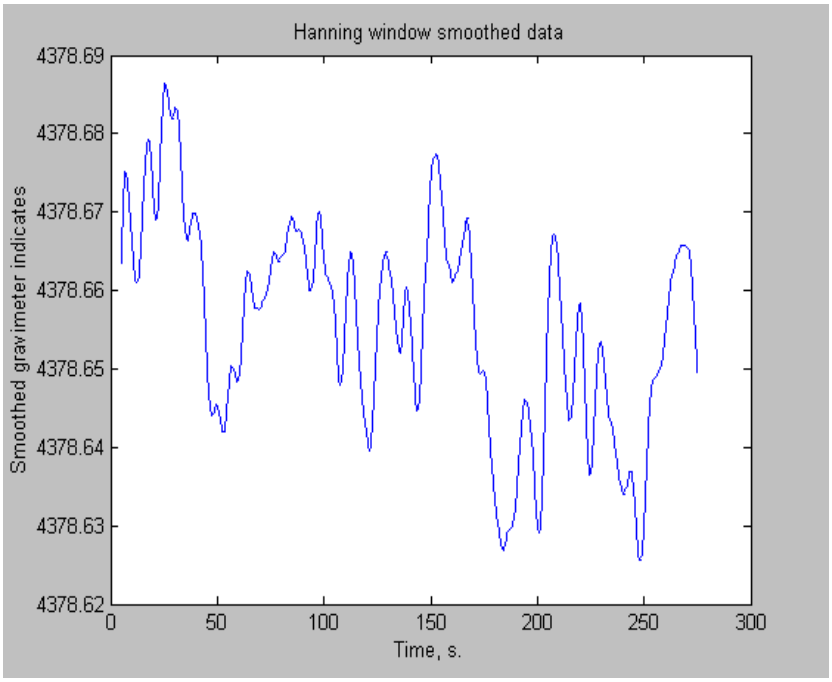


Рис. 3. Результат сглаживания показаний гравиметра окном фон Хана.

2. Построение рабочей области двузвенного манипулятора.

Эта задача используется при освоении функций двумерной и трехмерной графики.

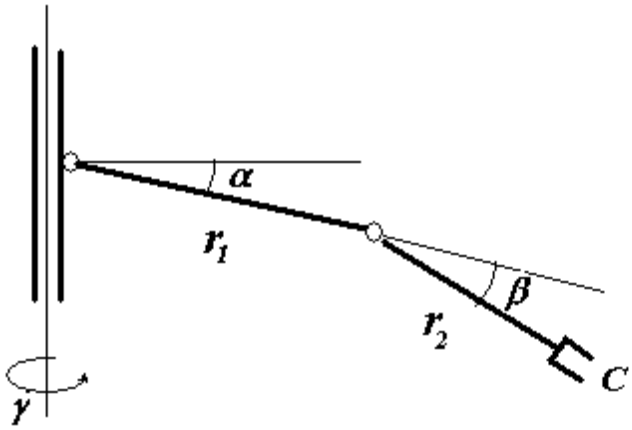


Рис. 4. Схема двузвенного манипулятора.

Для двузвенного манипулятора, изображенного на рис. 4, предлагается построить область точек пространства до которых

может дотягиваться захват *C* при заданных значениях длин звеньев и ограничениях на углы поворота в сочленениях $-\alpha_{max} \le \alpha \le \alpha_{max}$, $-\beta_{max} \le \beta \le \beta_{max}$, $-\gamma_{max} \le \gamma \le \gamma_{max}$. Задача такого типа является традиционной при построении и исследовании манипуляторов [3] и шагающих аппаратов [4].

Задачу предлагается решать в два этапа: на первом этапе построить непрерывные векторы *X* и *Y,* содержащие координаты границы рабочей области плоского манипулятора при $\gamma=0$, а на втором использовать эту границу для построения пространственной поверхности. Алгоритм построения границы рабочей области сводится к кусочному построению границы: сначала рисуется часть границы для которой α меняется на интервале $[-\alpha_{max,}\alpha_{max},]$, а $\beta=0.$ далее $\alpha=\alpha_{max},$, $\beta \in [0, \beta_{max}]$; $\alpha \in [-\alpha_{max,}\alpha_{max},]$, а $\beta=\beta_{max}$ и так далее до замыкания кривой. Функция решающая эту задачу имеет вид:

```
function  [X,Y]=achie(r1, r2,  Amax, Bmax)

%  r1,r2 – длины суставов
%   Amax – максимальное значения угла alpha
%   Bmax –  максимальное значения угла beta
%
%  X,Y – координаты границы рабочей области

n=100;
Ap=linspace(0,Amax,n);
Bp=linspace(0,Bmax,n);
Am=fliplr(Ap);
Bm=fliplr(Bp);

X=r1*cos([-Am,Ap])+r2*cos([-Am,Ap]+Bmax);
Y=r1*sin([-Am,Ap])+r2*sin([-Am,Ap]+Bmax);
X(2*n+1:3*n)=r1*cos(Amax)+r2*cos(Amax+Bm);
Y(2*n+1:3*n)=r1*sin(Amax)+r2*sin(Amax+Bm);
X(3*n+1:4*n)=(r1+r2)*cos(Am);
Y(3*n+1:4*n)=(r1+r2)*sin(Am);
X(4*n+1:5*n)=(r1+r2)*cos(-Ap);
Y(4*n+1:5*n)=(r1+r2)*sin(-Ap);
X(5*n+1:6*n)=r1*cos(-Amax)+r2*cos(-Amax-Bp);
Y(5*n+1:6*n)=r1*sin(-Amax)+r2*sin(-Amax-Bp);
X(6*n+1:8*n)=r1*cos([-Am,Ap])+r2*cos([-Am,Ap]-Bmax);
Y(6*n+1:8*n)=r1*sin([-Am,Ap])+r2*sin([-Am,Ap]-Bmax);

if r1*sin(Amax)+r2*sin(Amax-Bmax) < …
```

```
                    r1*sin(-Amax)+r2*sin(-Amax+Bmax),
    Bk=Amax+asin(r1/r2*sin(Amax));
     Bp=linspace(Bk,Bmax,n);
    Bm=fliplr(Bp);
     X(8*n+1:9*n)=r1*cos(Amax)+r2*cos(Amax-Bm);
    Y(8*n+1:9*n)=r1*sin(Amax)+r2*sin(Amax-Bm);
    X(9*n+1:10*n)=r1*cos(-Amax)+r2*cos(-Amax+Bp);
     Y(9*n+1:10*n)=r1*sin(-Amax)+r2*sin(-Amax+Bp);
end
plot(X,Y)
h=gca;
xmin=min(X)-0.1*abs(min(X));
ymax=max(Y)*1.1;
set(h,'Xlim', [xmin, (r1+r2)*1.1],'Ylim',[-ymax,ymax]);
```

Пространственную граничную поверхность рабочей области манипулятора можно представить в результате выполнения следующих операторов

```
G=linspace(-Gmax,Gmax,100);
Xp=X'*cos(G);
Yp=Y'*ones(size(G));
Zp=X'*sin(G);
mesh(Xp,Yp,Zp)
view([3,-1,-1])
```

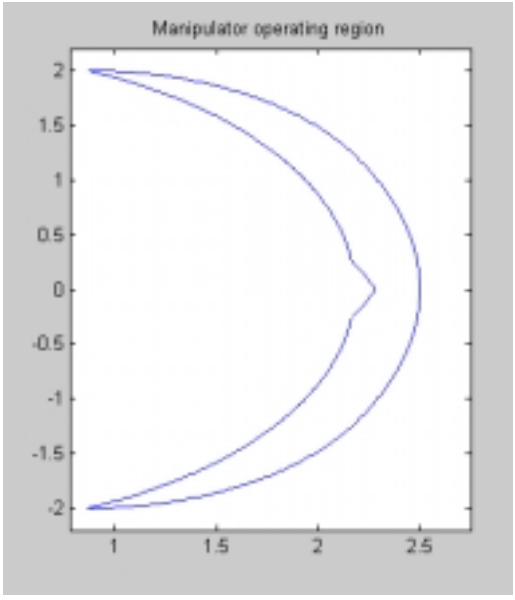


Рис. 5. Рабочая область плоского манипулятора при $r_1=1$, $r_2=1.5$, $\alpha_{max}=\pi/6, \beta_{max}=\pi/3$.

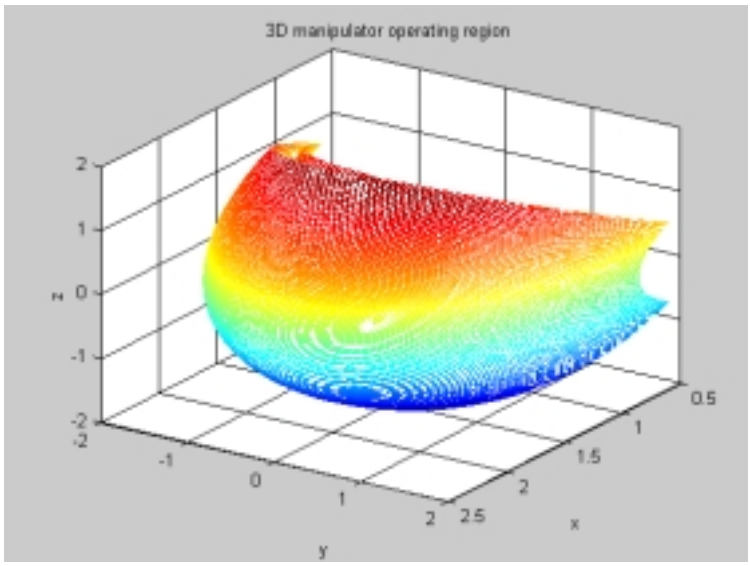


Рис. 6. Рабочая область пространственного манипулятора при $\gamma_{max}=\pi/4$.

3. Моделирование плоского движения тяжелой материальной точки, брошенной в круглую трубу (случай абсолютно упругого удара).

Решение этой задачи позволяет закрепить пройденный материал по работе с визуализацией результатов и используется для освоения процедуры отыскания корней полинома и использованию простейшего меню. Одновременно слушатели знакомятся с классом детерминированных динамических систем, чье поведение может носить хаотический характер.

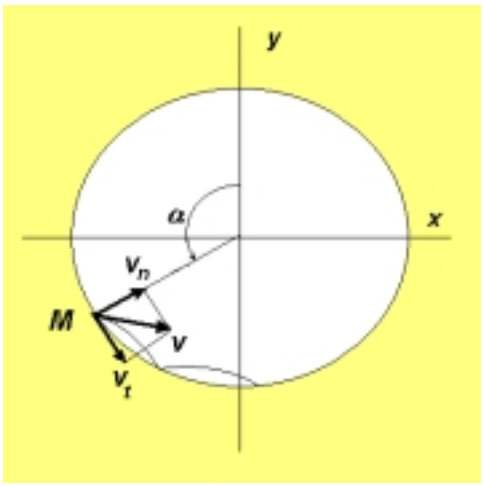


Рис. 7. Описание параметров.

Слушателям предлагается рассмотреть плоскую задачу о движении шарика в виде материальной точки, брошенной в

горизонтально расположенную круглую трубу в поле вертикальной силы тяжести [5]. Для упрощения задачи будем считать массу точки и радиус трубы единичными, а удельную силу тяжести полагаем равной *a=2*.

После отскока шарика от стенки трубы его положение опишем углом α, образованным вертикалью и радиусом трубы, проведенным в точку столкновения. Для удобства задания начальных условий начальную скорость шарика опишем в виде суммы двух составляющих: v_t - касательной к поверхности трубы и v_n – нормальной к этой поверхности. Радиус вектор и скорость точки *M*, записанные в проекциях на оси неподвижной декартовой системы координат имеют вид

$$\vec{r}_+ = -\begin{bmatrix}\cos\alpha\\ \sin\alpha\end{bmatrix}; \qquad \vec{v}_+ = \begin{bmatrix} v_{n+}\sin\alpha - v_{t+}\cos\alpha \\ -v_{n+}\cos\alpha - v_{t+}\sin\alpha \end{bmatrix}$$

Будем значениям скоростей до соударения приписывать индекс ‘-’ и индекс ‘+’ составляющим скорости после соударения. Траекторию движения шарика после соударения запишем в виде

$$\vec{r}(t) = \vec{r}_+ + \vec{v}_+ t - \begin{bmatrix}0\\2\end{bmatrix}\frac{t^2}{2}, \qquad \vec{v}(t) = \vec{v}_+ - \begin{bmatrix}0\\2\end{bmatrix}t.$$

Момент следующего соударения шарика со стенкой трубы найдем из условия $||x||^2=1$. Это условие можно преобразовать к уравнению 3-ей степени

$$t^3 - 2v_{y+}t^2 + (2y_+ + ||\vec{v}||^2)t + 2(x_+v_{x+} + y_+v_{y+}) = 0$$

Здесь x_+ и y_+ – компоненты радиус вектора $\vec{r}(t)$, а v_{x+} и v_{y+} компоненты вектора $\vec{v}$. Наименьший положительный действительный корень этого уравнения определяет момент времени τ следующего соударения, которое происходит в точке $\vec{r}_- = \vec{r}(\tau)$ (α=*-atan2(y_(τ),x_(τ))*) со скоростью $\vec{v}_- = \vec{v}(\tau)$. Условие изменения скорости в результате абсолютно упругого удара запишем в виде

$$\vec{v}_+ = \vec{v}_- - 2(\vec{v}_-, \vec{r}_-)\vec{r}_-$$

Программа, реализующая моделирование в соответствии с представленными вычислениями имеет вид.

```
function [x,y]=billiard(ax,vt,vr,n)

plot(sin(0:(pi/360):(2*pi)),cos(0:(pi/360):(2*pi)),'k')
hold on
acc=[0,-1];
if nargin==3,
```

```
 n=100;
elseif nargin < 3,
 error('Не хватает аргументов функции.')
 return
end

 xp=[cos(ax),-sin(ax)];
 vp=[vr*cos(ax)-vt*sin(ax), -vr*sin(ax)+vt*cos(ax)];

a=zeros(n,1);
a1=zeros(n,1);
x=zeros(100*n,1);
y=zeros(100*n,1);

for i=1:n,
  c=[1,          -4*vp(2),          4*(vp(1)^2+vp(2)^2-xp(2)),
8*(xp(1)*vp(1)+xp(2)*vp(2))];

 r=roots(c);
  r=sort(r);
  tau = 0;
   for j=1:3,
     if imag(r(j)) == 0 & real(r(j)) > 0,
       tau = real(r(j));
       break;
     end
   end
   if tau == 0,
     i
      error(' Polynomial have not     any real positive roots.');
     break;
   end
   t=(tau/100):(tau/100):tau;
   xt=ones(100,1)*xp+t'*vp+(t').^2*acc/2;
   x(((i-1)*100+1):(i*100))=xt(:,1);
   y(((i-1)*100+1):(i*100))=xt(:,2);

   vm=vp+acc*tau;
   xm=xp+vp*tau+acc*tau^2/2;

   if abs(norm(xm)-1) > 1.e-6,
     i
     disp(tau)
```

```
error(' Model point is not on circle.');
      break;
    end

    xp=xm;
    vp=vm-2*(vm*xm')*xm;
end
plot(x,y)
hold off
```

Для вызова этой функции целесообразно создать script-файл с программой

```
Names={'Alpha','vn',’vt’,'Steps'};
Title='Initial data ';
LineNo=4;
DefAns={'pi/6','1',’0.1’,'100'};
while str2num(DefAns{4})>0,
  DefAns=inputdlg(Names,Title,LineNo,DefAns);
   [x,y]=billiard(str2num(DefAns{1}),str2num(DefAns{2}), …
   str2num(DefAns{3}), str2num(DefAns{4}));
end
```

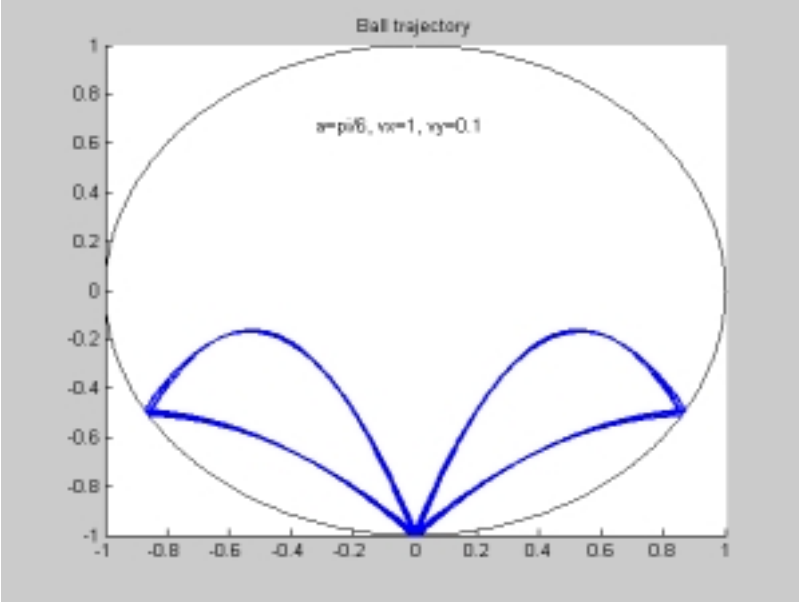


Рис. 8. Траектория движения шарика в трубе при начальных условиях $\alpha=\pi/6$, $v_n=0.1$, $v_t=1$.

На рис. 8 и 9 показаны результаты вычислений этой программы. Начальные условия решения, приведенного на рис. 8

соответствуют малой окрестности устойчивой циклической траектории. На рис. 9 приведена траектория соответствующая хаотическому поведению. При увеличении числа отскоков *n* траектории заполняют весь фазовый объём, соответствующий заданному уровню энергии в системе.

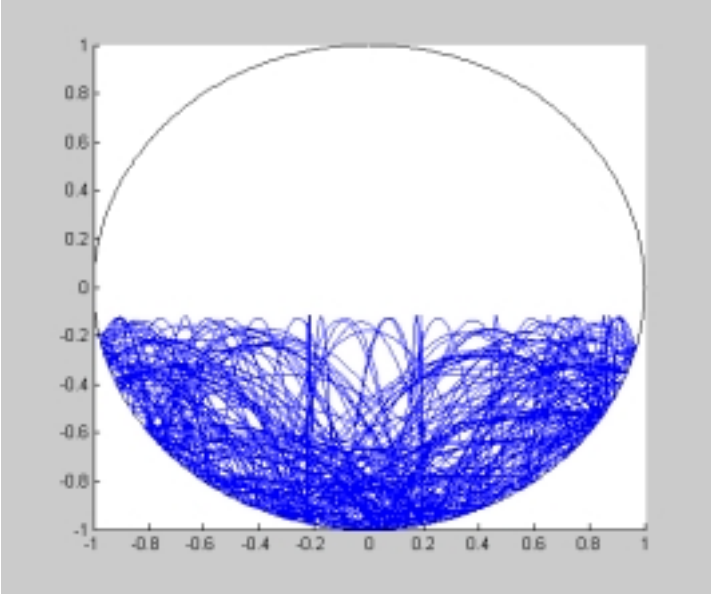


Рис. 9. Траектория движения шарика в трубе при начальных условиях $\alpha=\pi/6$, $v_n=0.15$, $v_t=1$, $n=300$.

4. Плоские модельные задачи спутниковой навигации

Используются при освоении функций решения нелинейных уравнений.

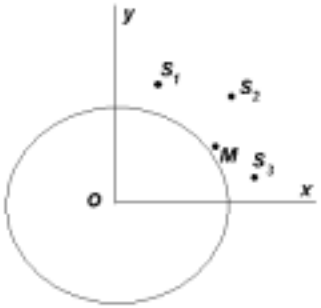


Рис. 10. Система координат в плоской модельной задаче спутниковой навигации.

Задача определения координат приемника в результате обработки сигналов спутниковой навигационной системы является одной из наиболее сложных задач, требующих глубокого знания

специальных математических методов обработки информации [6], [7]. Для решения в рамках предлагаемого спецкурса предлагаются ряд простейших задач, отражающих некоторые особенности спутниковой навигации.

Будем предполагать, что приемник спутниковой навигационной системы принимает одновременно сигналы нескольких спутников. Каждый сигнал, посланный спутником с номером *k,* содержит информацию о радиус-векторе $\vec{r}_k$ (точнее о его декартовых координатах x_k, y_k, z_k) спутника S_k и точном времени t_k отправления этого сигнала. Координаты радиус-вектора $\vec{r}$ приемника M обозначим через x, y, z соответственно, а через t обозначим время получения сигналов приемником.

Для плоской задачи определить координаты приемника можно в результате решения триангуляционной задачи. Для этого необходимо решить систему из двух уравнений

$$\left\|\vec{r} - \vec{r}_1\right\| = c(t - t_1) \qquad \left\|\vec{r} - \vec{r}_2\right\| = c(t - t_2)$$

Здесь c скорость распространения радиоволн, которую считаем постоянной.

Введем обозначение для дальностией $l_1=c(t-t_1)$ и $l_2=c(t-t_2)$ и запишем эту систему в координатном виде

$$(x - x_1)^2 + (y - y_1)^2 = l_1^2; \qquad (x - x_2)^2 + (y - y_2)^2 = l_2^2; \tag{4.1}$$

Использование переменной l_k позволяет отказаться от требования одновременности получения спутниковых сигналов. Выразим из второго уравнения неизвестную y

$$y = y_2 - \sqrt{l_2^2 - (x - x_2)^2} \tag{4.2}$$

Решение первого уравнения относительно x с помощью функции *fzero* используем для отыскания координат приемника

X=fzero('gps1',xo,[],[],l1,l2,r1,r2)

Функция **gps1** имеет вид

```
function f=gps1(x,l1,l2,r1,r2)
%  f=gps1(x,l1,l2,r1,r2)
% Рекомендуемое обращение
% x=fzero('gps1',3,[],[], 2.5, 2,[1,7],[5,5])
%Для приведенных цифр 1соответствует 1000 километров

    y=r2(2)-sqrt(l2^2 - (x – r2(1))^2);
    f=norm(r1-[x,y])-l1;
```

Недостатком этого метода вычислений является

необходимость задания такого начального приближения *x0,* при котором выражение под знаком радикала в формуле (4.2) неотрицательно.

Более эффективным для решения поставленной задачи представляется использование процедуры **fsolve** из *TOOLBOX'a OPTIM.* В этом случае решается система нелинейных уравнений (4.1) относительно переменных *x* и *y*. Эту задачу решает команда

```
r=fsolve('gps2',r0,[],[],l1,l2,r1,r2);
```

Эта команда вызывает функцию

```
function f=gps2(r,l1,l2,r1,r2)
%  f=gps2(r,l1,l2,r1,r2)
% Рекомендуемое обращение
% r=fsolve('gps2',[3,3],[],[], 2.5, 2,[1,7],[5,5])
% Для приведенных цифр 1 соответствует 1000 километров
 f(1)=norm(r-r1)-l1;
 f(2)=norm(r-r2)-l2;
```

Для предлагаемых в комментариях к функциям числовых значений начальных условий и первым и вторым способом отыскивается решение $\vec{r} = [3.1139,\ 5.6653]$. Между тем предлагаемая система уравнений имеет второе, решение $\vec{r}_* = [3.3361,\ 6.1098]$, ложное с точки зрения решения навигационной задачи. Это решение может быть получено, например, при начальном значении *r0=[3.3,7]*. Отбросить это решение как ошибочное позволяет информация о том, что оно соответствует точке, находящейся на высоте ~500км над поверхностью Земли. Для отделения этого значения можно также использовать сигнал, полученный от третьего спутника.

В завершение рассмотрим еще одну задачу этого цикла. Рассмотрим наличие ошибки синхронизации часов в приемнике. Полагаем, что все часы на спутниках синхронны (это достигается систематической коррекцией их), а часы на спутнике имеют систематическую ошибку Δt. В этом случае приемник определяет не истинную дальность до κ-го спутника, а величину именуемую, псевдодальность $l_k^* = l_k + e$. Здесь $e = c\Delta t$. Для получения координат приемника в этом случае также потребуется третий спутник. Систему уравнений, для получения координат составим из трех уравнений

$$\left\|\vec{r} - \vec{r}_1\right\| = l_1^* - e \qquad \left\|\vec{r} - \vec{r}_2\right\| = l_2^* - e \qquad \left\|\vec{r} - \vec{r}_3\right\| = l_3^* - e$$

Рассмотрим две попарные разности этих уравнений

$$\left\|\vec{r} - \vec{r}_2\right\| - \left\|\vec{r} - \vec{r}_1\right\| = l_2^* - l_1^* \qquad \left\|\vec{r} - \vec{r}_3\right\| - \left\|\vec{r} - \vec{r}_1\right\| = l_3^* - l_1^*$$

В эти уравнения уже не входят слагаемые, связанные с ошибками синхронизации часов. Таким образом, координаты приемника можно определять в результате вызова функции **fsolve**

```
r=fsolve('gps3',r0,[],[],l1,l2,l3,r1,r2,r3);
```

Эта команда вызывает функцию

```
function f=gps3(r,l1,l2,l3,r1,r2,r3)
%  f=gps3(r,l1,l2,l3,r1,r2,r3)
% Рекомендуемое обращение
%     r=fsolve('gps2',[3,3],[],[], 2.5, 2,3.92,[1,7],[5,5],[6,3])
% Для приведенных цифр 1 соответствует 1000 километров
f(1)=norm(r2-r)-l2-norm(r1-r)+l1;
 f(2)=norm(r3-r)-l3-norm(r1-r)+l1;
```

Приведенные задачи легко переформулируются для трехмерного случая. Такое задание можно дать слушателям для самостоятельной работы.

5. Моделирование процессов вставания и приседания антропоморфного многозвенника

Используются при освоении различных видов визуализации исследований.

Слушателям предлагается написать функцию, реализующую мультипликационное изображение вставания и приседания человека. Численные значения координат различных точек на теле человека сведены предварительно в виде таблицы в файл, каждая строка которого соответствует фиксированному моменту времени. Интервал времени, прошедший между записью двух последовательных строк постоянен. Заданы номера столбцов, в которых помещены координаты точек на плече, корпусе, в районе тазобедренного сустава, на колене, пятке и на пальцах стопы. Традиционно изображение проектируется на саггитальную плоскость (в профиль). Можно воспользоваться записями положений этих точек, полученными в результате эксперимента [8], либо численного моделирования типа [9]. В последнем случае мультипликация может оказаться полезной при оценке медиками правдоподобности математической модели.

Мультипликация может быть реализована одним из двух

способов: Первый представляет собой канонический способ, использующий функции **movie**. На первом этапе, при таком подходе производится накопление фреймов изображения в массив, а на втором прокрутка мультфильма.

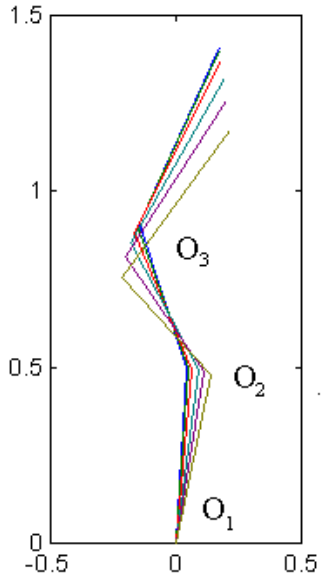


Рис. 11. Последовательность поз, отображаемая при мультипликации.

Сценарий, реализующий представленную последовательность операций представлен ниже

```
% чтение данных из файла
load adc.txt
%Задание столбцов с координатами точек
s=[23, 20, 17, 14, 11, 8];
e=[25, 22, 19, 16, 13, 10];
hold on
h=gca;
%Инициализация массива фреймов
M=moviein(floor(length(adc)/30));
% Накопление фреймов изображения
for k=1:30:length(adc),
  cla
   x=[adc(k,s(1)),adc(k,s(2)),adc(k,s(3)),adc(k,s(4)), …
          adc(k,s(5)),adc(k,s(6))];
   y=[adc(k,s(1)+1),adc(k,s(2)+1),adc(k,s(3)+1), …
          adc(k,s(4)+1),adc(k,s(5)+1),
   adc(k,s(6)+1)];
  plot(x,y)
  set(h,'XLim',[-600,0],'YLim',[0,1500])
```

```
  M(:,floor(k/30)+1) = getframe;
end
size(M)
cla
% Воспроизведение мультипликации
movie(M,1,1);
```

Рис. 11. Вид графического окна моделирования

Простые изображения, получаемые непосредственно в процессе компьютерного моделирования без привлечения функций из языков высокого уровня типа C++, можно получить преррисовывая простое изображение на стандартном сером фоне графического окна пакета *MATLAB*. Пример функции, реализующей мультипликацию такого типа, представлен в виде *GUI,* приведенном на рис. 12. При нажатии в этом окне кнопки “Показать анимацию” вызывается последовательность команд

```
x=adc(10,s)/100;
y=adc(10,s+1)/100;
hp=plot(x,y);
set(gca,'XLim',[-
9,3],'YLim',[0,16],'Visible','off','LineWidth',1);
for i=2:N,
  x=adc(i*10,s)/100;
  y=adc(i*10,s+1)/100;
% Вместо двух предшествующих команд
% могут вызываться функции вычисления координат
% точек в рамках специализированной
%  математической модели
  set(hp,'XData',x,'YData',y);
  pause(0.1)
```

end

6. Моделирование собственных и вынужденных колебаний цепочки упруго соединенных звеньев, совершающих поступательное движение

Используется при освоении матричных операций, вычислении собственных чисел и собственных векторов, решении задачи Коши для однородной и неоднородной линейной системы уравнений.

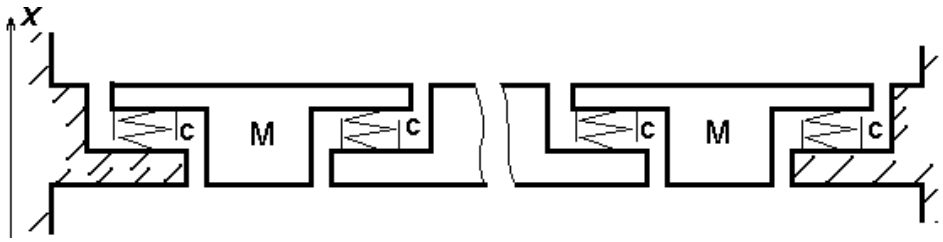


Рис. 12. цепочка упруго соединенных звеньев.

Слушателям предлагается исследовать собственные колебания цепочки звеньев, изображенной на рис. 13. Конструкция расположена в горизонтальной плоскости. Каждое звено этой цепочки имеет массу *M* и может совершать только поступательные движения вдоль оси *x*. Звенья соединены между собой упругими элементами жесткости *c* каждый. Требуется найти собственные частоты и собственные формы колебаний конструкции и промоделировать численно ее собственные колебания для случая, когда начальные условия соответствуют одной из собственных форм. Заметим, что обоснование математической модели такой структуры столь же просто, как и в случае продольно связанных осцилляторов [10], и столь же наглядно, как в традиционной дискретной модели струны (см. например [11]).

Уравнения собственных колебаний цепочки в матричном виде имеют вид

$$\ddot{x} = Cx$$

где $C = \begin{pmatrix} -2\omega_o & \omega_o & 0 & \dots & 0 & 0 \\ \omega_o & -2\omega_o & \omega_o & \dots & 0 & 0 \\ 0 & \omega_o & -2\omega_o & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -2\omega_o & \omega_o \\ 0 & 0 & 0 & \dots & \omega_o & -2\omega_o \end{pmatrix}$, а $\omega_o = c/m$.

Собственные частоты колебаний цепочки отискиваются из мнимых частей собственных чисел матрицы

$$A = \begin{bmatrix} 0 & I \\ C & 0 \end{bmatrix}$$

Здесь I – единичная матрица. Собственные формы, соответствующие указанным собственным частотам, отыскиваются как подвекторы соответствующих собственных векторов.

Текст функции, проводящей соответствующие вычисления для произвольного числа, приводится ниже

```
function [om,sv]=escalat1(m,c,N)
col=['y','b','r','g','m','k','c'];
Cm=zeros(N);
for k=1:N-1,
   Cm(k,k)=-2*c/m;
   Cm(k+1,k)=c/m;
   Cm(k,k+1)=c/m;
end
Cm(N,N)=-2*c/m;
A=[zeros(N),eye(N);Cm, zeros(N)];
[sv,Lam]=eig(A);
l=diag(Lam);
om=imag(l(1:2:2*N));
figure(1)
clf
hold on
for k=1:2:2*N,
   if abs(real(sv(N+1:2*N,k))) <= abs(imag(sv(N+1:2*N,k))),
      plot([0;imag(sv(N+1:2*N,k));0],col(rem((k+1)/2,7)+1))
   else
      plot([0;real(sv(N+1:2*N,k));0],col(rem((k+1)/2,7)+1))
   end
end
hold
offsys=ss(A,ones(length(A),1),eye(size(A)),zeros(length(A),1));
x0=real(sv(:,1));
[x,t]=initial(sys,x0);
figure(2)
clf
plot([0;x0(1:N);0]);
hold on
```

```
for i=1:30,
  plot([0,x(i,1:N),0]);
end
hold off
plot(t,x(:,1:2:end))
```

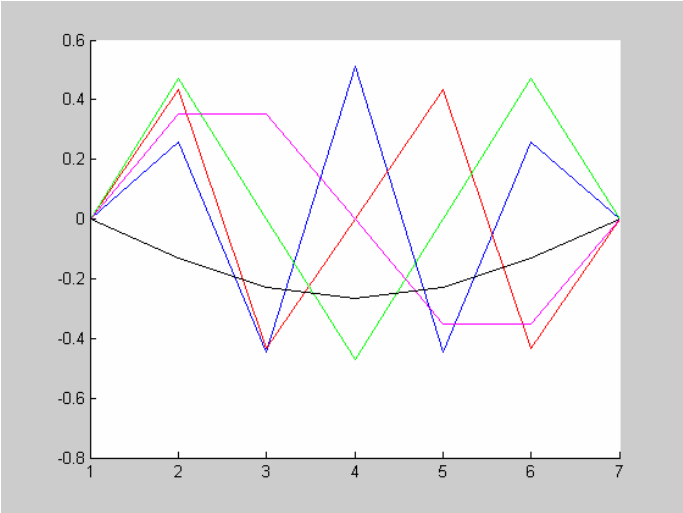


Рис. 13. Собственные формы колебаний цепочки из 5 звеньев.

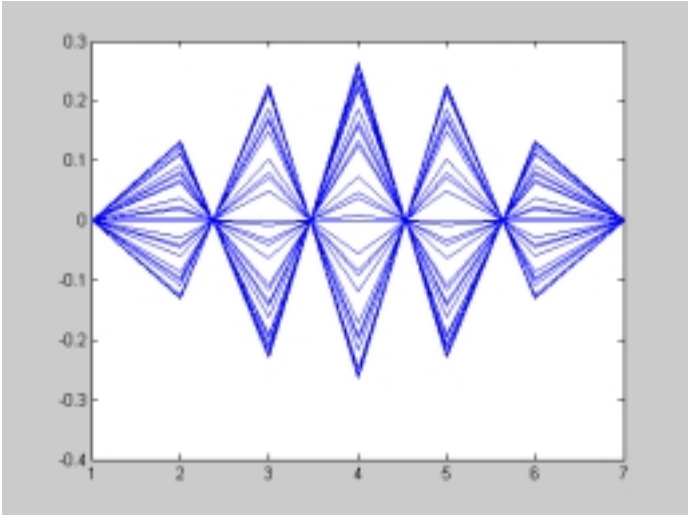


Рис. 14. Положения центров звеньев цепочки при собственных колебаниях одной из форм.

7. Двухмассовая модель вертикальных колебаний автомобиля с активной подвеской

Используется для освоения методов расчета частотных характеристик и знакомства с задачами проектирования алгоритмов управления.

Литература

1. *Болотин Ю.В., Голован А.А., Кручинин П.А., Парусников Н.А., Тихомиров В.В., Трубников С.А.* Задача авиационной гравиметрии. Алгоритмы. Некоторые результаты испытаний //Вестник МГУ. Математика. Механика. 1999. N 2. С. 36-41.
2. *Хардле В.* Прикладная непараметрическая регрессия. М.: Мир, 1993. 349 с.
3. *Черноусько Ф.Л., Болотник Н.Н., Градецкий В.Г.* Манипуляционные роботы. М.: Наука, 1989. 368 с.
4. *Devjanin E.A., Budanov V.M.* Motion Control for the Six-Legged Walking Machine //Proceedings European Mechanics Colloquium Euromech 375. Biology and Technology of Walking. Germany, Munich, 1998, pp. 101-107.
5. *Beletsky V.V., Kasatkin G.V., Starostin E.L.* The Pendulum as e Dynamical Billiard //Chaos, Solitons & Fractals, 1996, v. 7, No. 8, pp. 1145-1178.
6. *Leick A.* GPS satellite surveying. New York Chichester Brisbane Toronto Singapore, Wiley, 1995.
7. *Вавилова Н.Б., Голован А.А., Парусников Н.А., Трубников С.А.* Математические модели и алгоритмы обработки измерений спутниковой навигационной системы GPS. Стандартный режим. М.: Издательство центра прикладных исследований при механико-математическом факультете МГУ, 2001, 116 с.
8. *Mourey F., Grishin A., d'Athis P., Pozzo T., Stapley P.* Standing up from a chair as a dynamic equilibrium task: A comparison between young and elderly subjects //Journal of Gerontology. A. Biol. Sci. Med. Sci. 2000. V. 55. N 9. PP 425-431.
9. *Новожилов И.В., Кручинин П.А., Копылов И.А., Журавлев А.М., Гришин А.А., Демин П.П., Куликовский С.В., Моисеева Е.М.* Математическое моделирование сгибательно-разгибательных движений нижних конечностей при изменении вертикальной позы человека М.: Издательство механико-математического факультета МГУ, 2001. 52 с.
10. *Рабинович М.И., Трубецков Д.И.* Введение в теорию колебаний и волн. М., Ижевск: НИЦ “Регулярная и хаотическая динамика”, 2000, 560 с.
11. *Крауфорд Ф.* Волны. М.: Наука, 1984. (Берклеевский курс физики). 512с.

УДК 517.2

ИЛЛЮСТРИРУЮЩАЯ ПРОГРАММА РЕШЕНИЯ АНАЛИТИЧЕСКИХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ВТОРОГО ПОРЯДКА МЕТОДОМ МАЛОГО ПАРАМЕТРА ПУАНКАРЕ

Кулинич М.В.
Гимназия 38,г. Дзержинск, Нижегородской области
e-mail: kul@hydro.appl.sci-nnov.ru

Многие математические задачи, с которыми сталкиваются исследователи в различных областях науки и техники, не поддаются точному аналитическому решению в общем виде. Среди причин, затрудняющих его нахождение, можно указать, например, такие: нелинейность уравнений, зависимость коэффициентов от времени, нелинейность граничных условий на известных или неизвестных границах сложной формы и т. д. При решении таких задач используются различного рода приближения с комбинацией качественных, аналитических и численных методов. Среди приближенных аналитических методов наиболее распространенным является метод малого параметра Пуанкаре. Метод широко используется в качественной теории дифференциальных уравнений, как для нахождения решений, так и для нахождения периодических решений, описывающих предельные циклы [1]. Развитие компьютерных систем, берущих на себя всю тяжесть промежуточных аналитических вычислений, позволяет получать решение таких задач с высокой степенью точности в виде формул. В работе показываются возможности “toolbox Sumbolic” системы MATLAB 6.0 для построения приближенных аналитических решений уравнения второго порядка

$$x''+x=\mu\cdot f(x,x',\mu) \qquad (1)$$

с условиями

$$x(0)=A, x'(0)=0. \qquad (2)$$

Метод нахождения решения основан на известной теореме разложения Пуанкаре [2]. Сутью этой теоремы для построения приближенного решения является утверждение, что если функция $f(x, x', \mu)$ аналитична по своим переменным, то решение задачи

Коши (1)-(2) есть функция аналитическая и решение можно искать в виде ряда

$$x(t,\mu)=\sum_{k=0}^{\infty}x_k(t)\mu^k. \qquad (3)$$

Подставляя ряд (3) в уравнения (1)-(2) и приравнивая коэффициенты при одинаковых степенях μ, получим цепочку уравнений для нахождения коэффициентов разложения x_k

$$x''_0+x_0=0, \qquad x_0(0)=A, x'_0(0)=0, \qquad (4)$$

$$x_k''+x_k=f_{k-1}(x_0,x_1,...,x_{k-1},x'_k,x'_1,...,x'_{k-1}),$$

$$x_k(0)=A, x'_k(0)=0, k=1,2..., \qquad (5)$$

Решение порождающего уравнения (4) есть

$$x_0(t)=A\cdot\cos(t),$$

а решение уравнения (5) для x_k, находится с помощью формулы Дюамеля

$$x_k(t)=\int_0^t f_{k-1}\cdot\sin(t-\tau)\cdot d\cdot\tau.$$

Заметим, что в найденном таким образом решении всегда будут содержаться секулярные члены. Наличие этих членов, для достаточно больших времен при заданном малом параметре, будет приводить к росту разницы между точным и приближенным решением, которое находится в виде частичной суммы ряда (3).

Алгоритм нахождения приближенного решения в виде ряда (3), исключающий появление секулярных членов, разработан Линдстедом [3]. Этим методом могут находиться предельные циклы, соответствующие периодическим решениям задачи (1)-(2). Реализация базируется на двух тезисах:

1. Масштаб времени *t* деформируется таким образом, чтобы частота результирующего периодического решения зависела от малого параметра μ. Для этого вводится новое время $\tau=\omega t$, где ω - искомая частота периодического решения x(t) исходного уравнения, зависящая от μ. В этом новом масштабе времени период искомых колебаний равен 2π, т. е. в этом масштабе времени частота и период искомых колебаний не зависят от μ.
2. Правые части уравнений для *k*-го приближения используются для уточнения или нахождения коэффициентов предыдущих приближений, основываясь на периодичности этих приближений по τ. Эта же связь приближений используется для удовлетворения начальным условиям или нахождения их.

Выполняя замену переменных в (1)-(2), получим уравнение

$$\omega^2 x''+x=\mu\cdot f(x,\omega\cdot x',\mu) \qquad (6)$$

с условиями

$$x(0) = x_0(\mu), x'(0) = 0, \quad (7)$$

где функция $x_0(\mu)$ в общем случае неизвестна.

Решение задачи (6)-(7) ищется в виде ряда (3). Частоту колебаний ω считают функцией малого параметра μ. Зависимость $\omega(\mu)$, $x_0(\mu)$ находят в виде рядов

$$\omega(\mu) = 1 + \sum_{k=1}^{\infty} \omega_k \mu^k, \quad (8)$$

$$x_0(\mu) = \sum_{k=0}^{\infty} A_k \mu^k. \quad (9)$$

Зависимость для $\omega^2(\mu)$ удобно представить в виде

$$\omega^2(\mu) = 1 + \sum_{k=1}^{\infty} \Omega_k \mu^k. \quad (10)$$

Подставляя (3), (9) и (10) в уравнения (6),(7) , для нахождения коэффициентов разложения x_k, имеем цепочку уравнений

$$x''_0 + x_0 = 0, \qquad x_0(0) = A_0, x'_0(0) = 0, \quad (11)$$

$$x_k'' + x_k = f_{k-1} - \sum_{i=1}^{k} \Omega_i x''_{k-i}(\tau) \; x_k(0) = A_k, x'_k(0) = 0, k = 1,2.... \quad (12)$$

Вместе с условиями периодичности

$$x_k(0) = x_k(2 \cdot \pi), \qquad x'_k(0) = x'_{1k}(2 \cdot \pi) \quad (13)$$

цепочка уравнений (11),(12) будет определять алгоритм нахождения периодического решения задачи (1),2). Из уравнения (11) найдем

$$x_0(\tau) = A_0 \cdot \cos(\tau),$$

с неизвестным пока коэффициентом A_0. Подставляя найденное таким образом решение в уравнение для x_1, и решая его, неизвестные коэффициенты A_0, Ω_1 найдем из условий периодичности

$$x_1(0) = x_1(2 \cdot \pi), \qquad x'_1(0) = x'_1(2 \cdot \pi).$$

Нахождение приближенного периодического решения в форме ряда (3), таким образом, выливается в рекуррентную процедуру.

Изложенные выше алгоритмы (функция “muserie” и функция “periodical”) реализованы в системе MATLAB 6.0 в виде пакета функций, представленных в приложении. Реализована также интерфейсная часть (функция “poincare” без параметров) с естественной формой ввода информации через первое графическое окно. Правая часть уравнения (1) должна задаваться в виде многочлена переменных x, x'. Отображение найденного приближенного аналитического решения производится во второе

графическое окно в виде формул. В третье графическое окно результаты выводятся в виде кривых на фазовой плоскости. Кроме того, в это же графическое окно выводятся результаты, найденные с помощью функции ode45.

В качестве примера использования пакета приведем задачу нахождения решения и предельного цикла для уравнения Ван дер Поля

$$x''+x=\mu\cdot(1-x\cdot x)\cdot x',\ \mu<<1$$

с условиями

$$x(0)=A, x'(0)=0.$$

На рис. 1 представлено графическое окно интерфейсной части пакета, в котором указывается тип искомого решения, информация о максимальной степени разложения по малому параметру. Кроме того, для построения численного решения и решения в графической форме вводится информация об интервале интегрирования и значении малого параметра.

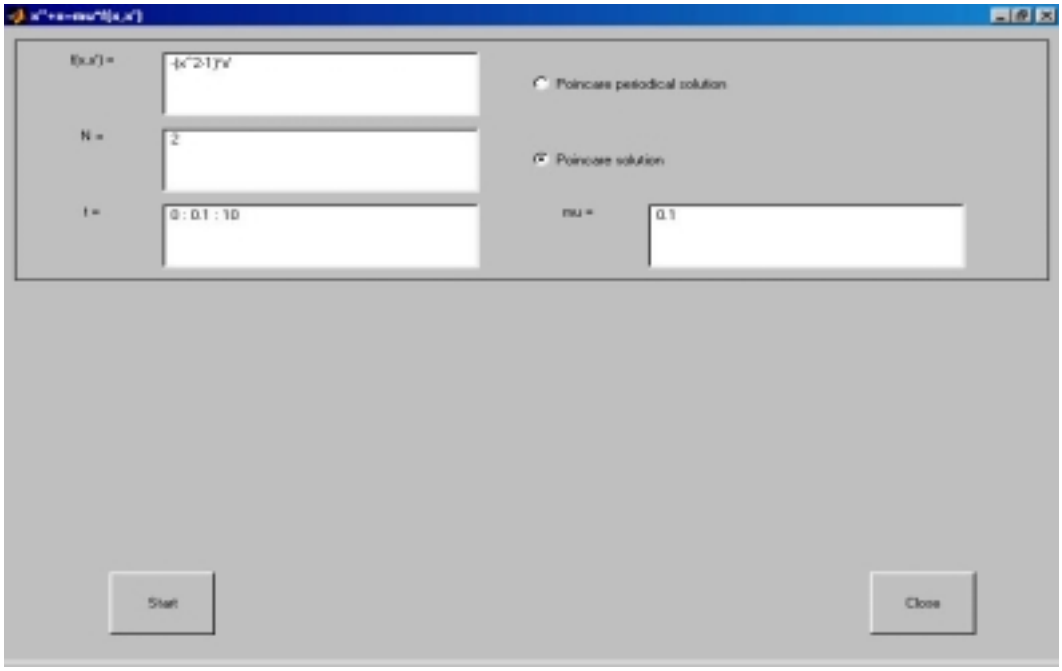


Рис. 1. Графическое окно интерфейсной части пакета.

На рис.2 представлено найденное аналитическое решение. На рис.3 найденное решение отображается на фазовой плоскости в графической форме и сравнивается с решением, найденным на заданном интервале с помощью функции ode45 системы MATLAB.

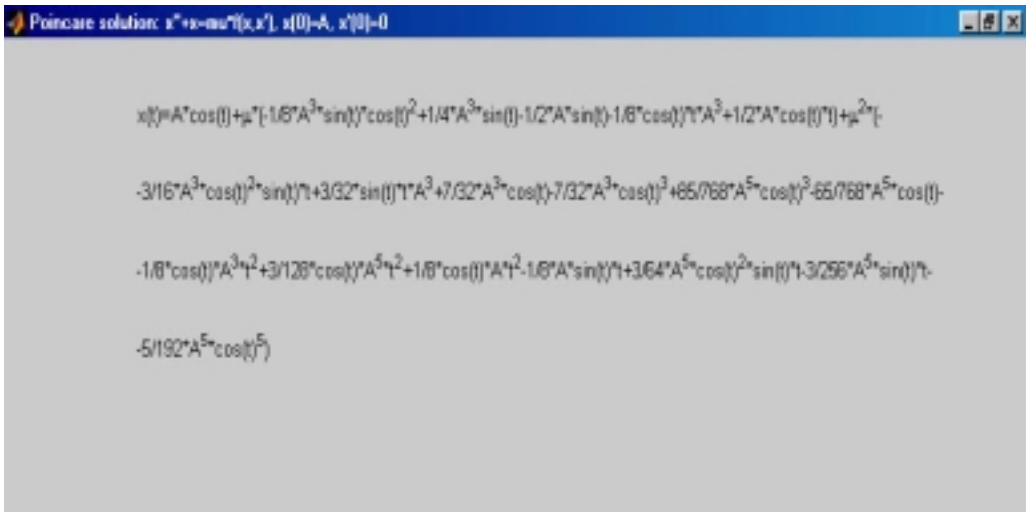


Рис. 2. Приближенное аналитическое решение уравнения Ван дер Поля.

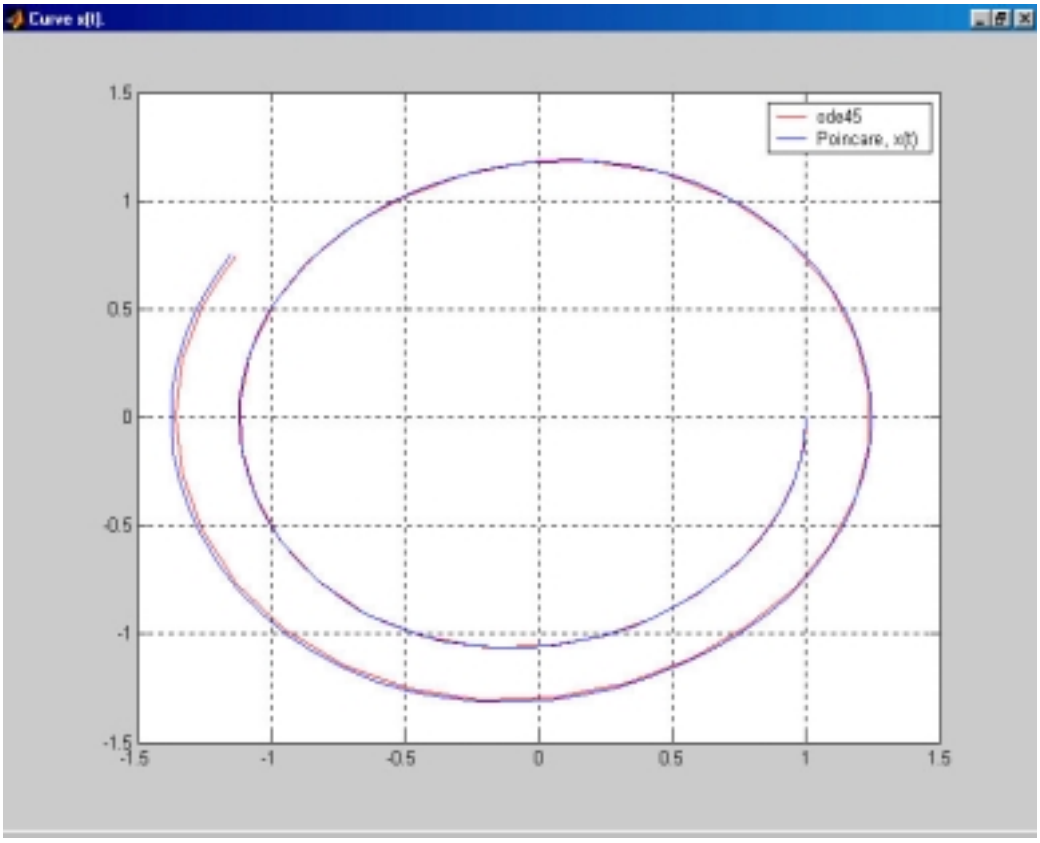


Рис. 3. Приближенное аналитическое решение уравнения Ван дер Поля и решение, найденное с помощью функции ode45 в графической форме.

На рис. 4 представлено приближенное периодическое аналитическое решение для уравнения Ван дер Поля. На рис. 5 отображен предельный цикл, соответствующий этому решению, а также решения, найденные с помощью функции ode45 для двух различных начальных условий.

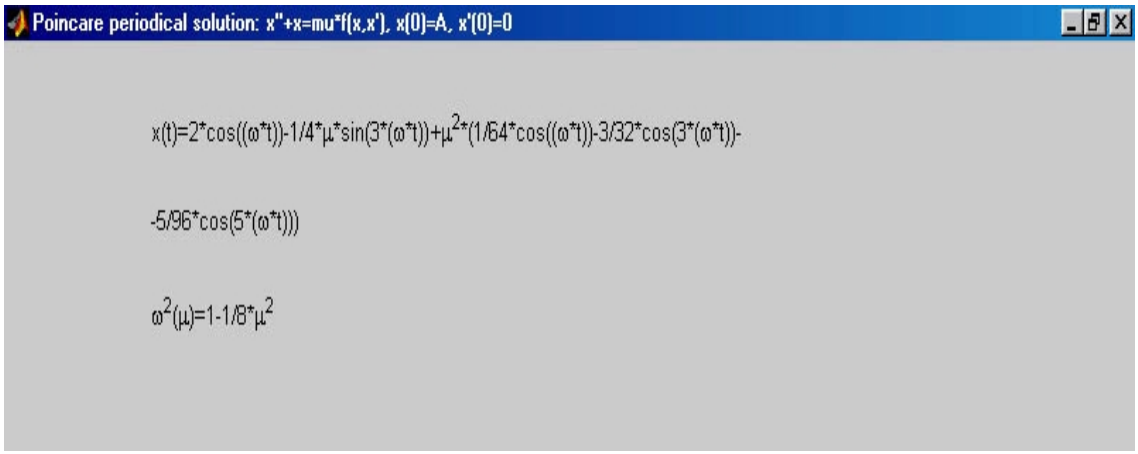


Рис. 4. Приближенное периодическое решение уравнения Ван дер Поля.

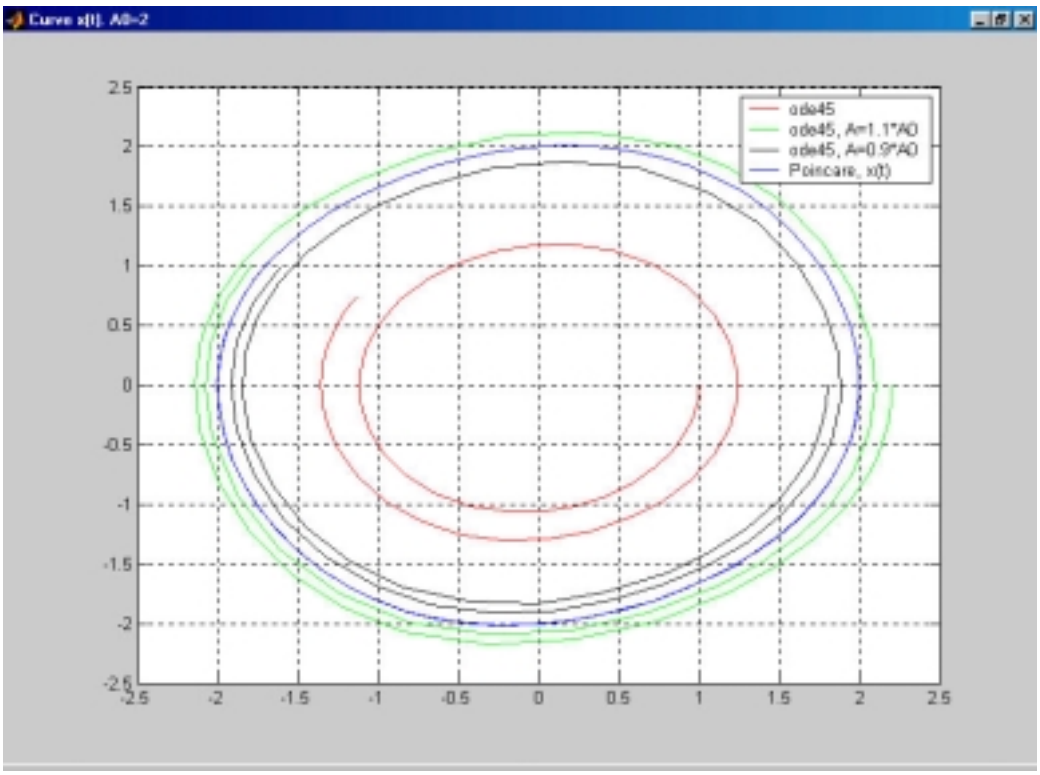


Рис. 5. Приближенное периодическое решение уравнения Ван дер Поля и решения, найденные с помощью функции ode45 в графической форме.

Литература

1. *Моисеев Н.Н.* Асимптотические методы нелинейной механики. М.: Наука, 1981. 400с.
2. *Бибиков Ю.Н.* Курс обыкновенных дифференциальных уравнений. М.: Высш. шк.,1991. 303с.
3. *Горяченко В. Д.* Элементы теории колебаний. Учебное пособие. Красноярск: Изд-во Краснояр. ун-та, 1995. 429с.

Приложение

```
function poincare(keyword, varargin)
%POINCARE Open GUI wrapper for 'muserie' and 'periodical' functions.
%  This function did not have any input or output parameters.
%  In graphical windows you may see solution and curve.
%  'muserie' used to find Poincare solution for differential equations, such as
%  x''+x=mu*f(x,x') with initial conditions x(0)=A, x'(0)=0.
%  'periodical' used to find Poincare periodical solution for differential
%  equations, such as x''+x=mu*f(x,x')
%  f(x,x') is polinom by variables (x,x')

%  Copyright 2001-2002 Michael Kulinich
%  $Revision 1.0 $ $Date: 2002/04/04 19:21:04 $

clear mu;
global mu A B;

p = .12*(1:7) - .02;
q = .60 - .14*(1:4);
r = [.10 .10];

A = 1; B = 0;

if nargin == 0
   blanks = '';

   init.f = char('-(x^2-1)*x''');
   init.t = '0 : 0.1 : 10';
   init.n = '1';
   init.fxlist = '';
   init.rb = 0;
   init.rb1 = 0;
   init.mu = '0.1';

   f = init.f;
   t = init.t;
   n = init.n;
   fxlist = sym(init.fxlist);
   rb = init.rb;
   rb1 = init.rb1;
   mu = init.mu;

   figg = figure('units', 'normalized', 'pos', [.01 .05 .45 .40], ...
               'menu', 'none', 'tag', 'figg', ...
               'NumberTitle', 'off', ...
               'Name', 'Curve x(t).');

   figsol = figure('units', 'normalized', 'pos', [.01 .50 0.940 .40], ...
                'menu', 'none', 'tag', 'figsol', ...
                'name', '', ...
                'NumberTitle','off', ...
                'Visible','on', ...
                'DoubleBuffer','on', ...
                'BackingStore','off', ...
                'Colormap',[], ...
                'NextPlot', 'replace');

   axes( ...
               'Units', 'normalized', ...
```

```
                'Visible', 'off', ...
                'NextPlot', 'replace');

    IF_figp = figure('units', 'normalized', 'pos', [.50 .05 .45 .40], ...
                'menu', 'none', ...
                'Name', 'x""+x=mu*f(x,x")', ...
                'tag', 'IF_figp', ...
                'NumberTitle', 'off', ...
                'Color', get(0, 'DefaultUIControlBackgroundColor'), ...
                'DefaultUIControlUnit', 'norm', 'UserData', fxlist);

    figure(IF_figp)
    set(gca, 'Visible', 'off');
    uicontrol('style', 'frame', 'pos', [0.01 0.60 0.98 0.38]);

    uicontrol('style', 'text', 'string', 'f(x,x") = ', 'pos', [0.04 0.86 0.09 0.10], ...
          'tag', 'fobj', 'UserData', f);
    uicontrol('style', 'text', 'string', 'N = ', 'pos', [0.04 0.74 0.09 0.10], ...
          'tag', 'nnum', 'UserData', n);
    uicontrol('style', 'text', 'string', 't = ', 'pos', [0.04 0.62 0.09 0.10], ...
          'tag', 'tstr', 'UserData', t);
    uicontrol('style', 'text', 'string', 'mu = ', 'pos', [0.50 0.62 0.09 0.10], ...
          'tag', 'mu', 'UserData', mu);

    uicontrol('pos', [.15 .86 .30 .10], 'style', 'edit', 'horiz', 'left', ...
          'backgroundcolor', 'white', ...
          'string', [blanks char(f)], 'tag', 'Sf', ...
          'CallBack', 'poincare Sfcallback');
    uicontrol('pos', [.15 .74 .30 .10], 'style', 'edit', 'horiz', 'left', ...
          'backgroundcolor', 'white', ...
          'string', [blanks n], 'tag', 'Sn', ...
          'CallBack', 'poincare Sncallback');
    uicontrol('pos', [.15 .62 .30 .10], 'style', 'edit', 'horiz', 'left', ...
          'backgroundcolor', 'white', 'string', [blanks t], 'tag', 'St', ...
          'CallBack', 'poincare Stcallback');
    uicontrol('pos', [.61 .62 .30 .10], 'style', 'edit', 'horiz', 'left', ...
          'backgroundcolor', 'white', 'string', [blanks mu], 'tag', 'Smu', ...
          'CallBack', 'poincare Smucallback');

    uicontrol('pos', [.50 .86 .42 .10], 'style', 'radiobutton', 'horiz', 'left', ...
          'tag', 'Srb', ...
          'string', 'Poincare periodical solution', 'UserData', 0, ...
          'CallBack', 'poincare Srbcallback');
    uicontrol('pos', [.50 .74 .42 .10], 'style', 'radiobutton', 'horiz', 'left', ...
          'tag', 'Srb1', ...
          'string', 'Poincare solution', 'UserData', 0, ...
          'Value', 1, ...
          'CallBack', 'poincare Srb1callback');


    uicontrol('pos', [p(7) q(4) r], 'string', 'Close', ...
          'CallBack', 'poincare close');
    uicontrol('pos', [p(1) q(4) r], 'string', 'Start', ...
          'CallBack', 'poincare start');
else
    switch keyword
    case 'Sfcallback'
       IF_figp = findobj(0, 'tag', 'IF_figp');
       f = get(gco, 'string');
       fhndl = findobj(IF_figp, 'tag', 'fobj');
```

```
    set(fhndl, 'UserData', f);
case 'Stcallback'
    IF_figp = findobj(0, 'tag', 'IF_figp');
    t = get(gco, 'String');
    set(findobj(IF_figp, 'tag', 'tstr'), 'UserData', t);
case 'Sncallback'
    IF_figp = findobj(0, 'tag', 'IF_figp');
    n = get(gco, 'String');
    set(findobj(IF_figp, 'tag', 'nnum'), 'UserData', n);
case 'Srbcallback'
    IF_figp = findobj(0, 'tag', 'IF_figp');
    rb = get(gco, 'Value');
    if rb ~= 1
       set(findobj(IF_figp, 'tag', 'Srb'), 'Value', 1);
    else
       set(findobj(IF_figp, 'tag', 'Srb'), 'UserData', rb);
       set(findobj(IF_figp, 'tag', 'Srb1'), 'UserData', 0);
       set(findobj(IF_figp, 'tag', 'Srb1'), 'Value', 0);
    end
case 'Srb1callback'
    IF_figp = findobj(0, 'tag', 'IF_figp');
    rb1 = get(gco, 'Value');
    if rb1 ~= 1
       set(findobj(IF_figp, 'tag', 'Srb1'), 'Value', 1);
    else
       set(findobj(IF_figp, 'tag', 'Srb1'), 'UserData', rb1);
       set(findobj(IF_figp, 'tag', 'Srb'), 'UserData', 0);
       set(findobj(IF_figp, 'tag', 'Srb'), 'Value', 0);
    end
case 'Smucallback'
    IF_figp = findobj(0, 'tag', 'IF_figp');
    mu = get(gco, 'string');
    set(findobj(IF_figp, 'tag', 'mu'), 'UserData', mu);
case 'close'
    close(findobj(0, 'tag', 'figg'));
    close(findobj(0, 'tag', 'figsol'));
    close(findobj(0, 'tag', 'IF_figp'));
    if exist('ff.m', 'file'), delete('ff.m');, end
case 'start'
    clear ff;
    IF_figp = findobj(0, 'tag', 'IF_figp');

    t = get(findobj(IF_figp, 'tag', 'tstr'), 'UserData');
    n = str2num(char(get(findobj(IF_figp, 'tag', 'nnum'), UserData')));
    figg = findobj(0, 'tag', 'figg');
    figsol = findobj(0, 'tag', 'figsol');
    f = get(findobj(IF_figp, 'tag', 'fobj'), 'UserData');
    rb = get(findobj(IF_figp, 'tag', 'Srb'), 'UserData');
    mu = str2num(get(findobj(IF_figp, 'tag', 'mu'), 'UserData'));

    waitH = waitbar(0, 'Please wait...');

    [min, step, max] = parsetime(t);
    writeff(f);
    svmu = mu;

    if rb == 0
       [r, r1, ax] = muserie(n, parseinput(f));

       s = strcat('x', '(t)=', char(r));
```

```
    disp(s);

    figure(figsol);
    set(figsol, 'Name', 'Poincare solution: x""+x=mu*f(x,x''), x(0)=A, x''(0)=0');
    cla

    s = strrep(s, 'mu', '\mu');
    L = 120;
    j = L; k = 1;
    for i = 1 : -0.1 : 0
       if j >= length(s)
          text(0, i, char(s(k : length(s))));
          break
       end
       while s(j) ~= '+' & s(j) ~= '-', j = j - 1;, end
       text(0, i, char(s(k : j)));
       k = j; j = j + L;
    end

    %%% FIGURE 2
    figure(figg);
    set(figg, 'Name', strcat('Curve x(t).'));
    clear figg

    [tt, y] = ode45('ff', [min max], [A B]);
    plot(y(:, 1), y(:, 2), 'r');
    hold on

    rr = evalexpr(r, min, step, max, mu);
    rr1 = evalexpr(r1, min, step, max, mu);
    plot(rr, rr1, 'b');
    hold off


    legend('ode45', 'Poincare, x(t)');
    grid on

else
    [r, r1, x_0, x1_0, rA, Omega, ax, rWo] = periodical(n, parseinput(f));

    s = strcat('x(t)=', char(rWo));
    disp(s);
    s1 = 0;
    if r ~= 0
       s1 = strcat('omega^2(mu)=', char(Omega));
       disp(s1);
    end

    if r == 0, s = 'No periodical solution';, end

    figure(figsol);
    set(figsol, 'Name', 'Poincare periodical solution: x""+x=mu*f(x,x''), x(0)=A, x''(0)=0');
    cla

    s = strrep(s, 'mu', '\mu');
    s = replaceb(s);
    L = 120;
    j = L; k = 1;
    i = 1;
    while i >= 0
```

```
            if j >= length(s)
               text(0, i, char(s(k : length(s))));
               break
            end
            while s(j) ~= '+' & s(j) ~= '-', j = j - 1;, end

            text(0, i, char(s(k : j)));
            k = j; j = j + L;
            i = i - 0.1;
         end

         s1 = strrep(s1, 'mu', '\mu');
         s1 = strrep(s1, 'omega', '\omega');
         s1 = replaceb(s1);
         j = L; k = 1;
         i = i - 0.1;
         while i >= 0
            if j >= length(s1)
               text(0, i, char(s1(k : length(s1))));
               break
            end
            while s1(j) ~= '+' & s1(j) ~= '-', j = j - 1;, end
            text(0, i, char(s1(k : j)));
            k = j; j = j + L;
            i = i - 0.1;
         end

         t = eval(sym(t));

         %%% FIGURE 2
         figure(figg);
         set(figg, 'Name', strcat('Curve x(t). A0=', char(rA)));
         clear figg

         [tt, y] = ode45('ff', [min max], [A B]);
         plot(y(:, 1), y(:, 2), 'r');
         hold on

         [tt, y] = ode45('ff', [min max], [1.1*double(rA) B]);
         plot(y(:, 1), y(:, 2), 'g');
         hold on

         [tt, y] = ode45('ff', [min max], [0.9*double(rA) B]);
         plot(y(:, 1), y(:, 2), 'k');
         hold on

         rr = evalexpr1(r, t, mu);
         rr1 = evalexpr1(r1, t, mu);
         plot(rr, rr1, 'b');
         hold off

         legend('ode45', 'ode45, A=1.1*A0', 'ode45, A=0.9*A0', 'Poincare, x(t)');
         grid on
      end
      close(waitH);
   end
end
return

function r = evalexpr(expr, min, step, max, mu)
```

```
global A B;
r = 0; i = 1; r = zeros(1, max/step + 1);
for t = min : step : max, r(i) = eval(expr);, i = i + 1;, end
return

function r = evalexpr1(expr, t, mu)

global A B;
if expr ~= 0, r = eval(expr);, else r = 0;, end
return

function r = replaceb(s)
ss = char(s);

i = 1;
while i <= length(ss)
   if ss(i) == '^'
      i = i + 1;
      if ss(i) == '('
         ss(i) = '{';
         while ss(i) ~= ')', i = i + 1;, end
         ss(i) = '}';
      end
   end
   i = i + 1;
end

r = ss;
return
%%% END



function [answ, dansw, x] = muserie(N, func)
clear mu;
syms A B f x t mu y sx sxc sxcs rpi rps;

f = A * cos(t);
x = initx(N);

x(1) = f;
for i = 1 : N, i
   sx   = rightp(lsum(N, mu, x), func);
   sxc  = collect(sx, mu);
   sxcs = maple('series', sxc, mu, i + 1);
   for j = 1 : i, fx(j) = simple(maple('coeff', sxcs, mu, j - 1));, end
   rpi  = expand(int(fx(i) * sin(y - t), t, 0, y));
   rps  = subs(rpi, y, t, 0);
   x(i + 1) = rps;
end

answ  = 0; for i = 1 : N + 1, answ = answ + mu^(i - 1) * x(i);, end
dansw = diff(answ, 't', 1);
return
%%% END

function [answ, dansw, x0, dx0, rA, Omega, x, answWo] = periodical(N, func)
clear mu;
syms x f A t omega mu l rpf d rp rpf rpfc eqrp e;
```

```
f     = A * cos(t);
x     = initx(N + 1);
omega = initomega(N + 1);
answ  = 0;
answo = 1;
rA    = 1;
frA   = 0;

x(1) = f;
for i = 1 : N + 1, i
   clear A;
   syms A;
   sx   = mu * rightp(lsum(i, mu, x), func);
   pp1  = maple('series', sx, mu, i + 1);
   for j = 1 : i, fx(j) = maple('coeff', pp1, mu, j);, end

   rp   = simple(fx(i) - omegax(i, x, omega));
   srp  = simplify(rp);
   rpf  = maple('map', 'combine', srp, 'trig');
   rpf  = collect(rpf, 'cos');
   rpf  = collect(rpf, 'sin');
   rpf  = collect(rpf);
   rpfc = coeffs_sf(rpf);
   if rpfc(3) == 0, omega(i) = solve(rpfc(2), omega(i));
   else
      %%% HACK OF SOLVE FUNCTION
      rpfc(3) = 2*rpfc(3);
      [q, w] = solve(sym(rpfc(2)), sym(rpfc(3)), sym(omega(i)), sym('A'));
      good = 0;
      for k = 1 : length(q)
         if isreal(double(q(k))), good = double(q(k));, break;, end
      end
      for k = 1 : length(q)
         if isreal(double(q(k)))
            if good < double(q(k)), good = double(q(k));, end
         end
      end
      index = find(double(q) == good);


      omega(i) = w(index);
      x(i) = subs(x(i), 'A', q(index), 0);
      rpfc = subs(rpfc, 'A', q(index), 0);

      if frA == 0, rA = q(index);, frA = 1;, end
   end

   rpfc(2) = 0;
   eqrp    = rpfc(1);                          % a0
   for j = 4 : 2 : length(rpfc), n = j / 2;
      eqrp = eqrp - (1/((n)^2-1))*(rpfc(j)*cos(n*t)+rpfc(j+1)*sin(n*t));
   end

   d = -diff(feqrp(0, rpfc), 't', 1);
   x(i + 1) = A * cos(t) + d * sin(t) + eqrp;
end

for i = 0 : N, answ = answ + mu^i * x(i + 1);, end
answWo = answ;
for i = 1 : N, answo = answo + mu^i * omega(i);, end
```

```
sanswo = simple(sym(sqrt(answo)));
Omega = answo;
if sanswo ~= 1
   answ = subs(answ, 't', sanswo * t, 0);
   answWo = subs(answWo, 't', '\omega*t', 0);
else
   answ = subs(answ, 't', t, 0);
   answWo = subs(answWo, 't', t, 0);
end
dansw = diff(answ, 't', 1);
x0 = x(1);
dx0 = diff(x(1), 't', 1);
return

function r = feqrp(t, rpfc)
r = rpfc(1);
for j = 4 : 2 : length(rpfc), n = j / 2;
   r = r - (1/((n^2)-1))*(rpfc(j)*cos(n*t)+rpfc(j+1)*sin(n*t));
end
return

function su = initomega(n)
for k = 1 : n, su(k) = sym(sprintf('omega%d', k));, end
return

function su = omegax(n, x, omega)
su = 0; for i = 1 : n, su = su + omega(i) * diff(x(n - i + 1), 't', 2);, end
return
%%% END

function su = initx(n)
for k = 1 : n + 1, su(k) = sym(sprintf('x%d', k - 1));, end
return
%%% END
function su = lsum(n, mu, x)
su = 0; for k = 0 : n - 1, su = su + mu^k * x(k + 1);, end
return
%%% END
function r = coeffs_sf(s)
syms r M;

ss = char(s);
k  = 1;

pa0 = cut_coeff(ss, 1, 'lr');
if findstr(ss(1 : pa0), 'cos') > 0, pa0 = 0;, end
if findstr(ss(1 : pa0), 'sin') > 0, pa0 = 0;, end
if pa0 > 0, M(k) = sym(ss(1 : pa0));, k = k + 1;,
elseif pa0 == 0, M(k) = 0;, k = k + 1;, end

numc = coeffs_num(ss);

cs  = 1;
svk = 1;
nr  = 0;
while svk <= numc
  if cs == 1
    if svk ~= 1, sv = sprintf('cos(%d*t)', svk);, else, sv = 'cos(t)';, end
    cs = 0; nr = nr + 1;
  else
```

```
    if svk ~= 1, sv = sprintf('sin(%d*t)', svk);, else, sv = 'sin(t)';, end
    cs = 1; nr = nr + 1;
  end

  if nr == 2, svk = svk + 1;, nr = 0;, end

  n  = findstr(ss, sv);
  ns = size(n);

  if ns ~= 0
    M(k) = sym(maple('coeff', ss, sv));
    k    = k + 1; ne = 0;
  else
    M(k) = 0;
    k    = k + 1;
  end
end

r = M;
return
%%% END

function r = cut_coeff(s, pos, way)
j  = pos;
bn = 0;
if strcmp(way, 'rl') == 1
  while j > 1
    if s(j) == ')', bn = bn + 1;,
    elseif s(j) == '(', bn = bn - 1;,
    elseif bn == 0 & (s(j) == '+' | s(j) == '-'), break;, end
    j = j - 1;
  end
elseif strcmp(way, 'lr') == 1
  if s(j) == '+' | s(j) == '-', j = j + 1;, end
  while j <= length(s)
    if s(j) == '(', bn = bn + 1;,
    elseif s(j) == ')', bn = bn - 1;,
    elseif bn == 0 & (s(j) == '+' | s(j) == '-'), j = j - 1;, break;, end
    j = j + 1;
  end
else
  error('cut_coeff: "way" is unknown');
end
if j > length(s), j = length(s);, end
r = j;
return

function r = coeffs_num(s)
ss = char(s);
nc = findstr(ss, 'cos');
ns = findstr(ss, 'sin');
M  = 0; k = 1;

for i = 1 : length(nc)
  p = nc(i) + 4; np = p;
  while np <= length(ss) & length(str2num(ss(np : np))) > 0, np = np + 1;, end
  if np - p > 0, num = str2num(ss(p : np - 1));, else, num = 1;, end
  M(k) = num; k = k + 1;
end
```

```
for i = 1 : length(ns)
  p = ns(i) + 4; np = p;
  while np <= length(ss) & length(str2num(ss(np : np))) > 0, np = np + 1;, end
  if np - p > 0, num = str2num(ss(p : np - 1));, else, num = 1;, end
  M(k) = num; k = k + 1;
end

r = max(M);
return
%%% END
function r = parseinput(s)
ss = char(s);
r  = '';

i = 1; j = 1;
while i <= length(ss)
   if ss(i) == ''''
      r(j - 1) = 0;
      r = strcat(r, 'diff(x,"t",1)');
      j = j + 12;
   else
      r = strcat(r, ss(i));
      j = j + 1;
   end
   i = i + 1;
end
return
%%% END
function [min, step, max] = parsetime(s)
min = 0; step = 0.1; max = 0;
ss = char(s);
tt = 0;

%% culculate number of ':'
for i = 1 : length(ss)
   if ss(i) == ':', tt = tt + 1;, end
end

if tt == 1
   [lp, r] = strtok(ss, ':');
   min = str2num(lp); max = str2num(r(2:length(r)));
elseif tt == 2
   [lp, r] = strtok(ss, ':');
   min = str2num(lp);
   [lp, r] = strtok(r, ':');
   step = str2num(lp);
   max = str2num(r(2:length(r)));
else
   disp('Error while parsing time string');

   return
end

return
%%% END
function r = rightp(x, func)
r = eval(func);
return
%%% END
function writeff(s)
```

```
ss = char(s);
r  = '';
i  = 1;

r = strcat(r, 'r(2)=-y(1)+mu*(');
while i <= length(ss)
   if ss(i) == 'x'
      if (i < length(ss)) & (ss(i + 1) == '')
         r = strcat(r, 'y(2)');
         i = i + 1;
      else, r = strcat(r, 'y(1)');, end
   else
      r = strcat(r, ss(i));
   end
   i = i + 1;
end
r = strcat(r, ');');

fid = fopen('ff.m', 'w');

fprintf(fid, 'function r=ff(t,y)\n');
fprintf(fid, 'global mu;\n');
fprintf(fid, 'r(1)=y(2);\n');
fprintf(fid, r); fprintf(fid, '\n');
fprintf(fid, 'r=r'';\n');
fprintf(fid, 'return\n');

fclose(fid);
return
%%% END
```

УДК 681.142.2

ОПЫТ ИСПОЛЬЗОВАНИЯ СИСТЕМЫ MATLAB ПРИ ПРОВЕДЕНИИ ЛАБОРАТОРНЫХ РАБОТ ПО КУРСУ «ТЕОРИЯ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ»

Малицкий П.М.
Московский государственный университет (МГТУ) «СТАНКИН»
e-mail: mic_@softhome.net

За последние полвека функциональные возможности и направления применения компьютеров развивались стремительно, оказывая огромное влияние на эволюцию научной и технической мысли. Особенно значительны перемены в теории и практике разработки систем управления, где на всех этапах проектирования, производства, функционирования систем в целом и отдельных ее подсистем в настоящее время, как правило, используют компьютерные средства и технологии.

Поэтому такие курсы, как «Основы автоматики», «Теория автоматического управления» и другие аналогичные им, включены в учебные планы различных инженерных специальностей и являются одним из важнейших элементов общетехнического образования.

В МГТУ «СТАНКИН» разработана примерная программа дисциплины «Теория автоматического управления», рекомендованная Министерством образования России для направлений подготовки (специальностей) в области техники и технологии, сельского и рыбного хозяйства. Программа написана в соответствии с Государственным образовательным стандартом высшего профессионального образования для направлений подготовки (специальностей) 550200 – Автоматизация и управление; 552900 – Технология, оборудование и автоматизация машиностроительных производств; 553000 – Системный анализ и управление. Авторы программы: заведующий кафедрой «Электротехника, электроника и автоматика» МГТУ «СТАНКИН», д.т.н., профессор Кузовкин В.А. и профессор СПбГТУ Козлов В.Н. Подготавливается к изданию полный комплект методических материалов по курсу «Теория автоматического управления», включающий в себя учебник «Теория автоматического

управления», написанный в соответствии с указанной выше примерной программой, учебное пособие «Теория автоматического управления в примерах и задачах» (издано в 2001 году) и Методические указания к выполнению лабораторно-вычислительного практикума (находится в стадии апробации).

В МГТУ «СТАНКИН» методы теории автоматического управления изучаются в течение одного семестра на всех технических факультетах в дисциплине «Теория автоматического управления», в течение двух семестров в дисциплине «Основы автоматического управления» (специальность 190100 «Приборостроение») и в течение одного семестра в магистерском курсе «Цифровое управление техническими системами».

Каждая лабораторная работа по курсу «Теория автоматического управления» проводится после того, как на лекции прочитан материал, относящийся к тематике данной лабораторной работы, и проведено семинарское занятие, на котором аналитические методы теории закреплены путем решения типовых задач.

Лабораторный практикум будет включать описания семи лабораторных работ (из которых ниже будут рассмотрены только первые три):

1. Исследование характеристик типовых звеньев системы автоматического управления.
2. Анализ соединений типовых звеньев.
3. Анализ устойчивости линейных систем автоматического управления.
4. Синтез линейных систем автоматического управления.
5. Исследование характеристик линейных импульсных систем автоматического управления.
6. Исследование характеристик основных структур дискретно-непрерывных систем автоматического управления.
7. Анализ устойчивости линейных дискретных систем автоматического управления.

Система MATLAB имеет бесспорные преимущества по сравнению с другими системами с точки зрения проведения перечисленных лабораторных работ, так как ряд ее пакетов (в частности, пакеты анализа и синтеза систем управления) специально предназначен для решения самых разнообразных задач теории и практики автоматического управления.

Описание каждой лабораторной работы включает в себя:

1) цель работы,

2) вопросы для подготовки к работе,

3) порядок выполнения работы,

4) порядок оформления отчета с результатами и выводами.

Так как каждой лабораторной работе предшествует семинарское занятие, на котором студенты самостоятельно решают типовые задачи, то помимо обязательного выполнения предписанного в описании перечня заданий, студенты проверяют с помощью системы MATLAB правильность полученных на семинарском занятии решений и в отчете по лабораторной работе делают соответствующие выводы. Каждая лабораторная работа защищается студентом индивидуально.

ЛАБОРАТОРНАЯ РАБОТА 1.
Исследование характеристик типовых звеньев систем автоматического управления

Цель работы: исследование влияния параметров типовых звеньев САУ на частотные и временные характеристики с помощью пакета программ MATLAB.

Вопросы для подготовки к работе:

1. Передаточная функция звена: определение (математическое), свойства, связь с его дифференциальным уравнением.
2. Временные характеристики звена (переходная и весовая): определения (математические), связь между ними.
3. Частотные характеристики звена и их физическая интерпретация; построение частотных характеристик с использованием передаточной функции.
4. Методы построения АФЧХ звена по известным его АЧХ и ФЧХ, а также построения АЧХ и ФЧХ по известной АФЧХ.
5. Типовые звенья; классификация, дифференциальные уравнения.

$$W_i(p) = \frac{K_i}{1+T_i p}$$

6. Примерные графики частотных и временных характеристик рассматриваемых в данной работе типовых звеньев.

Порядок выполнения работы

1.1. Исследовать характеристики трех инерционных звеньев с передаточной функцией

$$W_i(p) = \frac{K_i}{p}.$$

Параметры K_i и T_i звеньев взять из табл. 1.

1.2. Исследовать характеристики трех интегрирующих звеньев с передаточной функцией. Параметры K_i звеньев взять из табл. 1.

1.3. Исследовать характеристики трех дифференцирующих звеньев с передаточной функцией

$$W_i(p) = K_i p$$

Параметры K_i звеньев взять из табл. 1.

1.4. Исследовать характеристики трех форсирующих звеньев с передаточной функцией:

$$W_i(p) = K_i(1 + T_i p).$$

Параметры Ki и Ti звеньев взять из табл. 1.

1.5. Исследовать характеристики трех колебательных звеньев (1, 2 и 3) с передаточной функцией

$$W_i(p) = \frac{K_i}{T^2_i p^2 + 2\xi_i T_i p + 1}.$$

Параметры K_i, ξ_i и T_i звеньев взять из таблицы 2.

1.6. Исследовать характеристики трех колебательных звеньев (4,5 и 6) и четвертого звена с $\xi_4 = 0$.

2. Для каждого из перечисленных в пунктах 1.1- 1.6 на дисплее ПЭВМ получить совмещенные кривые АЧХ, ФЧХ, АФЧХ, переходной и импульсной функций всех заданных в каждом пункте звеньев.
3. Полученные совмещенные кривые перенести на миллиметровую бумагу, изменив масштаб таким образом, чтобы исходные данные, текст задания и все характеристики располагались на листе формата А4.
4. Сделать вывод о влиянии параметров звеньев на поведение частотных и временных характеристик.
5. Для звеньев, у которых переходная функция $h(t)$ сходится при $t \to \infty$ к установившемуся значению $h_{уст}$, проанализировать влияние коэффициента передачи K и постоянной времени T на величину $h_{уст}$ и времени регулирования t_p.
6. Для переходных функций колебательных звеньев исследовать связь частоты колебаний $\omega = 2\pi/T$ и декремента затухания D с коэффициентом демпфирования ξ.
7. Оформить отчет о проделанной лабораторной работе и защитить его.

Таблица 1.

Параметры инерционных звеньев

Вариант №	Звено 1		Звено 2		Звено 3	
	K_1	T_1 [мс]	K_2	T_2[мс]	K_3	T_3[мс]
1	2	3	4	5	6	7
1	5	25	5	75	10	25
2	5	30	5	90	10	30
3	5	40	5	120	10	40
4	5	15	5	45	10	15
5	5	20	5	60	10	20
6	5	35	5	105	10	35
7	4	25	4	75	8	25
8	4	30	4	90	8	30
9	4	35	4	105	8	35
10	4	20	4	60	8	20
11	4	15	4	45	8	15
12	4	40	4	120	8	40
13	3	15	3	45	6	15
14	3	20	3	60	6	20
15	3	25	3	75	6	25
16	3	30	3	90	6	30
17	3	35	3	105	6	35
18	3	40	3	120	6	40

Параметры колебательных звеньев (для всех звеньев K = 10)

Таблица 2

Ва-ри-ант	Звено 1		Звено 2		Звено 3		Звено 4		Звено 5		Звено 6	
№	T_1 [мс]	ξ_1	T_2 [мс]	ξ_2	T_3 [мс]	ξ_3	T_4 [мс]	ξ_4	T_5 [м]	ξ_5	T_6 [мс]	ξ_6
1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.1	0.4	1	0.4	10	0.4	1	0.1	1	0.5	1	0.8
2	0.2	0.4	2	0.4	20	0.4	2	0.1	2	0.45	2	0.7
3	0.3	0.4	3	0.4	25	0.4	3	0.2	3	0.3	3	0.7
4	0.4	0.4	4	0.4	20	0.4	4	0.2	4	0.45	4	0.8
5	0.5	0.4	5	0.4	25	0.4	5	0.3	5	0.6	5	0.8
6	0.6	0.4	6	0.4	24	0.4	6	0.3	6	0.5	6	0.7
7	0.1	0.4	0.8	0.4	8	0.4	0.8	0.3	0.8	0.45	0.8	0.8
8	0.2	0.4	1.6	0.4	16	0.4	1.6	0.15	1.6	0.5	1.6	0.8
9	0.3	0.4	2.4	0.4	24	0.4	2.4	0.15	2.4	0.45	2.4	0.7
10	0.4	0.4	2.6	0.4	25	0.4	2.6	0.25	2.6	0.5	2.6	0.75
11	0.5	0.4	4	0.4	24	0.4	4	0.25	4	0.45	4	0.7
12	0.6	0.4	5	0.4	25	0.4	5	0.35	5	0.55	5	0.75
13	0.1	0.4	2	0.4	20	0.4	2	0.2	2	0.5	2	0.8
14	0.2	0.4	3	0.4	25	0.4	3	0.1	3	0.5	3	0.8
15	0.3	0.4	4	0.4	24	0.4	4	0.1	4	0.45	4	0.7
16	0.4	0.4	3	0.4	25	0.4	3	0.15	3	0.55	3	0.8
17	0.5	0.4	3	0.4	20	0.4	3	0.25	3	0.45	3	0.7
18	0.6	0.4	4	0.4	20	0.4	4	0.15	4	0.5	4	0.8

ЛАБОРАТОРНАЯ РАБОТА 2
Анализ соединений типовых звеньев

Цель работы: исследование соединений типовых звеньев САУ; анализ частотных характеристик соединений в зависимости от параметров звеньев

Вопросы для подготовки к работе:

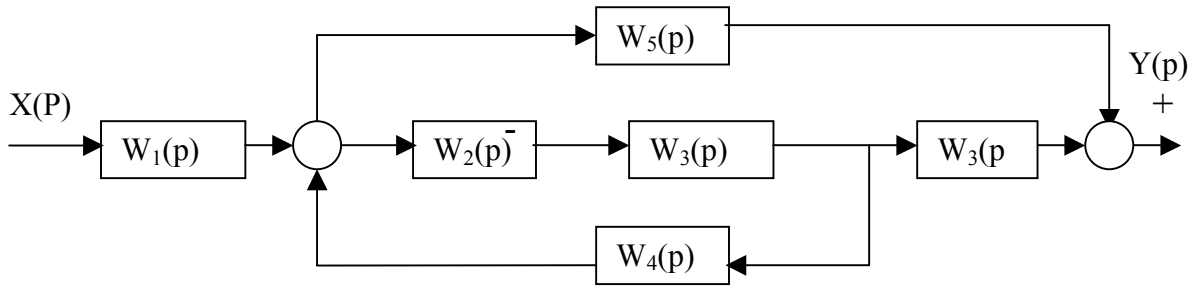
1. Структурная схема; основные правила составления структурных схем.
2. Определение передаточных функций системы, состоящей из звеньев с известными передаточными функциями, в случае:
) последовательного соединения двух звеньев;
) параллельного согласного соединения двух звеньев;
) параллельного встречного соединения (с обратной связью) двух звеньев.
3. Основные правила преобразования структурных схем.
4. Вычисление передаточной функции многоконтурной системы с использованием формулы Мэйсона.

Порядок выполнения работы

1. Исследовать соединения:
) последовательное для двух инерционных звеньев;
) параллельное согласное для двух инерционных звеньев;
) параллельное встречное с жесткой обратной связью;
) параллельное встречное с гибкой обратной связью.

Параметры звеньев взять из таблицы 1.

2. Для каждого из четырех видов соединений построить совмещенные кривые АЧХ, ФЧХ, и АФЧХ и переходной функции.
3. Полученные совмещенные кривые нанести на миллиметровую бумагу, изменив масштаб таким образом, чтобы исходные данные, текст задания и все характеристики располагались на листе формата А4.
4. Сделать выводы о влиянии параметров звеньев на поведение частотных характеристик для каждого вида соединения. Подтвердить выводы соответствующими расчетами.
5. Преобразовать соединение вида



путем перемещения:

а) звена через узел суммирования;

б) звена через узел разветвления.

6. Нарисовать схему каждого способа перемещения с последующей реализацией её на ПЭВМ. Параметры звеньев взять из табл. 1 и 2 по указанию преподавателя.
7. Правильность построения схем проверить путем совмещения характеристик АФЧХ для каждого способа перемещения элементов схемы.
8. Оформить отчет о проделанной лабораторной работе и защитить его.

ЛАБОРАТОРНАЯ РАБОТА 3
Анализ устойчивости линейных систем автоматического управления

Цель работы: исследование устойчивости линейных САУ.

Вопросы для подготовки к работе:

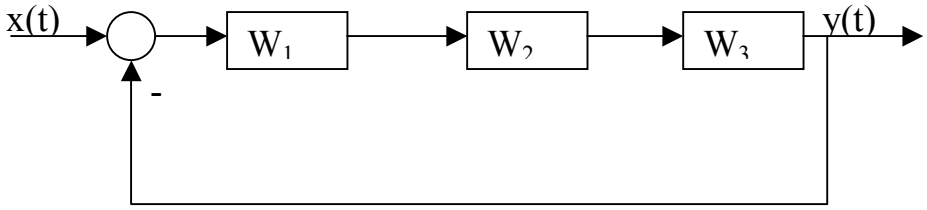
1. Устойчивость САУ: определение, необходимое условие устойчивости; алгебраические и частотные критерии устойчивости.
2. Корневой критерий устойчивости.
3. Критерии устойчивости Гурвица, Михайлова и Найквиста.
4. Запасы устойчивости по амплитуде и фазе и их использование.
5. Показатели качества, характеризующие процесс управления: установившаяся ошибка, время регулирования, перерегулирование и другие.
6. Связь величины установившейся ошибки с порядком астатизма САУ.
7. Зависимость показателей качества от расположения корней

характеристического уравнения.

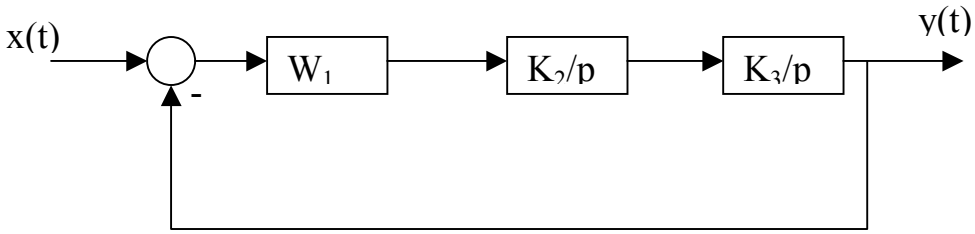
Порядок выполнения работы

1. В соответствии со структурной схемой, взятой из таблицы 3 по указанию преподавателя, смоделировать на компьютере замкнутую САУ.
2. Для заданной структурной схемы разомкнутой и замкнутой САУ определить корни характеристического уравнения и построить АФЧХ.
3. Определить устойчивость разомкнутой и замкнутой САУ и проверить вывод об устойчивости каждой из них расчетным путем, используя критерии Гурвица, Михайлова и Найквиста.
4. Определить предельное значение Кпред статического коэффициента усиления разомкнутой САУ, при котором она находится на границе устойчивости.
5. Исследовать устойчивость разомкнутой и замкнутой САУ при величинах статического коэффициента усиления $K = 0{,}95K_{пред}$ и $K_{пред}=1{,}05\ K_{пред}$ путем определения корней характеристического уравнения и построения АФЧХ. Для устойчивых систем определить запасы устойчивости по амплитуде и фазе.
6. Для шести колебательных звеньев (параметры которых взять из таблицы 1.1 по указанию преподавателя) исследовать влияние параметров звеньев на показатель качества (установившуюся ошибку, время регулирования, перерегулирование), на расположение корней характеристического уравнения и величины запасов устойчивости по амплитуде и фазе. При этом показатели качества следует определять по переходным функциям, а запасы устойчивости по АФЧХ звеньев.
7. Исследовать связь величины установившейся ошибки с порядком астатизма систем, построенных в соответствии со структурной схемой, взятой из таблицы 3.1 путем комбинирования инерционных и интегрирующих звеньев.
8. В процессе исследования получить совмещенные кривые требуемых графиков и перенести на миллиметровую бумагу, изменив масштаб таким образом, чтобы исходные данные, текст задания и все характеристики располагались на листе формата А4.
9. Оформить отчет о проделанной лабораторной работе и защитить его.

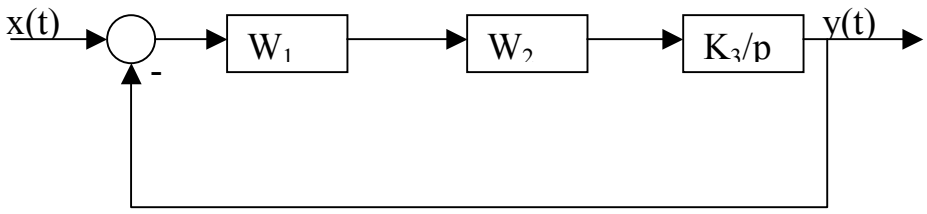
Пример 1.



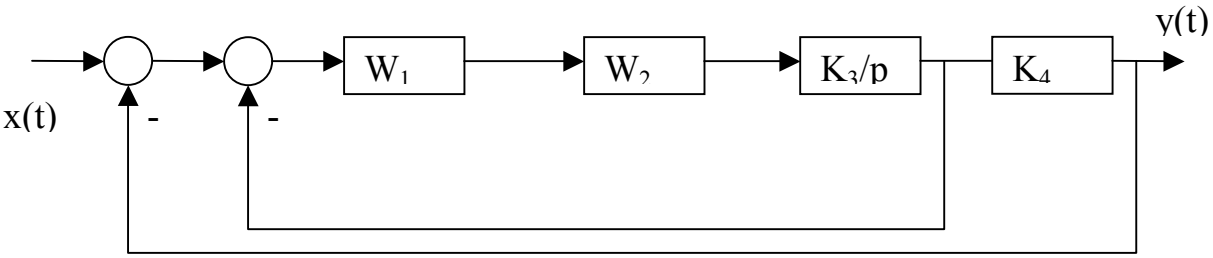
№	**K1**	T1[мс]	K2	T2[мс]	K3	T3[мс]
1	50	0.05	6	0.25	10	0.15
2	50	0.05	60	0.25	10	0.15
3	50	0.05	6	0.25	10	0.15
4	50	0.05	60	0.25	100	0.15



№	K1	T1[мс]	K2	K3
1	100	0.05	2	4
2	50	0.05	2	4
3	100	0.25	2	4
4	50	0.25	2	4



№	K_1	T_1[мс]	K_2	T_2[мс]	K_3
1	25	0.025	8	0.04	4
2	30	0.025	8	0.04	4
3	10	0.025	8	0.04	4
4	100	0.025	8	0.04	4



№	K_1	K_1	K_2	T_2[мс]	K_3	K_4
1	4	0.05	5	0.25	2	100
2	4	0.05	5	0.25	2	15
3	4	0.05	5	0.25	2	50
4	4	0.05	5	0.25	2	5

Рассмотрим примеры выполнения некоторых заданий лабораторной работы 1.

Пример 1.

Требуется построить частотные (АЧХ, ФЧХ и АФЧХ) и временные (переходную и весовую характеристики трех инерционных звеньев с передаточной функцией

$$W_i(p) = \frac{K_i}{1+T_i p},$$

$K_1 = 8$, $T_1 = 50$ мс,
$K_2 = 8$, $T_2 = 100$ мс,
$K_3 = 16$, $T_3 = 50$ мс.

Параметры звеньев выбраны таким образом, что у первого и второго звеньев коэффициенты передачи K равны, а постоянные времени T – разные; у первого и третьего звеньев наоборот: равны постоянные времени T, а коэффициенты передачи K – разные. Благодаря такому выбору параметров совмещение частотных и

временных характеристик на одном графике значительно облегчит исследование указанных характеристик в зависимости от того или иного параметра.

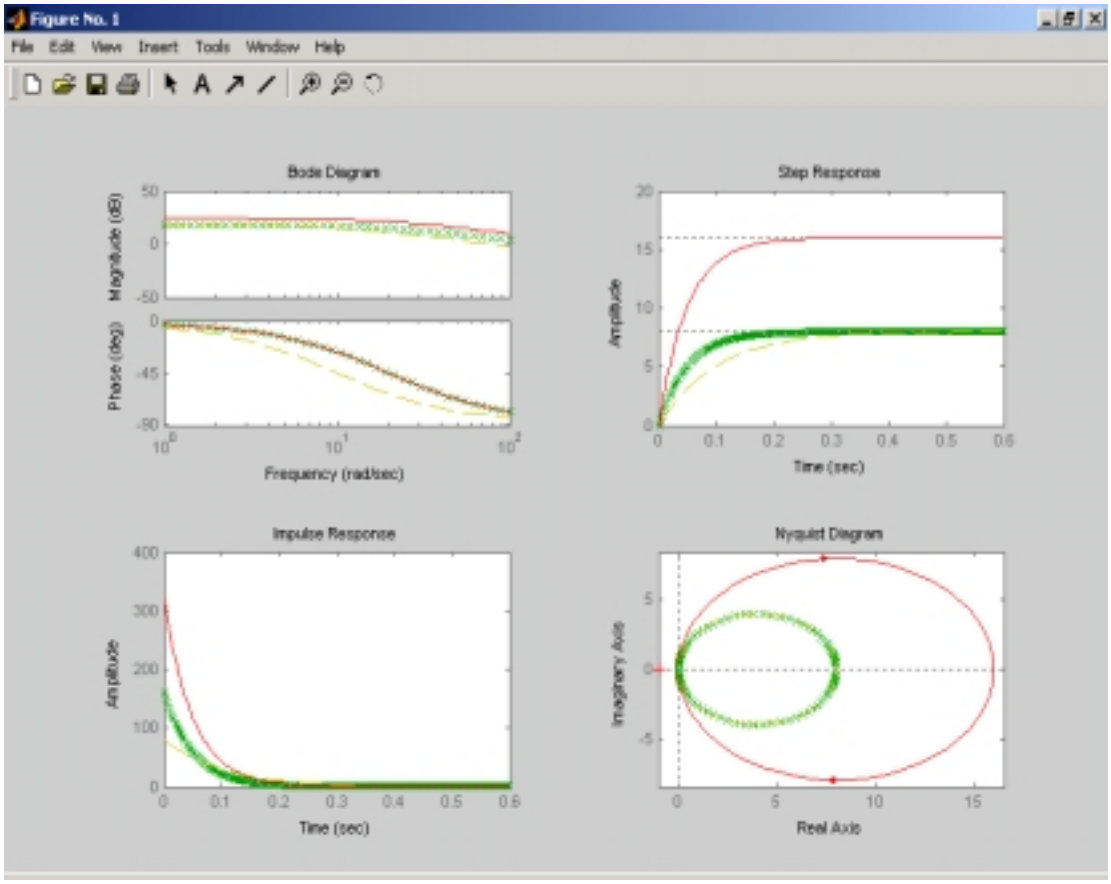


Рис. 1.

Запишем текст программы анализа на языке системы MATLAB:

```
i1=tf([8],[0.05 1])
i2=tf([8],[0.1 1])
i3=tf([16],[0.05 1])
subplot(221)
bode(i1,'gx',i2,'y--',i3,'r')
subplot(222)
step(i1,'gx',i2,'y--',i3,'r')
subplot(223)
impulse(i1,'gx',i2,'y--',i3,'r')
subplot(224)
nyquist(i1,'gx',i2,'y--',i3,'r')
```

Первые строки программы задают три инерционных звена в

виде tf-моделей (т.е. в виде передаточных функций). Функция **subplot(m,n,p)** разбивает графическое окно на m×n подокон, при этом m – число подокон по горизонтали, n –число подокон по вертикали, а p – номер подокна, в которое будет выводиться текущий график (подокна отсчитываются последовательно по строкам). Функции **bode**, **step**, **impulse** и **nyquist** строят характеристики АЧХ и ФЧХ (**bode**), переходную (**step**), импульсную (**impulse**) и АФЧХ (**nyquist**) соответственно всех трех звеньев на одном графике, при этом характеристики первого звена строятся зелеными крестиками, второго звена – желтыми штриховыми линиями, третьего звена – сплошными красными линиями.

Результат выполнения данной программы – рис. 1.

Из построенных характеристик видно, у первого и второго звеньев с равными коэффициентами передачи К АФЧХ совпадают, а у первого и третьего звеньев с равными постоянными времени Т совпадают ФЧХ. Этот же результат можно получить и аналитическим путем.

Пример 2.

Построить частотные (АЧХ, ФЧХ и АФЧХ) и временные (переходную и весовую) характеристики трех колебательных звеньев с передаточной функцией

$$W_i(p) = \frac{K_i}{T^2_i p^2 + 2\xi_i T_i p + 1},$$

$$K_1 = 10,\ T_1 = 0.25 \text{ мс},\ \xi_1 = 0.35,$$
$$K_2 = 10,\ T_2 = 0.5 \text{ мс},\ \xi_2 = 0.35,$$
$$K_3 = 10,\ T_3 = 1.0 \text{ мс},\ \xi_3 = 0.35.$$

Параметры звеньев выбраны таким образом, что у всех трех звеньев коэффициенты передачи K и коэффициенты демпфирования ξ равны, а постоянные времени T – разные. Благодаря такому выбору параметров совмещение частотных и временных характеристик на одном графике значительно облегчит исследование указанных характеристик в зависимости от величины коэффициенты демпфирования.

Запишем текст программы анализа на языке системы MATLAB:

```
i1=tf([10],[0.0625 0.175 1])
i2=tf([10],[0.25 0.35 1])
i3=tf([10],[1 0.7 1])
subplot(221)
bode(i1,'gx',i2,'y--',i3,'r')
```

```
subplot(222)
step(i1,'gx',i2,'y--',i3,'r')
subplot(223)
impulse(i1,'gx',i2,'y--',i3,'r')
subplot(224)
nyquist(i1,'gx',i2,'y--',i3,'r')
```

Результат выполнения данной программы показан на рис. 2.

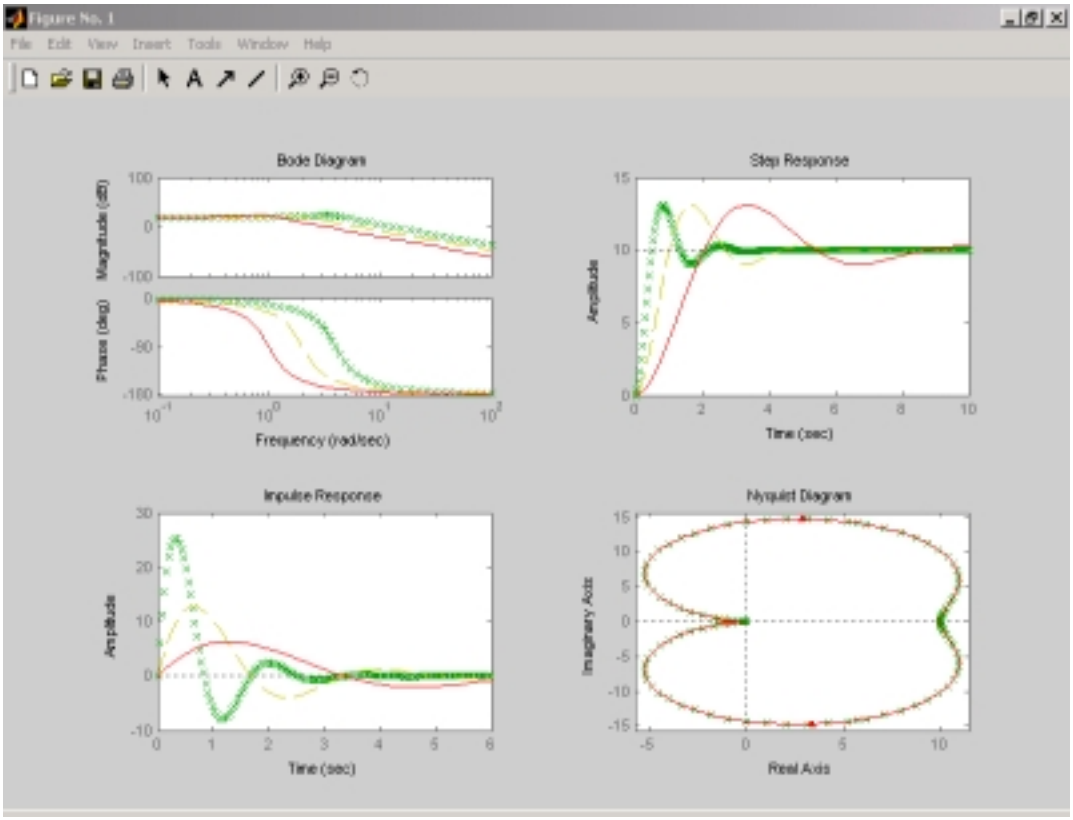


Рис. 2.

Из построенных характеристик видно, что АФЧХ всех трех звеньев совпадают, в то же время любая выбранная точка АФЧХ для каждого звена соответствует разным частотам. Этот же результат можно получить и аналитическим путем.

Пример 3.

Построить частотные (АЧХ, ФЧХ и АФЧХ) и временные (переходную и весовую) характеристики трех колебательных звеньев с передаточной функцией

$$W_i(p) = \frac{K_i}{T^2{}_i p^2 + 2\xi_i T_i p + 1},$$

$K_1 = 10$, $T_1 = 0.5$ мс , $\xi_1 = 0.1$
$K_2 = 10$, $T_2 = 0.5$ мс , $\xi_2 = 0.5$
$K_3 = 10$, $T_3 = 0.5$ мс , $\xi_3 = 0.8$

Параметры звеньев выбраны таким образом, что у всех трех звеньев коэффициенты передачи K и коэффициенты демпфирования ξ равны, а постоянные времени T – разные. Благодаря такому выбору параметров совмещение частотных и временных характеристик на одном графике значительно облегчит исследование указанных характеристик в зависимости от величины коэффициенты демпфирования.

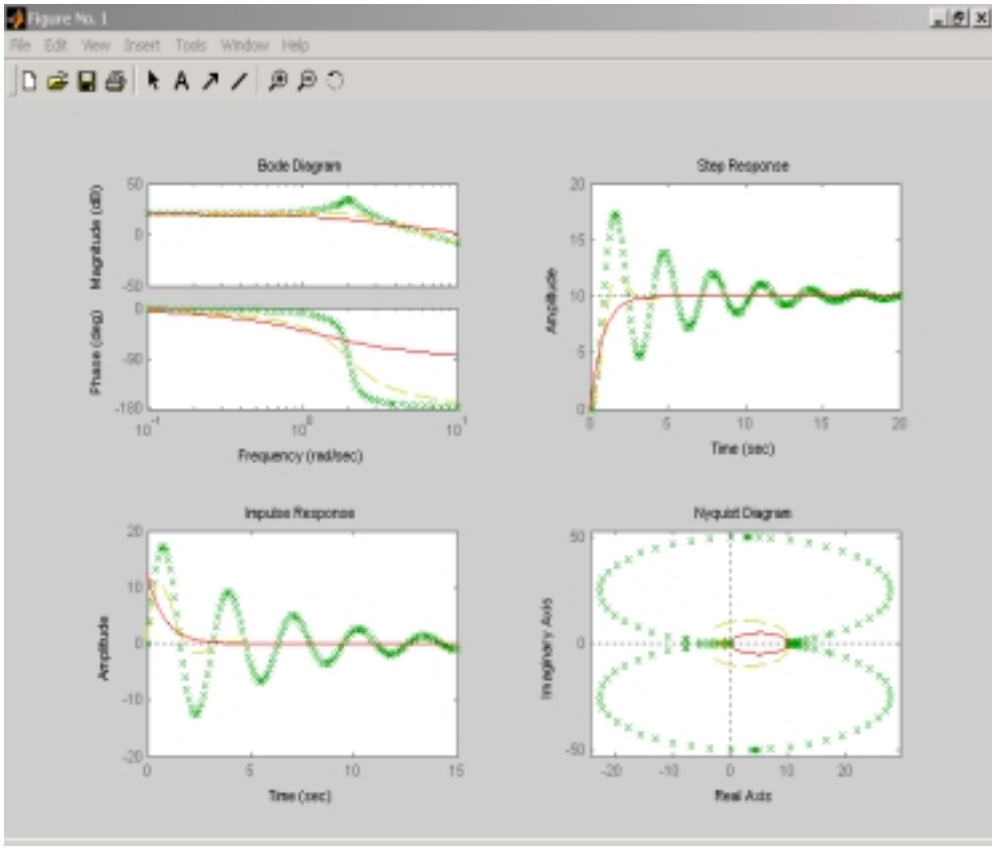


Рис. 3.

Запишем текст программы анализа на языке системы MATLAB:

```
i1=tf([10],[0.25 0.1 1])
i2=tf([10],[0.25 0.5 1])
i3=tf([10],[0.25 0.8 1])
subplot(221)
bode(i1,'gx',i2,'y--',i3,'r')
subplot(222)
```

```
step(i1,'gx',i2,'y--',i3,'r')
subplot(223)
impulse(i1,'gx',i2,'y--',i3,'r')
subplot(224)
nyquist(i1,'gx',i2,'y--',i3,'r')
```

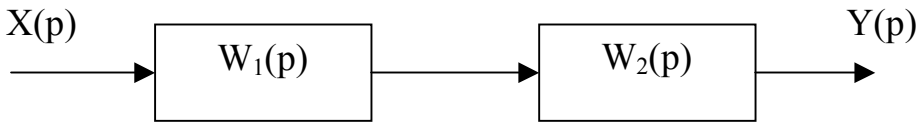
Результат выполнения данной программы показан на рис. 3.

Из построенных характеристик видно, что при малых коэффициентах демпфирования ξ у АФЧХ наблюдается всплеск при частоте излома характеристики, при больших – переходной процесс из колебательного становится апериодическим. Этот же результат можно получить и аналитическим путем.

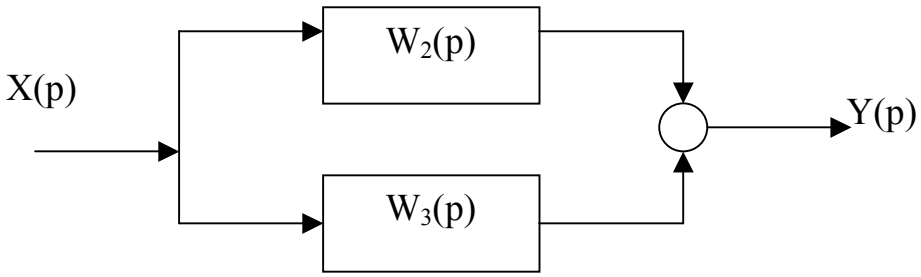
Рассмотрим примеры выполнения некоторых заданий лабораторной работы 2.

Из соединений типовых звеньев наиболее распространены три: последовательное, параллельное и параллельно-встречное (или с обратной связью):

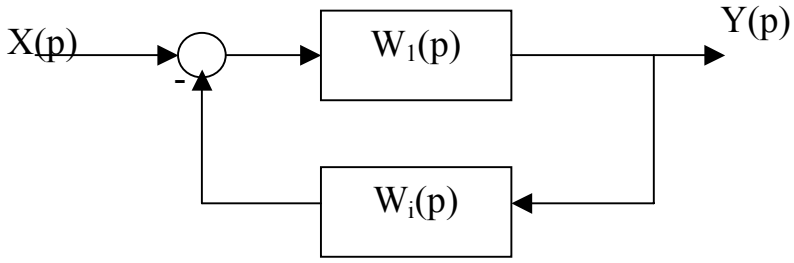
А) последовательное(series)



Б) параллельное (parallel)



В) с обратной связью (feedback)



Пример 4.

Построить частотные (АЧХ, ФЧХ и АФЧХ) и временные (переходную и весовую) характеристики четырех соединений инерционных звеньев с передаточной функцией

$$W_i(p) = \frac{K_i}{1+T_i p},$$

$K_1 = 8$, $T_1 = 50$ мс,
$K_2 = 8$, $T_2 = 100$ мс,
$K_3 = 16$, $T_3 = 50$ мс.

Для определенности реализуем последовательное (series) соединение первого и второго звеньев, параллельное (parallel) соединение второго и третьего звеньев, два соединения первого звена с жесткой и гибкой обратной связью (feedback) соответственно.

Запишем текст программы построения характеристик соединений на языке системы MATLAB:

```
1=tf([8],[0.05 1])
i2=tf([8],[0.1 1])
i3=tf([16],[0 1])
i4=tf([10],[0 1])
i5=tf([0.05 0],[0 1])
A=series(i1,12)
B=parallel(i2,i3)
C1=feedback(i1,i4)
C2=feedback(i1,i5)
subplot(221)
bode(A,'r.',B,'g--',C1,'y+',C2,'b')
subplot(222)
step(A,'r.',B,'g--',C1,'y+')
subplot(223)
impulse(A,'r.',B,'g--')
subplot(224)
nyquist(A,'r.',B,'g--',C1,'y+',C2,'b')
```

Первые строки программы определяют четыре соединения инерционных звена, заданных в виде tf-моделей (т.е. в виде передаточных функций): 1) последовательное (series) соединение первого и второго звеньев, 2) параллельное (parallel) соединение второго и третьего звеньев, 3) соединение (feedback) первого звена с жесткой обратной связью (т.е. со звеном с передаточной функцией Wi(p)=10), 4) соединение (feedback) первого звена с гибкой обратной связью (т.е. со звеном с передаточной функцией Wi(p)=0.05p).

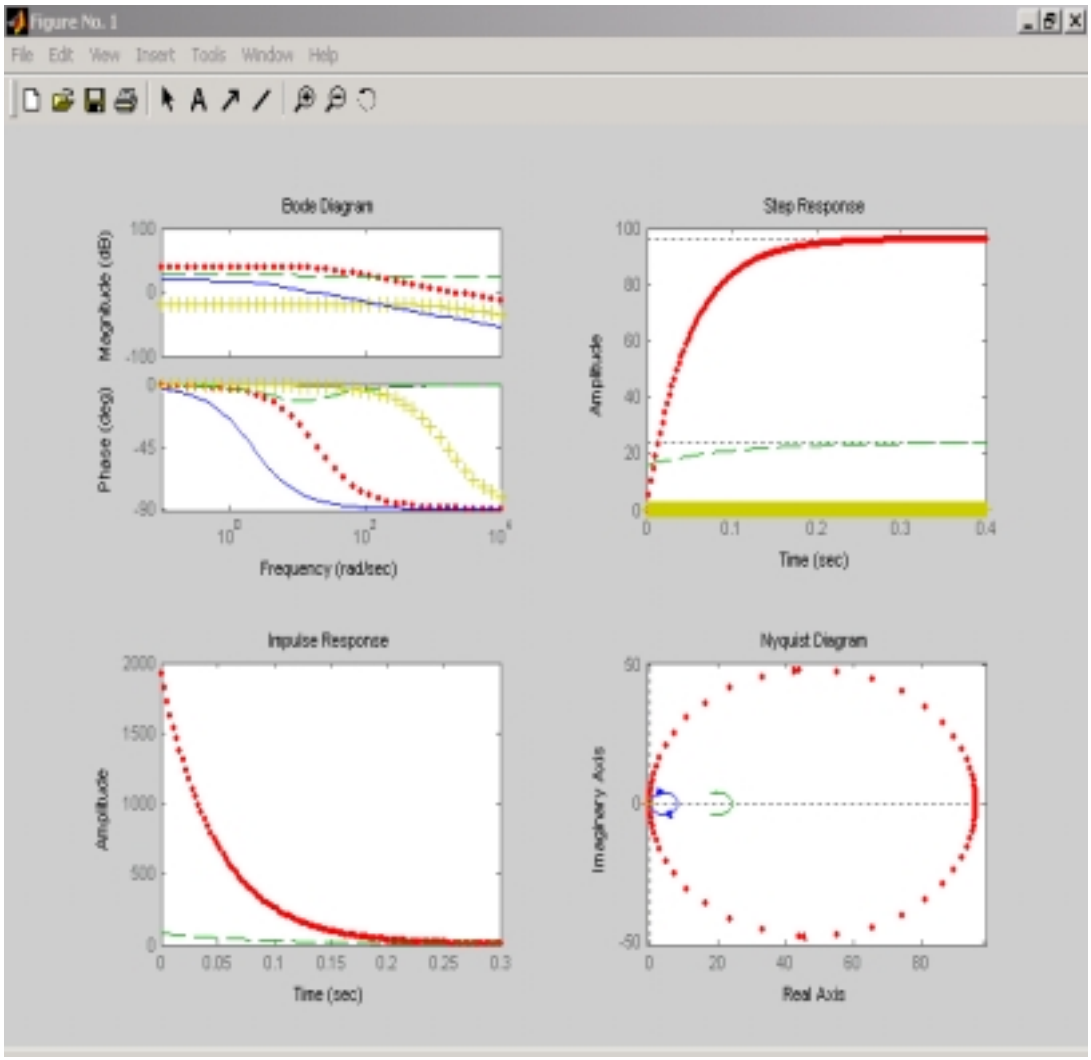


Рис. 4.

Функция **subplot(m,n,p)** разбивает графическое окно на m×n подокон, при этом m – число подокон по горизонтали, n – число подокон по вертикали, а p – номер подокна, в которое будет выводиться текущий график (подокна отсчитываются последовательно по строкам). Функции bode и nyquist строят характеристики АЧХ и ФЧХ (**bode**) и АФЧХ (**nyquist**) соответственно всех четырех соединений на одном графике, функция **step** строит переходные характеристики для последовательного, параллельного соединений и для соединения с жесткой обратной связью на одном графике, а функция **impulse** строит импульсные характеристики последовательного и параллельного соединений звеньев на одном графике, при этом характеристики первого соединения строятся красными точками, второго соединения - зелеными штриховыми линиями, третьего соединения – желтыми плюсами, четвертого соединения – сплошными синими линиями. Результат выполнения данной

программы представлен на рис. 4.

Рассмотрим пример выполнения некоторых заданий лабораторной работы 3.

Исследовать устойчивость замкнутой системы автоматического управления, контур прямой связи которой представляет собой последовательное соединений двух инерционных звеньев с передаточной функцией

$$W_i(p) = \frac{K_i}{1+T_i p},$$

К1 = 3, Т1 = 0.025,

К2 = 4, Т2 = 0,04

и одного интегрирующего звена с передаточной функцией

$$W_3(p) = \frac{K_3}{p}.$$

Контур прямой связи замыкается единичной отрицательной обратной связью, т.е. контур обратной связи представляет собой пропорциональное звено с передаточной функцией $K_{ос} = 1$.

Запишем текст программы на языке MATLAB:

```
i1=tf([3],[0.025 1])
i2=tf([4],[0.025 1])
i3=tf([5.4],[1 0])
i4=tf([1],[0 1])
A1=series(i1,i2)
A=series(A1,i3)
B=feedback(A,i4)
subplot(221)
bode(A,'r',B,'g--')
subplot(222)
pzmap(A,'r',B,'g--')
subplot(223)
margin(B)
subplot(224)
nyquist(A,'r',B,'g--')
```

Первые две строки программы задают два инерционных звена, а третья строка – интегрирующее звено (все звенья задаются в виде tf-моделей, т. е. в виде передаточных функций). Следующие три строки указывают вид соединения звеньев (две – последовательные соединения, третья – соединение с обратной связью). Функция **subplot(m,n,p)** разбивает графическое окно на m×n подокон, при этом m – число подокон по горизонтали, n –число

подокон по вертикали, а p – номер подокна, в которое будет выводиться текущий график (подокна отсчитываются последовательно по строкам). Функции **bode**, **nyquist** и **pzmap** строят на одном графике соответственно характеристики АЧХ и ФЧХ (**bode**), АФЧХ (**nyquist**) и указывает расположение полюсов и нулей (**pzmap**) как разомкнутой, так и замкнутой систем, при этом характеристики разомкнутой системы строятся сплошными красными линиями, а замкнутой - зелеными штриховыми линиями. Функция **margin** вычисляет запасы устойчивости по фазе и модулю, а также соответствующие частоты для замкнутой системы. Для разомкнутой системы запас по фазе не определен (так как она не является устойчивой), поэтому данная функция к ней не применима. Результат выполнения данной программы – рис. 5.

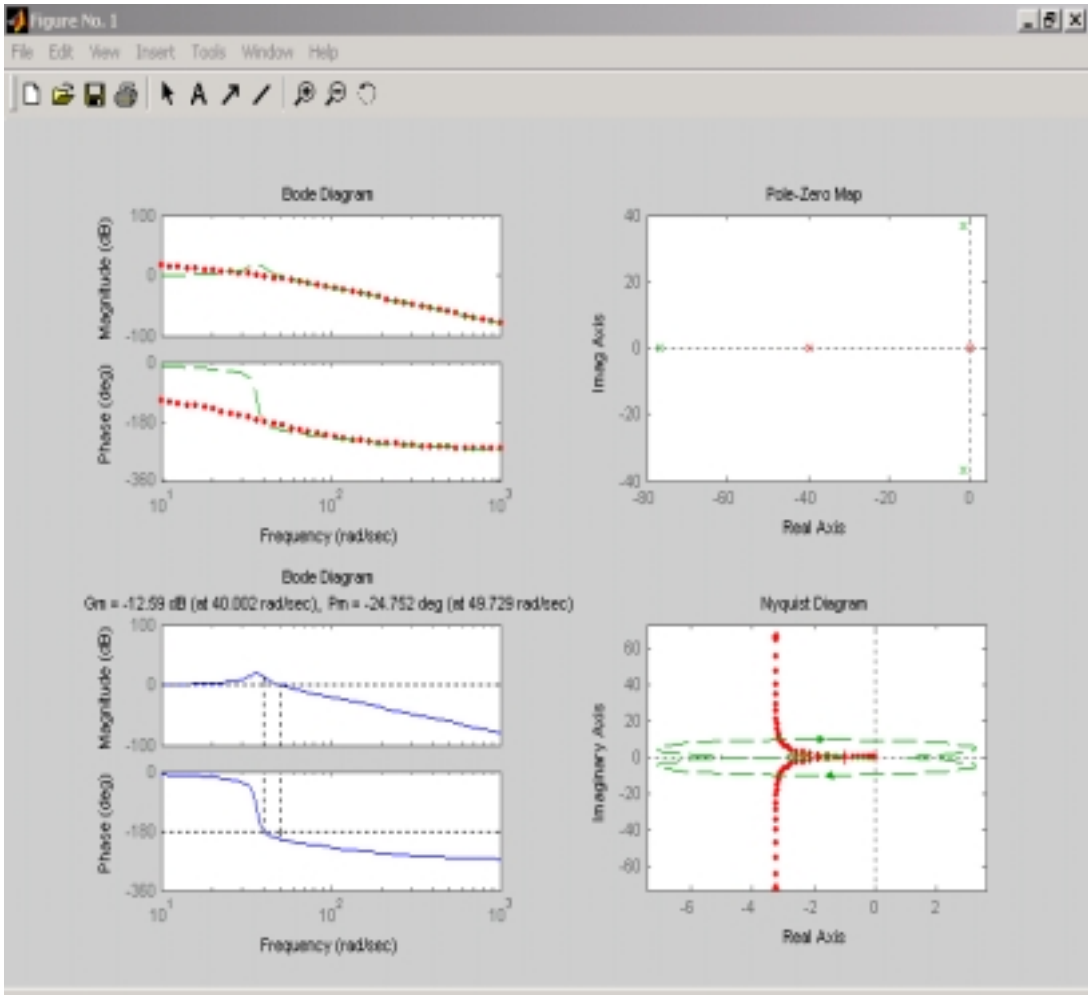


Рис. 5.

Выше были рассмотрены только три лабораторные работы из планируемых семи: 1) исследование характеристик типовых звеньев систем автоматического управления; 2) анализ соединений типовых

звеньев; 3) анализ устойчивости линейных систем автоматического управления.

В дальнейшем предполагается провести апробацию остальных четырех лабораторных работ комплекса, а также принять участие в постановке лабораторных работ по анализу и синтезу нелинейных систем автоматического управления.

УДК 519.6

MATLAB В КУРСЕ ЧИСЛЕННЫХ МЕТОДОВ И КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

Тарасевич Ю.Ю., Пономарева И.С.
Астраханский государственный педагогический университет, г. Астрахань
e-mail: tarasevich@astranetl.ru

При построении курса численных методов необходимо ответить на принципиальный вопрос: нужно ли студентов учить программировать численные методы или их нужно учить *применять* численные методы? Представьте, что студентов радиотехнического вуза до 5 курса учат собирать детекторные приемники, при этом студенты не в состоянии самостоятельно включить телевизор. Очевидна абсурдность этой картины, но обучение численным методам зачастую строится именно по такой схеме. За время обучения студенты успевают запрограммировать на самом примитивном уровне простейшие численные методы, при этом оказываются абсолютно беспомощными при решении реальных задач на применение численных методов. «Увы, времена меняются; <…> классические формулы почти абсолютно бесполезны. Они являются музейными экспонатами, хотя и прекрасными» [1]. Представляется целесообразным начинать обучение с применения численным методов и лишь после овладения каким либо из стандартных пакетов переходить к программированию численных методов. MATLAB представляет возможность решить обе задачи [2]. На первом этапе студенты учатся применять стандартные численные методы, заложенные в пакет, а на втором – программировать численные методы и создавать интерфейс. Типичная задача, предлагаемая студентам: «Зависимость скорости тела от времени заданы таблицей. Требуется определить путь, пройденный телом за определенный промежуток времени, максимальную скорость и максимальное ускорение тела, скорость тела в определенный момент времени». Задача включает в себя целый набор численных методов: интерполяцию, интегрирование, дифференцирование, оптимизацию.

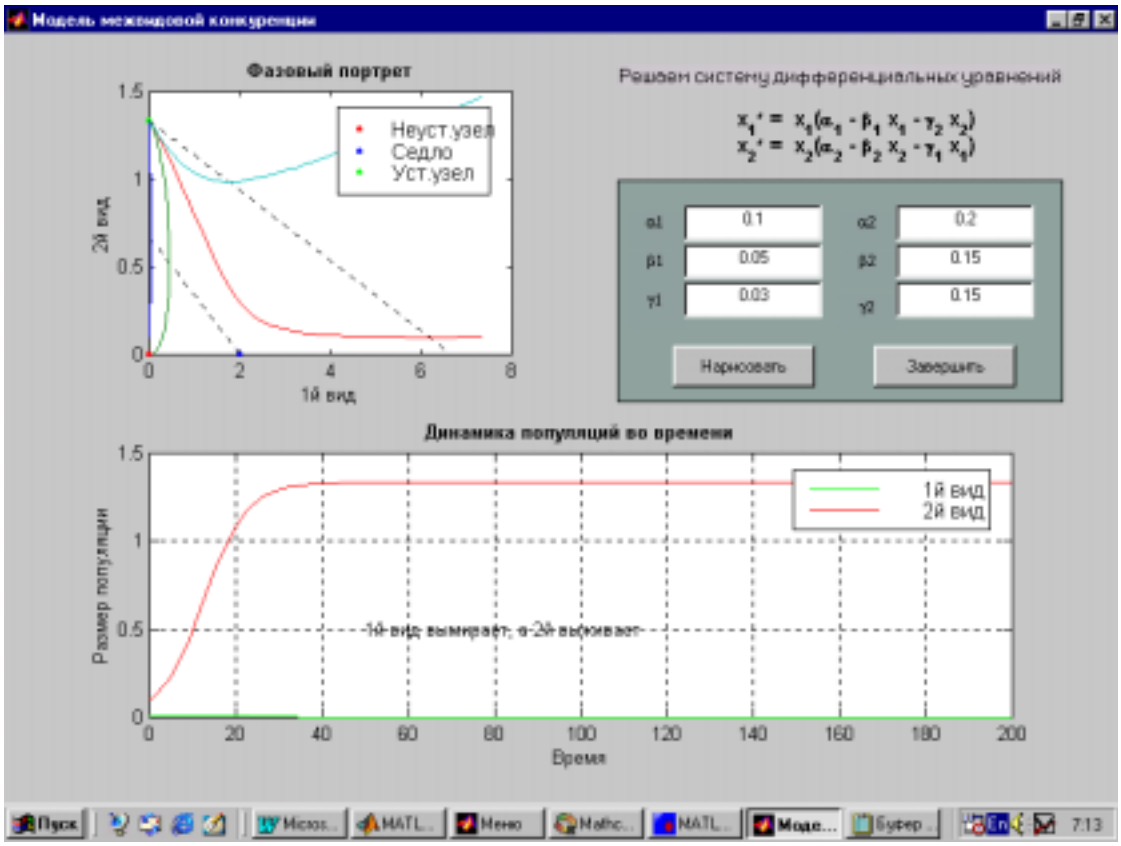


Рис. 1. Одна из лабораторных работ по динамике биологических популяций.

Курс математического и компьютерного моделирования представляет прекрасное поле для применения и закрепления навыков применения численных методов. С другой стороны, в связи с ограниченностью времени аудиторных занятий желательно на занятиях сделать упор на содержательной стороне модели, а техническую часть перенести на самостоятельную работу. При изучении курса представляется целесообразным сделать акцент на исследовании нелинейных моделей. Роберт Мэй заметил: «Не только в научных исследованиях, но и в повседневном мире экономики и политики всем нам будет лучше, если больше людей поймет, что простые нелинейные системы не обязательно проявляют простые динамические свойства» [3]. Изучение таких явлений как динамический хаос и процессы самоорганизации имеет большое мировоззренческое значение.

В разные годы при проведении практических занятий использовались пакеты Mathcad, Maple, MATLAB. На основании опыта преподавания дисциплины было подготовлено учебное пособие [4] и разработан комплекс лабораторных работ с использованием пакета MATLAB. Комплекс имеет удобный графический интерфейс (рис. 1, 2) и включает следующие разделы: дифференциальные модели (нелинейные колебания физического

маятника, динамика биологических популяций, колебательные реакции в химии, автоколебательные процессы), отображения (логистическое отображение, отображение Хенона и др.), геометрические модели. Набор лабораторных работ постоянно расширяется. Комплекс позволяет эффективно использовать время аудиторных занятий, делая упор на анализе результатов. Кроме того, студентам на занятиях предлагается самостоятельно создавать и исследовать имитационные модели с использованием расширения Simulink.

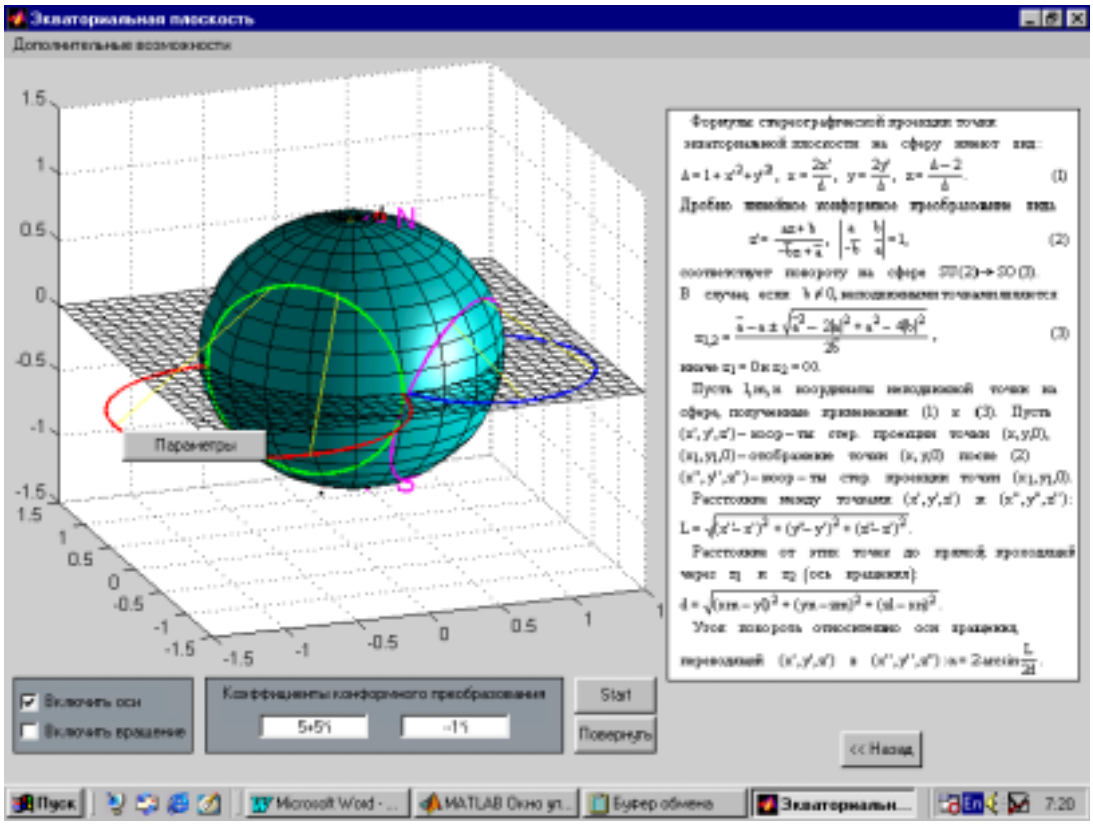


Рис. 2. Лабораторная работа по геометрическому моделированию.

При изучении курса студенты получают индивидуальные задания для самостоятельной работы. Ими составляются программы для таких задач как клеточные автоматы, геометрические и термические фазовые переходы, модели кинетического роста и некоторые другие. В этой части работы у них есть возможность продумать детали реализации модели и применить на практике полученные ранее знания в области программирования и численных методов.

Литература

1. *Press W. H. et al.* Numerical Recipes in C Cambridge University Press, 1986.
2. *Мэтьюз Дж. Г., Финк К. Д.* Численные методы. Использование MATLAB. Вильямс, 2001. 720 с.
3. *May R. M.* Simple mathematical models with very complicated dynamics // Nature, Vol. 261 June 10, 1976, pp. 459 - 467.
4. *Тарасевич Ю.Ю.* Математическое и компьютерное моделирование. Вводный курс: Учебное пособие. М.: Эдиториал УРСС, 2001. 144 с.

УДК 519.6

ИЗУЧЕНИЕ ТЕОРИИ ЛИНЕЙНЫХ ЭЛЕКТРИЧЕСКИХ ЦЕПЕЙ С ПОМОЩЬЮ СИСТЕМЫ MATLAB

Шмелёв В.Е.
Владимирский государственный университет, г. Владимир
e-mail: shmelioff_v_e@chat.ru

Современная теория линейных электрических цепей базируется на матричных методах их численного и символьного расчёта. Преимущество матричных методов заключается в их универсальности. Вычислительные алгоритмы, основанные на этих методах, позволяют проводить расчёты электрического состояния цепей практически неограниченной сложности.

На кафедре «Электротехника и электроэнергетика» Владимирского государственного университета разработан учебный вычислительный сценарий анализа линейной цепи произвольной сложности. Этот сценарий поддерживает три «экономичных» матричных метода: метод узловых потенциалов, метод напряжений ветвей дерева, метод контурных токов. Эти три метода основаны на одних и тех же матричных соотношениях с разной системой обозначений.

В качестве примера рассмотрим электрическую цепь, ориентированный граф которой изображён на рис.1.

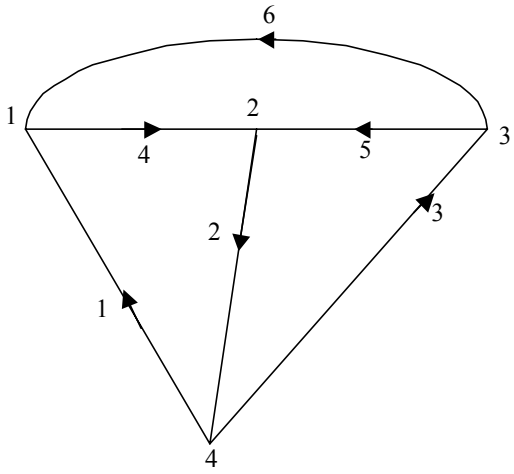


Рис. 1. Граф электрической цепи.

Пусть графу, изображённому на рис. 1, соответствует следующая схема (см. рис. 2).

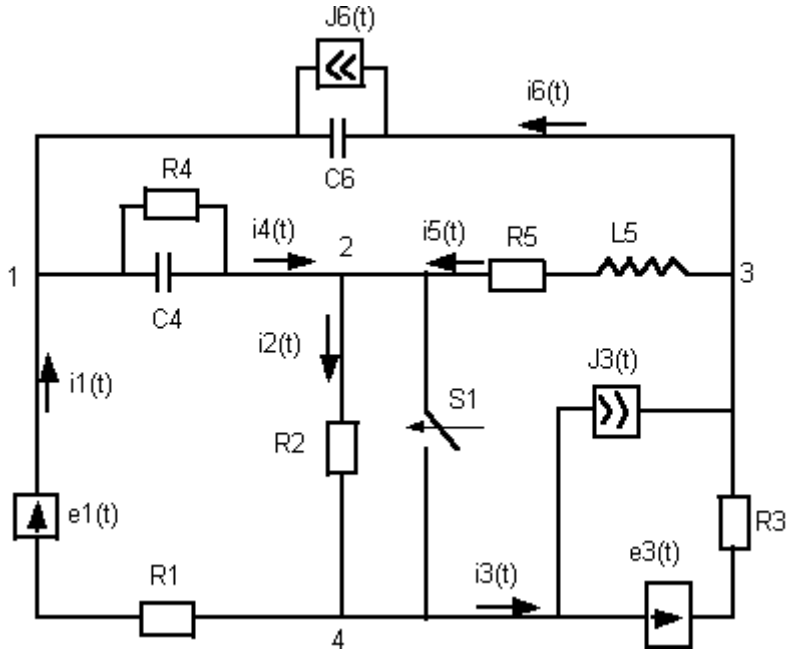


Рис. 2. Схема моделируемой электрической цепи.

Пусть по условию задачи дано:
R1=R2=R3=5кОм; R4=5кОм; R5=0.2кОм; C4=C6=1мкФ; L5=1Гн;
e1(t)=(100*sin(ωt)+20*cos(3ωt)–4*sin(5ωt))В;
J3(t)=-20мА; e3(t)=(50*cos(ωt)+10*cos(3ωt)+2*cos(5ωt))В;
J6(t)=(10+60*sin(ωt)+20*sin(3ωt)+12*sin(5ωt))мА;
Циклическая частота основной гармоники источников ω=1рад/мс. До коммутации (при разомкнутом ключе S1) в цепи имел место установившийся процесс. В момент времени t=0 происходит мгновенная коммутация: ключ S1 замыкается.

Требуется:

Определить выражения (в числовой форме) для мгновенных значений токов всех ветвей до и после коммутации. Построить графики мгновенных значений токов ветвей до и после коммутации.

Сначала определим комплексные амплитуды всех гармоник токов всех ветвей и напряжений на зажимах конденсаторов до коммутации. Для этого составим и введём топологическую матрицу узловых соединений:

$$A=\begin{bmatrix} -1 & 0 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 1 & 1 \end{bmatrix}.$$

Эта матрица однозначно соответствует графу и схеме, изображённым на рис. 1 и 2, и не зависит от частоты (а, значит, и от номера гармоники). Остальные параметры цепи (адмитансы и комплексные амплитуды источников ветвей) зависят от номера гармоники. Для ускорения расчётов матрицы параметров цепи введём сразу для всех гармоник:

```
omega=1;
PV_=diag([0.2 0.2 0.2 0.2 5 0]);
PV1=diag([0.2 0.2 0.2 (0.2+j*omega) …
                                    (0.2+j*omega)^-1 j*omega]);
PV3=diag([0.2 0.2 0.2 (0.2+3j*omega) …
                                    (0.2+3j*omega)^-1 3j*omega]);
PV5=diag([0.2 0.2 0.2 (0.2+5j*omega) …
                                    (0.2+5j*omega)^-1 5j*omega]);
SV_=[0 0 20 0 0 -10].';
SV1=[0 0 0 0 0 -60].';
SV3=[0 0 0 0 0 -20].';
SV5=[0 0 0 0 0 -12].';
CV_=[0 0 0 0 0 0].';
CV1=[100 0 50i 0 0 0].';
CV3=[20i 0 10i 0 0 0].';
CV5=[-4 0 2i 0 0 0].';
```

Рассчитаем постоянную составляющую токов и напряжений ветвей:

```
серуе
<метод узловых потенциалов>
TM=A
PV=PV_
SV=SV_
CV=CV_
<Расчёт>
<Токи ветвей>
Токи ветвей   XC=

   -0.15625
   -9.6875
```

```
    -9.5313
     9.8437
   -19.531
    10
<Конец просмотра результатов расчёта>
XS([4,6]).'
ans =
     49.219     -53.125
XC_=XC; XS_=XS;
```

Из приведённой последовательности команд и сообщений в окне MATLAB следует, что постоянные составляющие токов ветвей до коммутации равны: $I_{1(0)}$=-0.15625мА; $I_{2(0)}$=-9.6875мА; $I_{3(0)}$=-9.5313мА; $I_{4(0)}$=9.8437мА; $I_{5(0)}$=-19.531мА; $I_{6(0)}$=10мА. Постоянные составляющие напряжений на конденсаторах: $U_{4(0)}$=-49.219В; $U_{6(0)}$=-53.125В.

Теперь рассчитаем первую гармонику токов и напряжений ветвей:

```
серуе
<метод узловых потенциалов>
PV=PV1
SV=SV1
CV=CV1
<Расчёт>
<Токи ветвей>
     12.49 +    1.7086i
     10.648 +   9.2257i
      -1.8422 +7.517i
     51.533 - 26.628i
    -40.884 +35.854i
     39.042 - 28.337i
<Конец просмотра результатов расчёта>
XS([4,6]).'
ans =
   -15.694 - 54.671i        -28.337 + 20.958i
XC1=XC; XS1=XS;
```

Из приведённой последовательности команд и сообщений в окне MATLAB следует, что комплексные амплитуды первых гармоник токов ветвей до коммутации равны: $\underline{I}_{1(1)}$=(12.49+1.7086i)мА; $\underline{I}_{2(1)}$=(10.648+9.2257i)мА; $\underline{I}_{3(1)}$=(-1.8422+7.517i) мА; $\underline{I}_{4(1)}$=(51.533–26.628i)мА; $\underline{I}_{5(1)}$=(–40.884+35.854i) мА;

$\underline{I}_{6(1)}$=(39.042–28.337i)мА. Комплексные амплитуды первых гармоник напряжений на конденсаторах: $\underline{U}_{4(1)}$=(–15.694–54.671i)В; $\underline{U}_{6(1)}$=(-28.337+20.958i)В.

Расчёт третьей гармоники токов и напряжений ветвей:

```
серуе
<метод узловых потенциалов>
PV=PV3
SV=SV3
CV=CV3
<Расчёт>
<Токи ветвей>
  -0.049943 +     2.4419i
  -0.012609 +     1.3635i
   0.037334 -     1.0784i
    -2.8554 +     1.1328i
     2.8428 +    0.23073i
    -2.8054 -     1.3091i
<Конец просмотра результатов расчёта>
XS([4,6]).'
ans =
   0.31276 + 0.97264i      -0.43638 + 7.6018i
XC3=XC; XS3=XS;
```

Из приведённой последовательности команд и сообщений в окне MATLAB следует, что комплексные амплитуды третьих гармоник токов ветвей до коммутации равны: $\underline{I}_{1(3)}$=(–0.049943+2.4419i)мА; $\underline{I}_{2(3)}$=(–0.012609+1.3635i) мА; $\underline{I}_{3(3)}$=(0.037334–1.0784i)мА; $\underline{I}_{4(3)}$=(–2.8554+1.1328i)мА; $\underline{I}_{5(3)}$=(2.8428+0.23073i)мА; $\underline{I}_{6(3)}$=(–2.8054–1.3091i)мА. Комплексные амплитуды первых гармоник напряжений на конденсаторах: $\underline{U}_{4(3)}$=(0.31276+0.97264i)В; $\underline{U}_{6(3)}$=(–0.43638+7.6018i)В.

Расчёт пятой гармоники токов и напряжений ветвей:

```
серуе
<метод узловых потенциалов>
PV=PV5
SV=SV5
CV=CV5
<Расчёт>
<Токи ветвей>
  -0.53318 +   0.019617i
  -0.26238 -   0.050731i
```

```
     0.2708 -   0.070347i
    -0.78229 -   0.079952i
     0.51991 +   0.029222i
    -0.24911 -   0.099569i
<Конец просмотра результатов расчёта>
XS([4,6]).'
ans =
  -0.022213 + 0.15557i        -0.019914 + 2.4498i
XC5=XC; XS5=XS;
```

Теперь по результатам расчёта гармонических составляющих запишем выражения для мгновенных значений токов ветвей до коммутации:

$i_1(t)$=(-0.15625+12.49*sin(ω*t*)+1.7086*cos(ω*t*)–
0.049943*sin(3ω*t*)+2.4419* cos(3ω*t*)–
0.53318*sin(5ω*t*)+0.019617*cos(5ω*t*))мА;

$i_2(t)$=(-9.6875+10.648*sin(ω*t*)+9.2257*cos(ω*t*)–
0.012609*sin(3ω*t*)+1.3635* cos(3ω*t*)–0.26238*sin(5ω*t*)–
0.050731*cos(5ω*t*))мА;

$i_3(t)$=(-9.5313–1.8422*sin(ω*t*)+7.517*cos(ω*t*)+0.037334*sin(3ω*t*)–
1.0784* cos(3ω*t*)+0.2708*sin(5ω*t*)–0.070347*cos(5ω*t*))мА;

$i_4(t)$=(9.8437+51.533*sin(ω*t*)–26.628*cos(ω*t*)–
2.8554*sin(3ω*t*)+1.1328* cos(3ω*t*)–0.78229*sin(5ω*t*)–
0.079952*cos(5ω*t*))мА;

$i_5(t)$=(-19.531–
40.884*sin(ω*t*)+35.854*cos(ω*t*)+2.8428*sin(3ω*t*)+0.23073*
cos(3ω*t*)+0.51991*sin(5ω*t*)+0.029222*cos(5ω*t*))мА;

$i_6(t)$=(10+39.042*sin(ω*t*)–28.337*cos(ω*t*)–
0.43638*sin(3ω*t*)+7.6018*cos(3ω*t*)–0.24911*sin(5ω*t*)–
0.099569*cos(5ω*t*))мА.

Начальные условия при коммутации:

$i_5(0)=I_{5(0)}+\text{imag}(\underline{I}_{5(1)})+\text{imag}(\underline{I}_{5(3)})+\text{imag}(\underline{I}_{5(5)})$;

$u_4(0)=U_{4(0)}+\text{imag}(\underline{U}_{4(1)})+\text{imag}(\underline{U}_{4(3)})+\text{imag}(\underline{U}_{4(5)})$;

$u_6(0)=U_{6(0)}+\text{imag}(\underline{U}_{6(1)})+\text{imag}(\underline{U}_{6(3)})+\text{imag}(\underline{U}_{6(5)})$

Последовательность команд для их вычисления:

```
xc50=XC_(5)+imag(XC1(5))+imag(XC3(5))+imag(XC5(5))
xc50 =
    16.583
xs40=XS_(4)+imag(XS1(4))+imag(XS3(4))+imag(XS5(4))
xs40 =
    -4.3244
```

```
xs60=XS_(6)+imag(XS1(6))+imag(XS3(6))+imag(XS5(6))
xs60 =
    -22.116
```

Итак, начальные условия: $i_5(0)$=16,583мА; $u_4(0)$=-4.3244В; $u_6(0)$=-22.116В.

Теперь можно приступить к расчёту токов ветвей после коммутации операторным методом. Для этого нужно подключить к работе вычислительного сценария серуе пакет Symbolic Match Toolbox путём описания двух символьных переменных:

```
syms t real
syms s
```

Здесь t – время (мс), s – «комплексная частота» ($мс^{-1}$). Нужно также ввести в расчёт операторные сопротивления или проводимости ветвей, операторные изображения источников (в том числе и начальных условий). Следует обратить внимание на то, что в результате коммутации ветвь №2 закорачивается, значит, её сопротивление становится равным нулю. Это приводит к тому, что для анализа цепи теперь нельзя применять метод узловых потенциалов без изменения топологической матрицы *A*. Чтобы перейти к методу контурных токов, нужно либо составить вручную, либо вычислить в MATLABe с помощью разработанной автором функции getqbm топологическую матрицу *B*:

```
[Q,TM]=getqbm(TM,[1,2,3],[4,5,6])
Q =
   1   0   0  -1   0   1
   0   1   0  -1  -1   0
   0   0   1   0  -1  -1
TM =
   1   1   0   1   0   0
   0   1   1   0   1   0
  -1   0   1   0   0   1
```

Теперь PV – матрица операторных сопротивлений ветвей:

```
PV=diag([5 0 5 (0.2+s)^-1 (0.2+s) 1/s])
```

Запустим вычислительный сценарий серуе и в ходе его выполнения введём операторные изображения источников ветвей:

```
серуе
<Метод контурных токов>
<ЭДС ветвей>
SV=[laplace(100*sin(t)+20*cos(3*t)–4*sin(5*t)) 0
laplace(50*cos(t)+10*cos(3*t)+2*cos(5*t)) 0 xc50 0].’
<источники тока ветвей>
CV=[0 0 –20/s xs40 0 xs60-
laplace(10+60*sin(t)+20*sin(3*t)+12*sin(5*t))].’
<Конец просмотра результатов расчёта>
XS=simple(XS)
XS =
[
1/562949953421312/s*(5905166695226235521 7*s^5+13228865055269
5046903*s^3+346916043247006323618*s^4-
239706219447159324162*s+578975365796011699830*s^2+10905637
90658117430*s^8+34233597375853372 5*s^9-
329325722751467520 00+441448834786479156 18*s^6+105774421862
73310837*s^7)/(1+5*s)/(25*s^2+20*s+52)/(s^2+1)/(s^2+9)/(s^2+25)]
[
1/562949953421312/s*(8995644324684478185 02*s^5+1840850372045
008904414*s^3+935245390427064469843*s^4+482047810165756240
398*s+128263379810669475438 5*s^2+320865290443616190 5*s^8+2
994392384906048500*s^9+119578416853282702963*s^6+962644420
35044352000+107967224291076941866*s^7)/(1+5*s)/(25*s^2+20*s+
52)/(s^2+1)/(s^2+9)/(s^2+25)]
[
1/562949953421312*(530411282229502955*s^8+31753556630970830
4*s^7+194144493076987845 45*s^6+112038168133872005 60*s^5+16
586178974055965254 5*s^4+844935114878996987 36*s^3+324813642
000882831755*s^2+757689580619600445 60*s+129197014310191104
000)/s/(25*s^2+20*s+52)/(s^2+1)/(s^2+9)/(s^2+25)]
[
1/281474976710656*(182749945266732375*s^8+37701841670677232
3*s^7+627633974425689366 9*s^6+150381797822849854 81*s^5+453
911843846870013 49*s^4+131407878633730112297*s^3+6927467960
5381704055*s^2+199432806426674205699*s+16466286137573376000
)/s/(25*s^2+20*s+52)/(s^2+1)/(s^2+9)/(s^2+25)]
[
5/562949953421312/s*(831152458114015668 10*s^5+17705756374474
6327854*s^3-939231529359221311 65*s^4-162961328126651862 00*s-
170048725074162142143*s^2-
185406230633005215*s^8+233378586447744950*s^9+126663739519
```

```
79520000-
867121209161618783 7*s^6+8889958003018892106*s^7)/(1+5*s)/(25*
s^2+20*s+52)/(s^2+1)/(s^2+9)/(s^2+25)]
[
1/562949953421312/s*(4249365364591776292 35*s^5+8232739027685
82218241*s^3+1057945111859668802050*s^4+803234693676241495
560*s+1553902057681493765270*s^2+3045120266943070550*s^8+1
485163478908790025*s^9+1187895938327157265 30*s^6+658651445
50293504000+529399920897091704 99*s^7)/(1+5*s)/(25*s^2+20*s+5
2)/(s^2+1)/(s^2+9)/(s^2+25)]

for ii=1:6, xs(ii)=ilaplace(XS(ii)); end
xs
xs =
[         -5-
735317161627430494201 71/126832922869296906895 36*exp(-1/5*t)-
102605295427800747759549009553 21/353280474813293032163332063
2320*exp(-
2/5*t)*3^(1/2)*sin(4/5*3^(1/2)*t)+3612054621523837951654166251 1
9/110400148379154072551041269760*exp(-
2/5*t)*cos(4/5*3^(1/2)*t)+127700/14677*cos(t)+330010/14677*sin(t)
+14178604/3788777*cos(3*t)-1000550/3788777*sin(3*t)-
5540920/105896977*cos(5*t)-417230714/529484885*sin(5*t)]
[ 190/13-
735317161627430494201 71/126832922869296906895 36*exp(-1/5*t)-
183391486523945139297849786147291/22963230862864047090616584
110080*exp(-
2/5*t)*3^(1/2)*sin(4/5*3^(1/2)*t)+5007890192339791435768760218 7
663/459264617257280941812331682 2016*exp(-
2/5*t)*cos(4/5*3^(1/2)*t)+280450/14677*cos(t)+454030/14677*sin(t)
+14884854/3788777*cos(3*t)-2121510/3788777*sin(3*t)-
21807530/105896977*cos(5*t)-406732694/529484885*sin(5*t)]
[                                                          255/13-
476318548962753686751603795664 1/937274729096491717984350371
840*exp(-
2/5*t)*3^(1/2)*sin(4/5*3^(1/2)*t)+3576811703863137599878190875 7
87/468637364548245858992175185920*exp(-
2/5*t)*cos(4/5*3^(1/2)*t)+11750/1129*cos(t)+9540/1129*sin(t)+6250/
33529*cos(3*t)-9920/33529*sin(3*t)-
51970/338329*cos(5*t)+6708/338329*sin(5*t)]
[
5+7341585635250010409228250420 39/720980560843455167680269 51
680*exp(-
```

**2/5*t)*3^(1/2)*sin(4/5*3^(1/2)*t)+13743360132881344771097157496
7/360490280421727583840134758 4*exp(-2/5*t)*cos(4/5*3^(1/2)*t)-
23850/1129*cos(t)+46310/1129*sin(t)+141458/33529*cos(3*t)-
120770/33529*sin(3*t)-83850/338329*cos(5*t)-
2233138/1691645*sin(5*t)]**
**[125/13-
7353171616274304942017 1/126832922869296906895 36*exp(-1/5*t)-
16688839560266318833270782481468 5/91852923451456188362466336
44032*exp(-2/5*t)*3^(1/2)*sin(4/5*3^(1/2)*t)-
12501150616951041802609018432029 5/45926461725728094181233168
22016*exp(-2/5*t)*cos(4/5*3^(1/2)*t)+590500/14677*cos(t)-
148000/14677*sin(t)-
1099900/3788777*cos(3*t)+11525500/3788777*sin(3*t)+4437520/105
896977*cos(5*t)+58447900/105896977*sin(5*t)]**
**[
10+7353171616274304942017 1/126832922869296906895 36*exp(-
1/5*t)+36120546215238379516541662511 9/27600037094788518137760
317440*exp(-
2/5*t)*3^(1/2)*sin(4/5*3^(1/2)*t)+30781588628340224327864702865
963/88320118703323258040833015808 0*exp(-
2/5*t)*cos(4/5*3^(1/2)*t)-
437750/14677*cos(t)+272020/14677*sin(t)+1806150/3788777*cos(3*t)
-12646460/3788777*sin(3*t)-20704130/105896977*cos(5*t)-
56348296/105896977*sin(5*t)]**

Здесь XS – матрица-столбец операторных изображений токов ветвей; xs – матрица-столбец выражений для мгновенных значений этих токов после коммутации.

По условию задачи нужно построить графики мгновенных значений токов ветвей до и после коммутации. Для удобства мгновенные токи ветвей до коммутации сведём в матрицу-столбец:

**toki_do=sym(XC_,'d')+sym(real(XC1),'d')*sin(t)+sym(imag(X
C1),'d')*cos(t)+sym(real(XC3),'d')*sin(3*t)+sym(imag(XC3),'d')*co
s(3*t)+ sym(real(XC5),'d')*sin(5*t)+sym(imag(XC5),'d')*cos(5*t)**
toki_do=
**[-
.1562500000000092+12.49047358433628*sin(t)+1.708616994614061*
cos(t)-.4994257501370082e-1*sin(3*t)+2.441944576805298*cos(3*t)-
.5331800410990823*sin(5*t)+.1961683471466901e-1*cos(5*t)]**
**[-
9.687499999999984+10.64830775053834*sin(t)+9.225662245727817*
cos(t)-.1260889373419641e-1*sin(3*t)+1.363526722005960*cos(3*t)-**

```
.2623773142016971*sin(5*t)-.5073058699808304e-1*cos(5*t)]
[ -9.531250000000016-
1.842165833797925*sin(t)+7.517045251113760*cos(t)+.37333681279
50803e-1*sin(3*t)-
1.078417854799339*cos(3*t)+.2708027268973874*sin(5*t)-
.7034742171275044e-1*cos(5*t)]
[    9.843749999999993+51.53261486683477*sin(t)-
26.62818591471495*cos(t)-
2.855379049083248*sin(3*t)+1.132800732407202*cos(3*t)-
.7822864517845715*sin(5*t)-.7995236519710094e-1*cos(5*t)]
[   -19.53125000000000-
40.88430711629643*sin(t)+35.85384816044278*cos(t)+2.8427701553
49045*sin(3*t)+.2307259895987890*cos(3*t)+.5199091375828733*si
n(5*t)+.2922177819900525e-1*cos(5*t)]
[                10.+39.04214128249850*sin(t)-
28.33680290932899*cos(t)-2.805436474069538*sin(3*t)-
1.309143844398133*cos(3*t)-.2491064106854868*sin(5*t)-
.9956919991174318e-1*cos(5*t)]
```

Команды визуализации рассчитанных токов:

```
ezplot(toki_do(1),[-2*pi,0])
hold on
ezplot(xs(1),[0,20])
axis auto
```

График тока $i_1(t)$ показан на рис.3. Остальные токи визуализируются аналогичной последовательностью команд. Их графики показаны на рис.4–8.

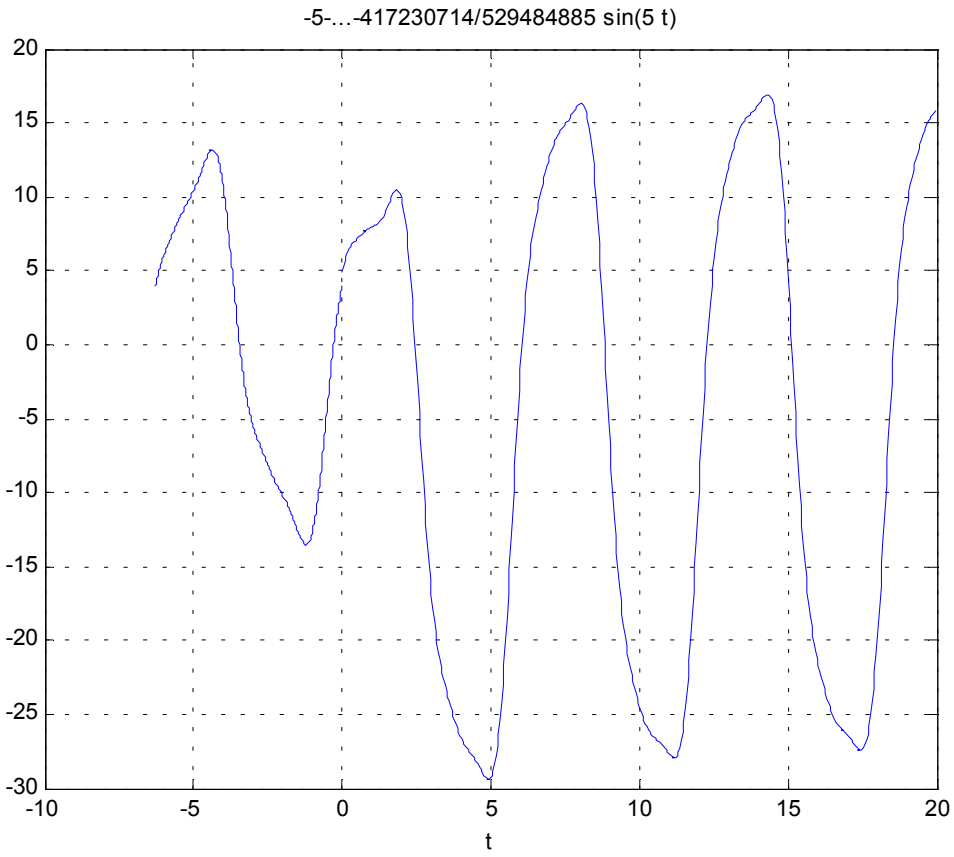


Рис. 3. График тока $i_1(t)$.

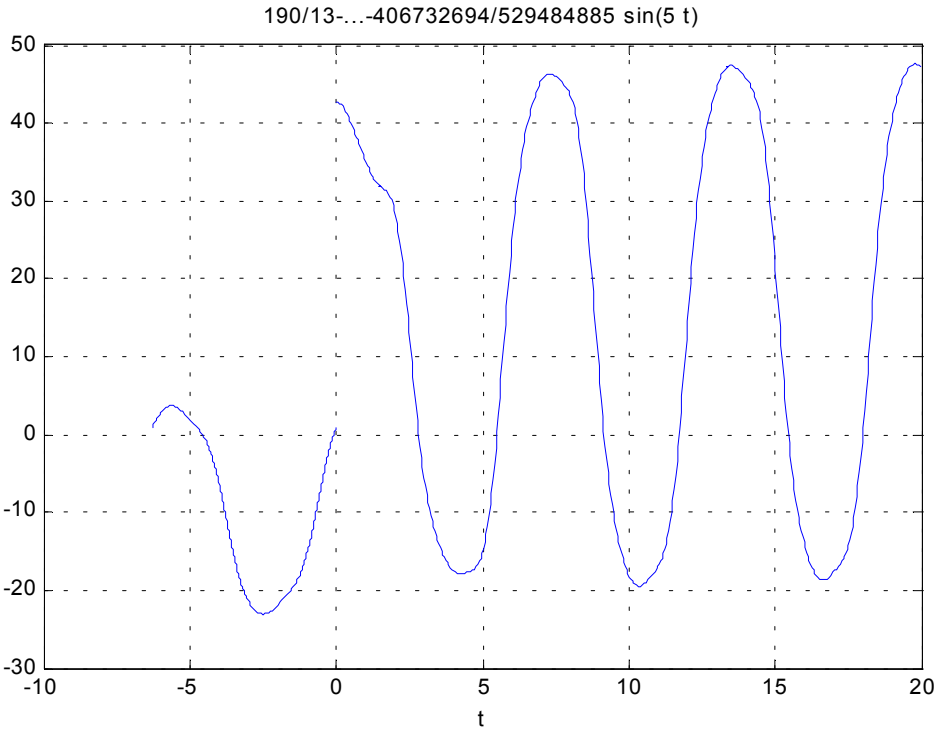


Рис. 4. График тока $i_2(t)$.

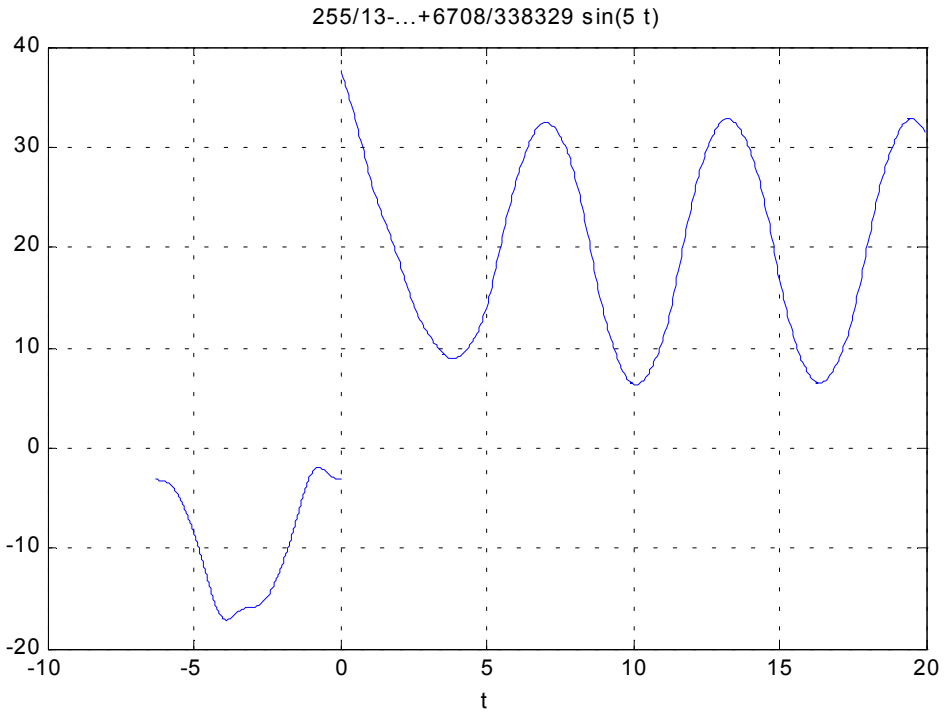


Рис. 5. График тока $i_3(t)$.

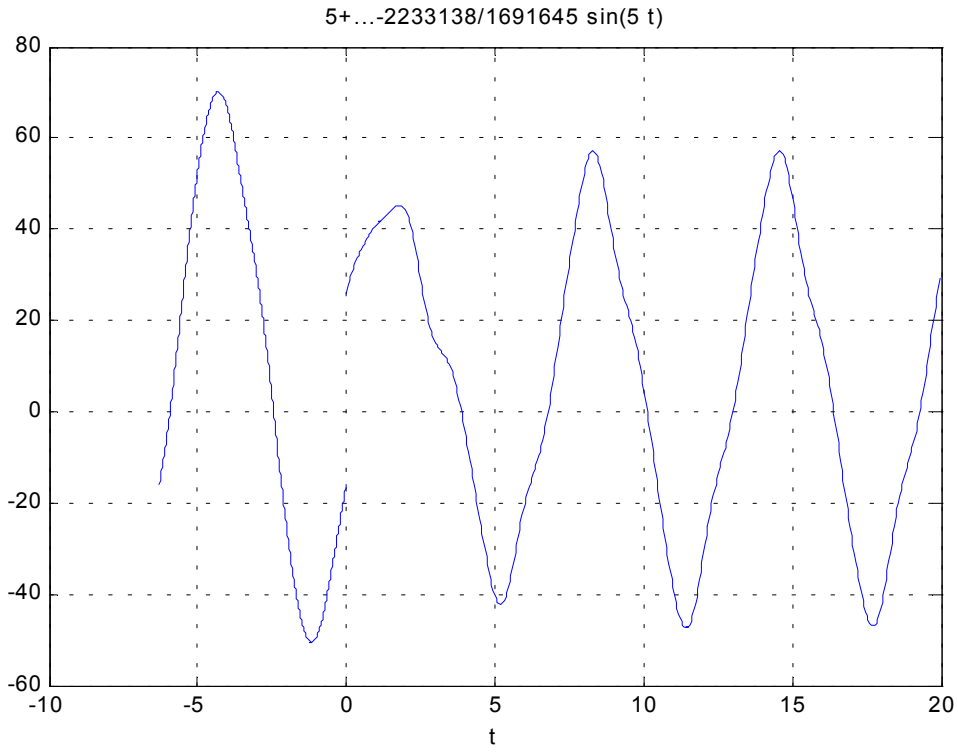


Рис. 6. График тока $i_4(t)$.

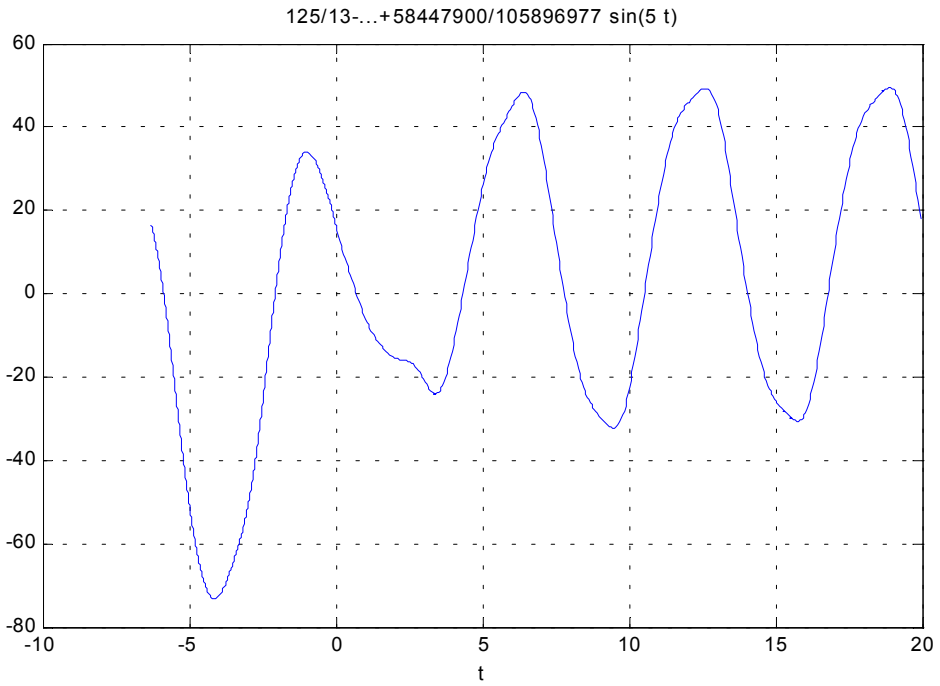


Рис. 7. График тока $i_5(t)$.

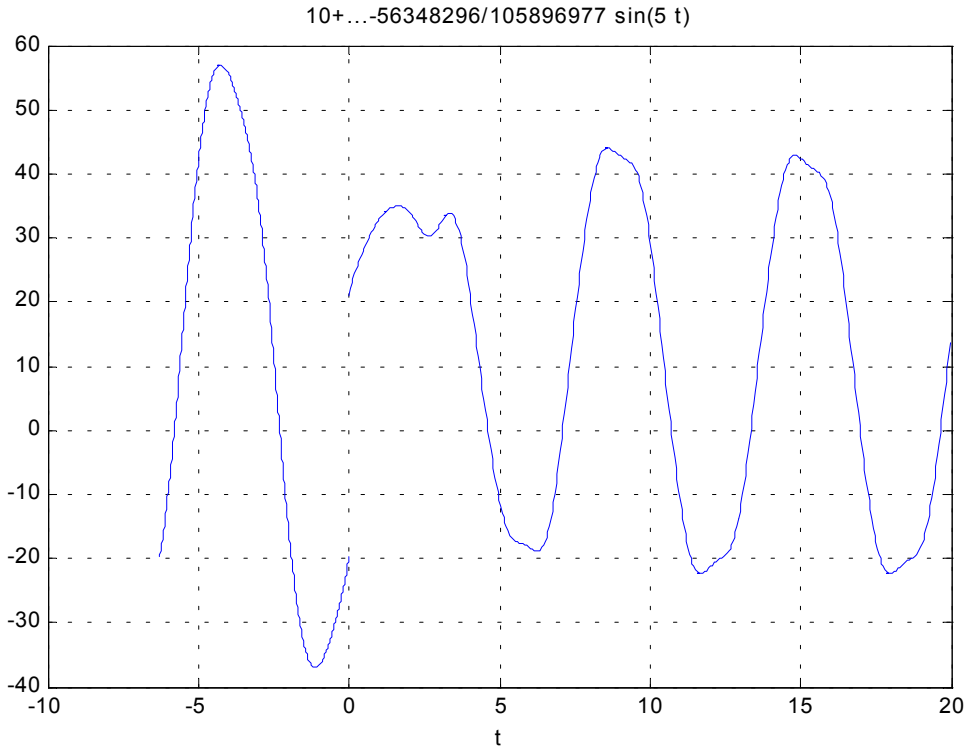


Рис. 8. График тока $i_6(t)$.

Учебные задания, даваемые студентам, гораздо проще, чем рассмотренная задача. Данная задача показывает богатые возможности вычислительного сценария серуе: анализ цепей постоянного тока, анализ синусоидальных режимов, анализ

периодических несинусоидальных режимов при разложении их в ряд Фурье, анализ переходных процессов операторным методом при замене численных арифметических операций символьными.

Студенты используют этот вычислительный сценарий при выполнении расчётно-графических работ и расчётной части лабораторных работ по теоретическим основам электротехники (теория линейных электрических цепей). Самостоятельной частью работы студентов является выполнение следующих операций: 1) составление и ввод топологической матрицы, матриц источников и импедансных (или адмитансных) параметров ветвей; 2) ручное составление узловых либо контурных уравнений или уравнений с напряжениями ветвей дерева; 3) сравнение этих уравнений с уравнениями, полученными машиной; 4) эквивалентное прямое и обратное преобразование ветвей с идеальными источниками ЭДС и тока; 5) расчёт средствами MATLAB баланса мощностей.

Названные выше «экономичные» методы анализа линейных электрических цепей основаны на следующих матричных соотношениях:

$$[Y^{(у)}][\varphi^{(у)}] = [J^{(у)}]; \quad [Y^{(с)}][U^{(д)}] = [J^{(с)}]; \quad [Z^{(к)}][I^{(к)}] = [E^{(к)}], \qquad (1)$$

где $[Y^{(у)}]=[A][Y^{(в)}][A].'$ – матрица узловых адмитансов; $[\varphi^{(у)}]$ – матрица узловых потенциалов; $[J^{(у)}]=[A]([J^{(в)}]-[Y^{(в)}][E^{(в)}])$ – матрица узловых источников тока; $[Y^{(с)}]=[Q][Y^{(в)}][Q].'$ – матрица адмитансов главных сечений; $[U^{(д)}]$ – матрица напряжений ветвей дерева; $[J^{(с)}]=[Q]([J^{(в)}]-[Y^{(в)}][E^{(в)}])$ – матрица источников тока главных сечений; $[Z^{(к)}]=[B][Z^{(в)}][B].'$ – матрица контурных импедансов; $[I^{(к)}]$ – матрица токов ветвей связи (контурных токов); $[E^{(к)}]=[B]([E^{(в)}]-[Z^{(в)}][J^{(в)}])$ – матрица контурных ЭДС. $[A]$, $[Q]$, $[B]$ – топологические матрицы: $[A]$ – матрица узловых соединений, $[Q]$ – матрица главных сечений, $[B]$ – матрица главных контуров. $[Y^{(в)}]$ – матрица адмитансов ветвей; $[Z^{(в)}]$ – матрица импедансов ветвей, причём $[Y^{(в)}]=[Z^{(в)}]^{-1}$. $[E^{(в)}]$ и $[J^{(в)}]$ – матрицы источников ветвей.

Результатом решения одного из уравнений (1) является столбец узловых потенциалов или столбец напряжений ветвей дерева или столбец контурных токов. Остальные параметры электрического состояния цепи определяются следующими матричными соотношениями:

$$[U^{(в)}] = [A].'[\varphi^{(у)}] = [Q].'[U^{(д)}]; \qquad [I^{(в)}] = [B].'[I^{(к)}];$$

$$[I^{(в)}] = [Y^{(в)}]([U^{(в)}]+[E^{(в)}])-[J^{(в)}]; \; [U^{(в)}] = [Z^{(в)}]([I^{(в)}]+[J^{(в)}])-[E^{(в)}];$$

$$[U^{(п)}] = [U^{(в)}]+[E^{(в)}]; \qquad [I^{(п)}] = [I^{(в)}]+[J^{(в)}],$$

где $[U^{(в)}]$ – столбец напряжений ветвей; $[I^{(в)}]$ – столбец токов ветвей; $[U^{(п)}]$ – столбец напряжений пассивных участков ветвей; $[I^{(п)}]$ – столбец токов пассивных участков ветвей. В общем случае

эти столбцы могут содержать комплексные амплитуды или комплексные действующие значения токов или напряжений. В последнем случае комплексные мощности определяются следующим образом:

$$[S^{(и)}] = \mathrm{conj}([I^{(п)}]).*[E^{(в)}]+\mathrm{conj}([J^{(в)}]).*[U^{(в)}];$$
$$[S^{(п)}] = \mathrm{conj}([I^{(п)}]).*[U^{(п)}];$$
$$[S^{(в)}] = \mathrm{conj}([I^{(в)}]).*[U^{(в)}],$$

где $[S^{(и)}]$ – столбец комплексных мощностей источников ветвей (генерируемые мощности); $[S^{(п)}]$ – столбец комплексных мощностей пассивных участков ветвей (приёмников); $[S^{(в)}]$ – столбец комплексных мощностей ветвей (здесь имеются в виду потребляемые мощности). В процессе составления баланса мощностей нетрудно убедиться, что

$$\mathrm{sum}([S^{(и)}])=\mathrm{sum}([S^{(п)}]), \qquad \mathrm{sum}([S^{(в)}])=0$$

Представленные выше матричные соотношения реализованы в вычислительном сценарии **серуе**.

Известно, что основным свойством линейных электрических цепей является принцип наложения (суперпозиции): реакция цепи на несколько приложенных одновременно возмущений равна сумме реакций на каждое возмущение в отдельности. Здесь под возмущением понимается действие источника, а под реакцией – токи и напряжения всех участков цепи. Матричная форма принципа наложения имеет вид:

$$[U^{(в)}] = [Z][J^{(в)}]+[K^{(u)}][E^{(в)}]; \qquad [I^{(в)}] = [Y][E^{(в)}]+[K^{(i)}][J^{(в)}],$$

где $[Z]$ – матрица входных и взаимных сопротивлений ветвей; $[K^{(u)}]$ – матрица коэффициентов передачи напряжения (коэффициентов распределения источников ЭДС); $[Y]$ – матрица входных и взаимных проводимостей ветвей; $[K^{(i)}]$ – матрица коэффициентов передачи тока (коэффициентов распределения источников тока). Матрицы $[Z]$, $[K^{(u)}]$, $[Y]$, $[K^{(i)}]$ называются матрицами коэффициентов передачи электрических сигналов. Они содержат в себе информацию обо всех возможных состояниях цепи.

В вычислительном сценарии **серуе** коэффициенты передачи электрических сигналов вычисляются следующим образом: по методу узловых потенциалов или напряжений ветвей дерева:

$$[Z] = [A].'[Y^{(у)}]^{-1}[A] = [Q].'[Y^{(с)}]^{-1}[Q]; [K^{(u)}] = -[Z][Y^{(в)}];$$
$$[K^{(i)}] = [Y^{(в)}][Z]-[1^{(в)}]; \qquad [Y] = [Y^{(в)}][K^{(u)}]+[Y^{(в)}];$$

по методу контурных токов:

$$[Y] = [B].'[Z^{(к)}]^{-1}[B]; \qquad [K^{(i)}] = -[Y][Z^{(в)}];$$
$$[K^{(u)}] = [Z^{(в)}][Y]-[1^{(в)}]; \qquad [Z] = [Z^{(в)}][K^{(i)}]+[Z^{(в)}],$$

где $[1^{(в)}]$ – единичная матрица, число строк и столбцов которой

равно числу ветвей. При замене вычислительных операций символьными коэффициенты передачи можно выразить аналитически через параметры элементов цепи (сопротивления, проводимости, ёмкости, индуктивности). Если параметры элементов цепи задать в операторной форме, то на выходе вычислительного сценария серуе будут операторные коэффициенты передачи сигналов, называемые передаточными функциями. Передаточные функции легко преобразуются в частотные характеристики при замене переменной. Обратное преобразование Лапласа от передаточных функций даёт импульсные характеристики. Чаще всего такие характеристики рассчитываются для четырёхполюсников (динамических звеньев).

Автором разработана также функция построения векторных и топографических диаграмм для визуализации результатов расчёта синусоидальных режимов цепи. За один вызов эта функция строит один контур векторной или топографической диаграммы. Построив все контуры диаграммы с помощью функции vdiagr, можно интерактивно в режиме plotedit расставить стрелки и сделать необходимые надписи. На рис. 9 показана топографическая диаграмма напряжений первой гармоники докоммутационного режима цепи, изображённой на рис. 2. Контуры диаграммы получены следующей последовательностью команд:

```
TM=[-1 0 0 1 0 -1;0 1 0 -1 -1 0; 0 0 -1 0 1 1];
omega=1;
PV1=diag([0.2 0.2 0.2 (0.2+j*omega) (0.2+j*omega)^-1 j*omega]);
SV1=[0 0 0 0 0 -60].';
CV1=[100 0 50i 0 0 0].';
серуе
проводимости ветвей
PV=PV1
источники тока ветвей
SV=SV1
ЭДС ветвей
CV=CV1
Напрqжениq ветвей   XS=
    -37.548 +    8.5431i
     53.242 +    46.128i
    -9.2108 -    12.415i
    -15.694 -    54.671i
    -44.031 -    33.714i
    -28.337 +    20.958i
vdiagr(1,[0 -XSP(1) CV(1) -XSP([4 2]).'])
```

vdiagr(1,[0 CV(3) -XSP([3 5 2]).'])
vdiagr(1,[0 CV(3) -XSP([3 6]).' -CV(1) XSP(1)])

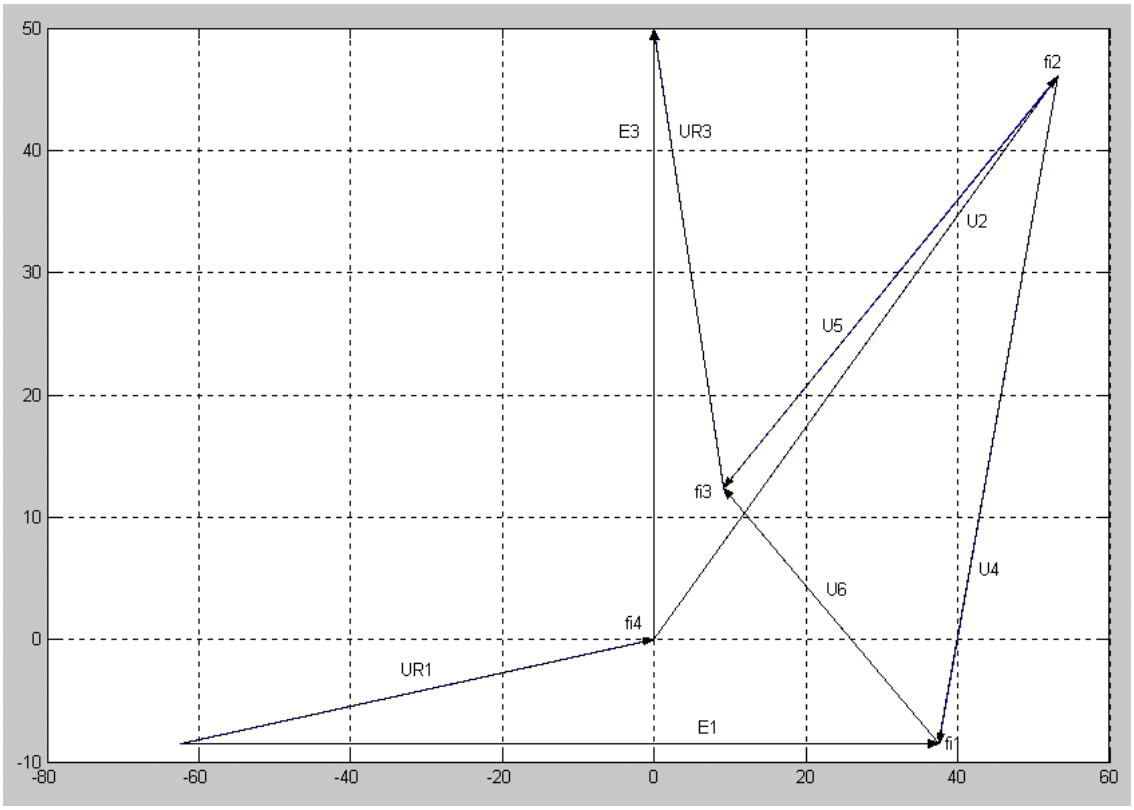


Рис. 9. Топографическая диаграмма первой гармоники режима цепи до коммутации.

Разработанный автором вычислительный сценарий **серуе**, а также функции **getqbm**, **vdiagr** являются мощным инструментом изучения практически всей теории линейных электрических цепей. Они могут использоваться также и для инженерных расчётов.

УДК 519.6

ИЗУЧЕНИЕ ОСНОВ ТЕОРИИ ЭЛЕКТРОМАГНИТНОГО ПОЛЯ С ПОМОЩЬЮ PARTIAL DIFFERENTIAL EQUATIONS TOOLBOX И ЯДРА MATLAB

Шмелёв В.Е.
Владимирский государственный университет, г. Владимир
e-mail: shmelioff_v_e@chat.ru

Введение

На кафедре «Электротехника и электроэнергетика» Владимирского государственного университета обучаются студенты по специальности 1004 «Электроснабжение». В учебный план этой специальности входит дисциплина «Теоретические основы электротехники», которая изучается студентами в течение трёх семестров. Первые два семестра – теория цепей, третий – теория электромагнитного поля (ТЭМП).

ТЭМП состоит из лекционного курса (34 часа), лабораторного практикума (17 часов), практических занятий (17 часов), расчётного задания из четырёх частей (50 часов). В практических занятиях и расчётном задании рассматриваются объекты, не требующие применения PDE Toolbox (достаточно одного ядра MATLAB). В лабораторном практикуме исследуются следующие объекты: электростатическое поле «коаксиального» кабеля со смещённой жилой, магнитостатическое поле цилиндрической катушки, цилиндрический электромагнитный экран.

Известно, что электромагнитное поле описывается дифференциальными уравнениями в частных производных (Partial Differential Equations (PDE)), иначе называемых уравнениями математической физики. Поля в названных объектах являются плоскопараллельными или осесимметричными. Распределение потенциалов этих полей можно описать эллиптическими PDE в двумерной расчётной области. Решение таких уравнений математической физики удобно выполнять с использованием PDE Toolbox MATLAB. Этот Toolbox поддерживает конечноэлементную

технологию решения эллиптических, параболических и гиперболических краевых задач.

Расчёт электростатического поля «коаксиального» кабеля

Коаксиальный кабель представляет собой длинный цилиндрический проводник, называемый жилой, находящийся в полости другого проводника, имеющего форму цилиндрической трубки и называемого оболочкой. Ось жилы совпадает с осью оболочки, т.е. жила и оболочка расположены в пространстве коаксиально. Жила и оболочка изолированы друг от друга диэлектриком, в котором может существовать электростатическое поле.

Если диэлектрик однородный, то электростатическое поле является плоскопараллелльным. При моделировании такого поля расчётной областью можно считать поперечное сечение кабеля. Оно схематично изображено на рис. 1.

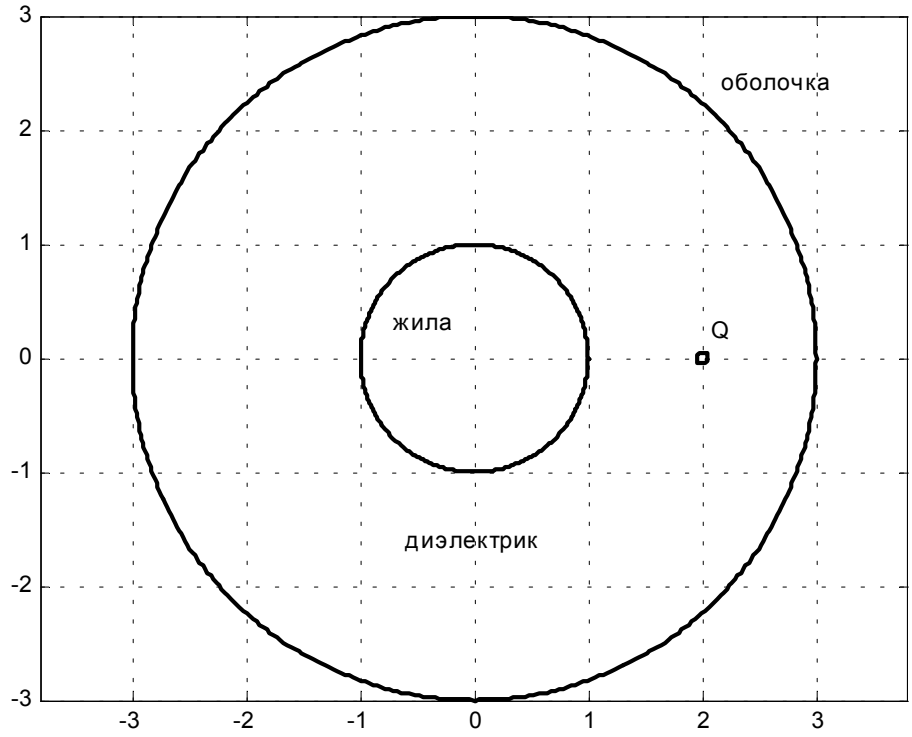


Рис. 1. Схематичное изображение поперечного сечения коаксиального кабеля.

Пусть линейная плотность заряда жилы равна $+\tau$, а заряд на единицу длины оболочки равен $-\tau$. Тогда электрическое смещение и напряжённость электрического поля в точке наблюдения Q равны:

$$\boldsymbol{D}(Q)=\mathbf{1}_r\cdot\tau/(2\pi r); \qquad \boldsymbol{E}(Q)=\mathbf{1}_r\cdot\tau/(2\pi r\varepsilon_a)\ .$$

Учитывая, что $\boldsymbol{E}(Q) = -\text{grad}\ \varphi(Q)$ и $\varphi_{об}=0$, получим

$$\varphi(Q) = \tau\cdot\ln(r_{об}/r)/(2\pi\varepsilon_a); \qquad U = \varphi_ж = \tau\cdot\ln(r_{об}/r_ж)/(2\pi\varepsilon_a);$$

$$c_0 = 2\pi\varepsilon_a/\ln(r_{об}/r_ж); \qquad \varphi(Q) = U\cdot\ln(r_{об}/r)/\ln(r_{об}/r_ж),$$

где $\varphi(Q)$ – скалярный электрический потенциал в точке наблюдения Q; r – радиальная координата точки Q; $r_{об}$ – радиус оболочки; $r_ж$ – радиус жилы; ε_a – абсолютная диэлектрическая проницаемость диэлектрика; U – напряжение между оболочкой и жилой; c_0 – ёмкость на единицу длины кабеля.

Эквипотенциалями электростатического поля в сечении коаксиального кабеля являются окружности, центры которых совпадают с центром жилы. Радиус r_k эквипотенциали φ_k определяется по формуле:

$$r_k = r_{об}\cdot(r_{об}/r_ж)\text{^}(-\varphi_k/U) \qquad (1)$$

Пусть относительная проницаемость диэлектрика ε=1, напряжение U=10В, $r_{об}$=10мм, $r_ж$=2мм, задано 10 шагов по потенциалу (через 1В), тогда c_0=0.034566пФ/мм. Картина аналитически рассчитанных по формуле (1) эквипотенциалей изображена на рис. 2.

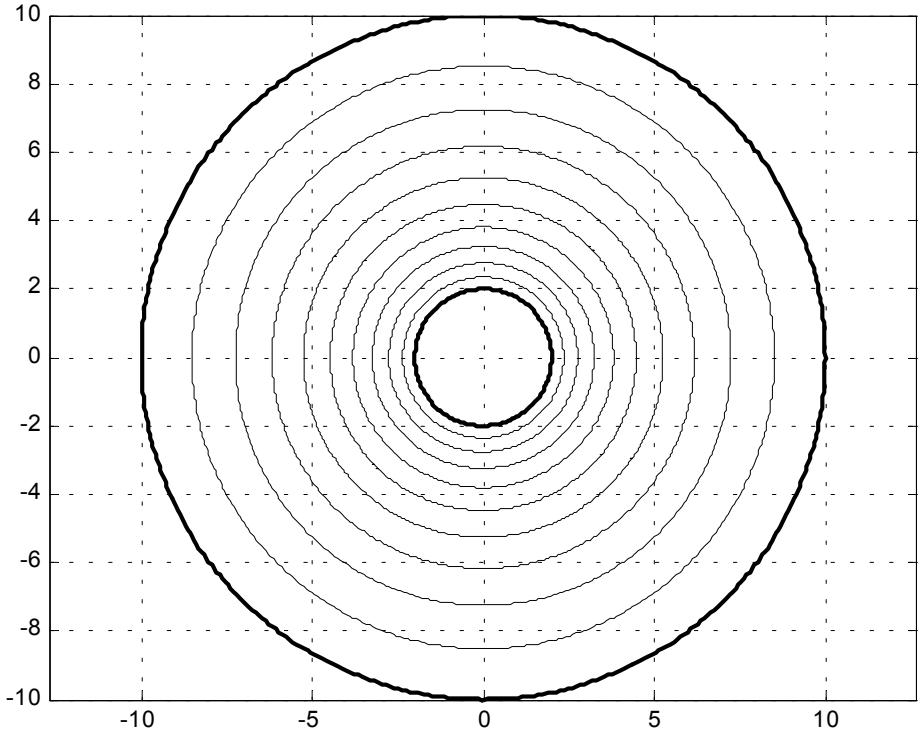


Рис. 2. Картина аналитически рассчитанных эквипотенциалей электростатического поля в коаксиальном кабеле

Если оси жилы и оболочки параллельны, но смещены друг относительно друга на расстояние d, то такой кабель будем называть «коаксиальным» со смещённой жилой. Можно доказать, что в поперечном сечении такого кабеля эквипотенциали электростатического поля представляют собой окружности разного радиуса с разными центрами. Алгоритм аналитического расчёта картины эквипотенциалей φ_k такого кабеля определяется следующими соотношениями:

$$s_1 = (r_{об}^2 - r_ж^2 - d^2)/(2d);$$
$$s_2 = (r_{об}^2 - r_ж^2 + d^2)/(2d);$$
$$a = (s_1^2 - r_ж^2)^{0,5} = (s_2^2 - r_{об}^2)^{0,5};$$
$$c_0 = \tau/U = 2\pi\varepsilon_a/\ln((s_2 - a)(s_1 + a)/(r_ж \cdot r_{об}));$$
$$\tau = c_0 U;$$
$$\varphi_ж = \tau \cdot \ln((s_1 + a)/r_ж)/(2\pi\varepsilon_a);$$
$$\varphi_{об} = \tau \cdot \ln(r_{об}/(s_2 - a))/(2\pi\varepsilon_a);$$
$$\chi_k = ((s_2 - a)(s_1 + a)/(r_ж \cdot r_{об}))\text{^}(\varphi_k/U);$$
$$s_k = a(\chi_k^2 + 1)/(\chi_k^2 - 1);$$
$$x_k = s_2 - s_k;$$
$$r_k = a|2\chi_k/(1 - \chi_k^2)|\,, \qquad (2)$$

где s_1 – расстояние от оси жилы до плоскости симметрии между электрическими осями $+\tau$ и $-\tau$; s_2 – расстояние от оси оболочки до плоскости симметрии между электрическими осями $+\tau$ и $-\tau$; a – расстояние между плоскостью симметрии и одной из осей $+\tau$ или $-\tau$; c_0 – ёмкость кабеля на единицу длины; $U = \varphi_ж - \varphi_{об}$ – напряжение между жилой и оболочкой; $\varphi_ж$, $\varphi_{об}$, φ_k – потенциалы жилы, оболочки и рассчитываемой эквипотенциали, отсчитываемые относительно плоскости симметрии; χ_k – отношение расстояний от точки наблюдения до отрицательной и положительной оси, характерное для всех точек эквипотенциали φ_k; s_k – расстояние от оси эквипотенциали до плоскости симметрии между электрическими осями $+\tau$ и $-\tau$; x_k – координата центра эквипотенциали φ_k относительно центра поперечного сечения оболочки; r_k – радиус эквипотенциали φ_k.

Пусть относительная проницаемость диэлектрика $\varepsilon=1$, напряжение U=10В, $r_{об}$=10мм, $r_ж$=2мм, смещение d=4мм, задано 10 шагов по потенциалу (через 1В), тогда c_0=0.039029пФ/мм. Картина аналитически рассчитанных по формулам (2) эквипотенциалей изображена на рис. 3.

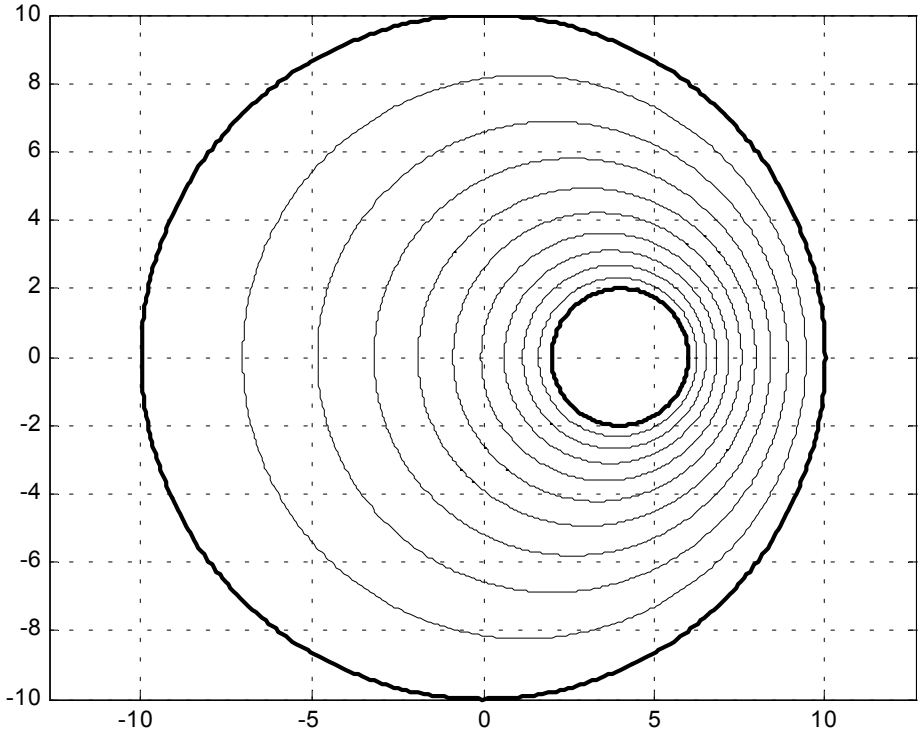


Рис. 2. Картина аналитически рассчитанных эквипотенциалей электростатического поля в «коаксиальном» кабеле со смещением

До внедрения системы MATLAB в учебный процесс электростатическое поле «коаксиального» кабеля со смещённой жилой моделировалось электрическим полем в водопроводной воде, налитой в электролитическую ванну цилиндрической формы. Роль «жилы кабеля» выполнял цилиндрический электрод. Напряжение подавалось от генератора сигналов низкочастотного. Скалярный электрический потенциал в точках наблюдения измерялся электронным вольтметром. Нарушение герметичности диэлектрического дна ванны заставило нас проводить экспериментальную часть этой лабораторной работы на «виртуальной» модели, прорисовываемой в GUI-приложении PDETool. В расчётную часть работы входит аналитическое определение эквипотенциалей (см. выше) и сравнение их с линиями, полученными в PDETool.

Прорисовка геометрии поперечного сечения коаксиального кабеля показана на рис. 3. В лабораторной работе такую прорисовку удобнее всего производить из командного окна MATLAB:

pdecirc(0,0,10,'C1')

pdecirc(0,0,2,'C2')

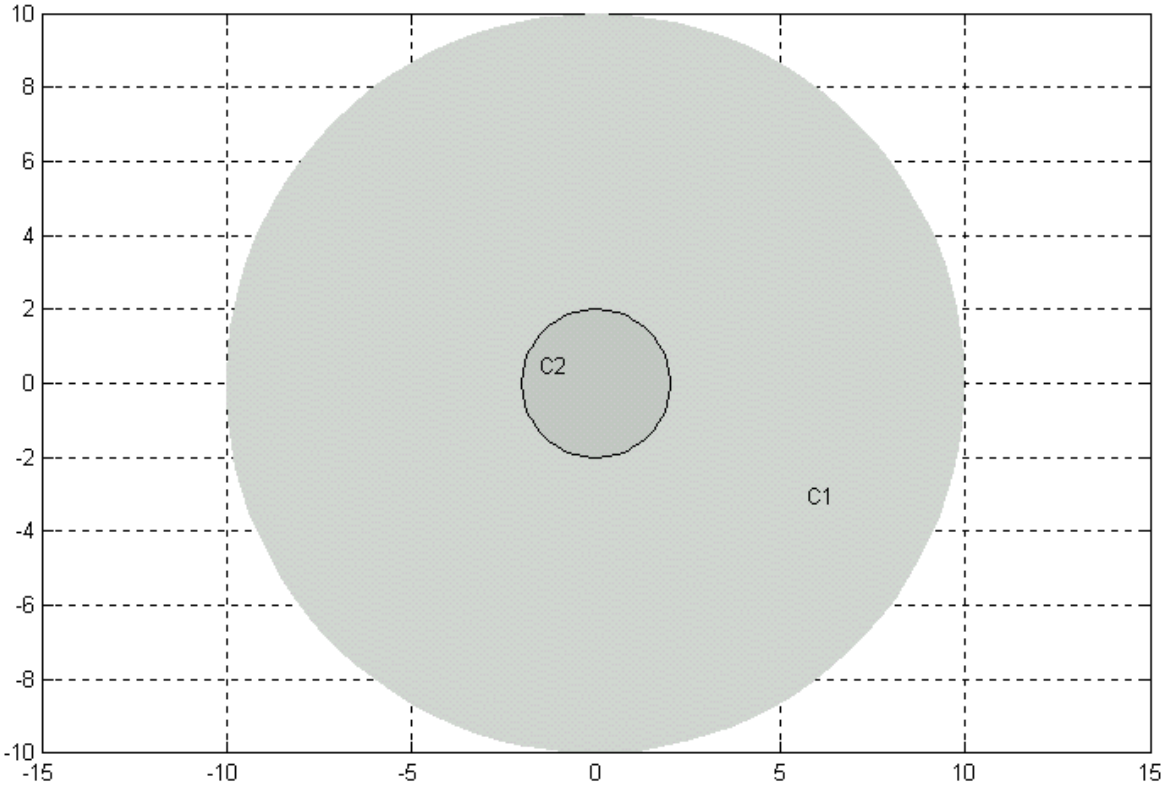


Рис. 3. Геометрия сечения коаксиального кабеля, прорисованная в PDETool

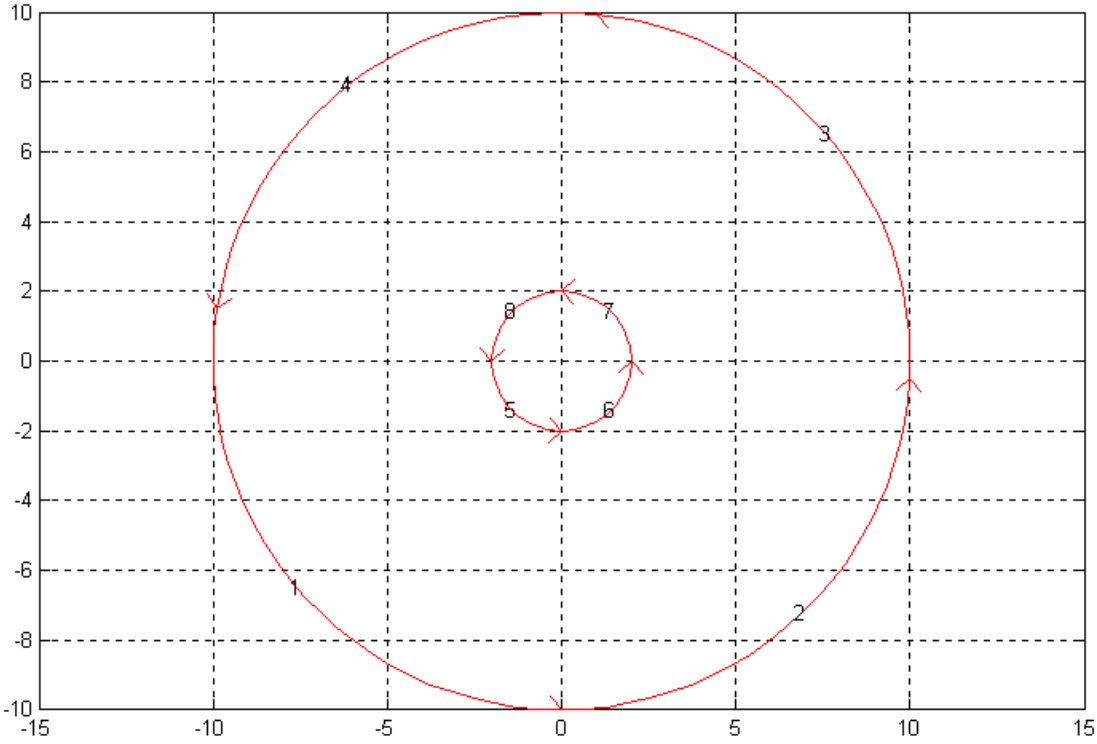


Рис. 4. Граничные сегменты расчётной области

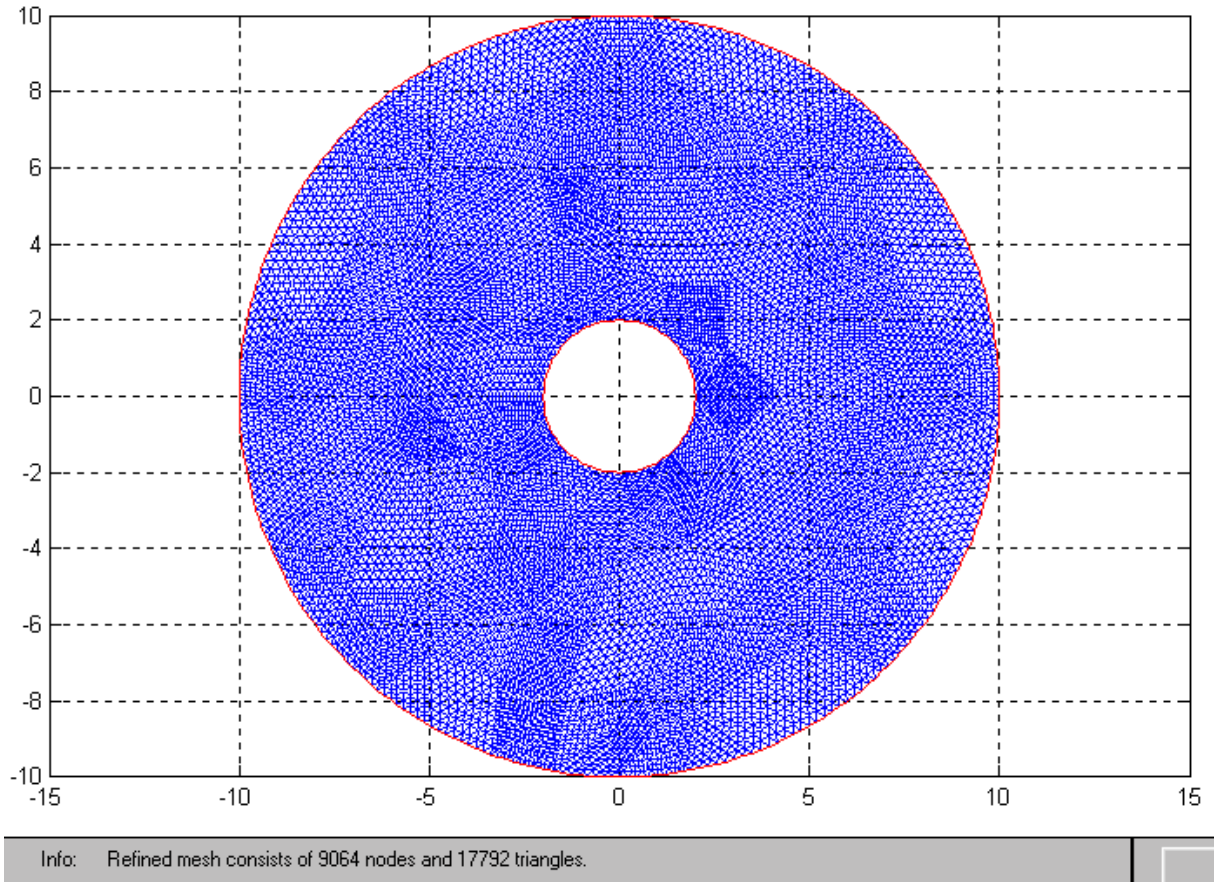


Рис. 5. Конечноэлементная сетка

Разметка граничных сегментов для ввода граничных условий показана на рис. 4. Сегменты 1, 2, 3, 4 принадлежат оболочке, сегменты 5, 6, 7, 8 принадлежат жиле. Выделим сегменты 5, 6, 7, 8 и введём для них граничные Дирихле: $\varphi=10$. Нулевые граничные условия Дирихле для сегментов 1, 2, 3, 4 уже заданы по умолчанию.

Сгенерированная конечноэлементная сетка после трёх переопределений изображена на рис. 5. В этой сетке 9064 узла и 17792 треугольников. Численно рассчитанное распределение скалярного электрического потенциала в диэлектрике в виде эквипотенциальных линий представлено на рис. 6. Визуальное сравнение рис. 2 и рис. 6 свидетельствует о совпадении аналитически и численно рассчитанной картины поля.

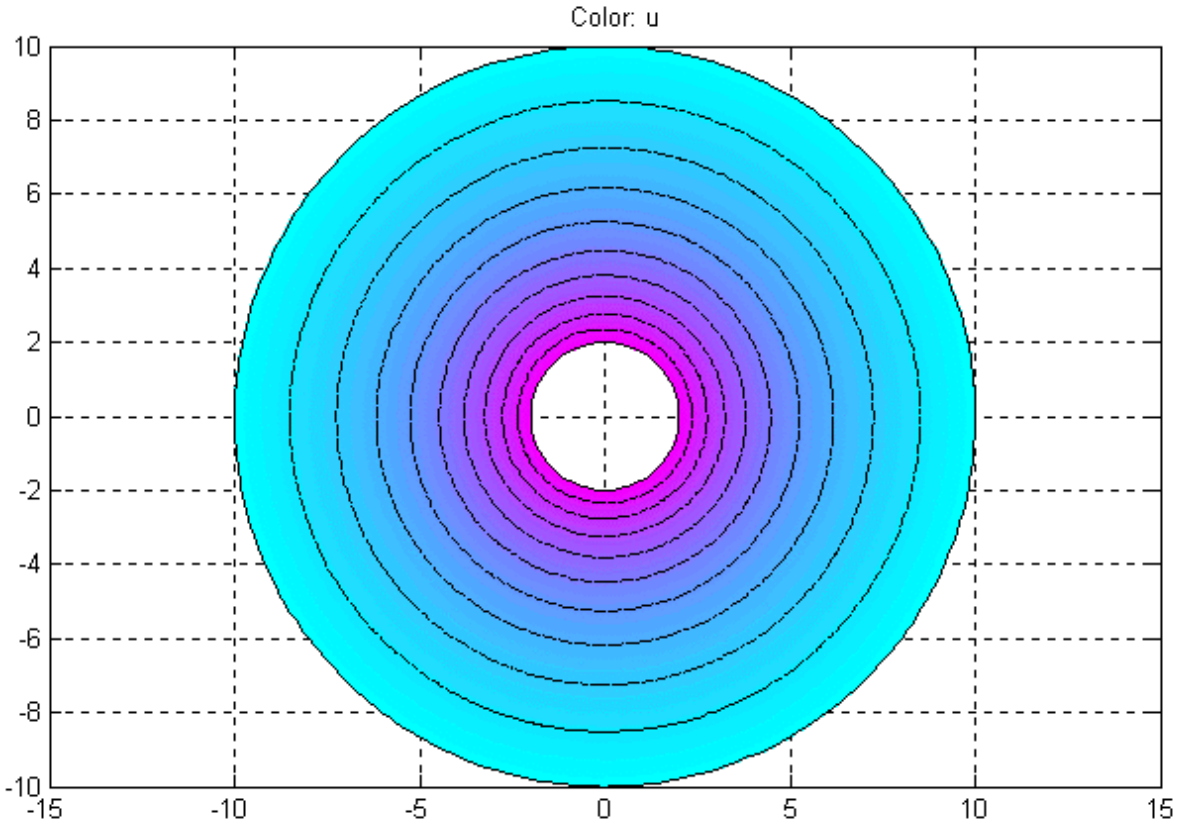


Рис. 6. Численно рассчитанная картина электростатического поля

Для расчёта ёмкости кабеля на единицу длины нужно определить энергию электрического поля, сосредоточенную в диэлектрике. Для этого надо выполнить следующую последовательность действий:

1. Экспортировать в рабочую область MATLAB конечноэлементную сетку и узловое распределение скалярного электрического потенциала;
2. С помощью функции **pdegrad** вычислить элементное распределение напряжённости электрического поля и проинтегрировать её квадрат по всей расчётной области;
3. Определить ёмкость на единицу длины по формуле c0=2Э/U2.

Пункты 2 и 3 выполняются следующей последовательностью команд:

```
[ex,ey]=pdegrad(p,t,-u);
ar=pdetrg(p,t);
HE=(ex.^2+ey.^2)*ar.'
HE                                                    =
```

```
     390.38
eps0=8.85419e-3; % Абсолютнаq диэлектрическаq
                 %проницаемость вакуума, пФ/мм
c00=eps0*HE/100
c00 =
    0.034565
```

Как видно, ёмкость, полученная численным способом, равна c_{00}=0.034565пФ/мм. Аналитически рассчитанная ёмкость равна c_0=0.034566пФ/мм. Отличие заметно только в пятой значащей цифре.

В PDETool опять перейдём в режим “Draw Mode” и подвинем жилу на 4 мм вправо. По аналогии рассчитаем картину поля и ёмкость на единицу длины. Картина эквипотенциалей изображена на рис. 7. Эта картина получена на сетке, состоящей из 8616 узлов и 16896 элементов. Ёмкость, полученная численным способом, равна c_{00}=0.039029пФ/мм. Аналитически рассчитанная ёмкость равна c_0=0.039029пФ/мм. Ошибка численного счёта $c_{00}-c_0$=2.7485e-7 (отличие заметно только в шестой значащей цифре).

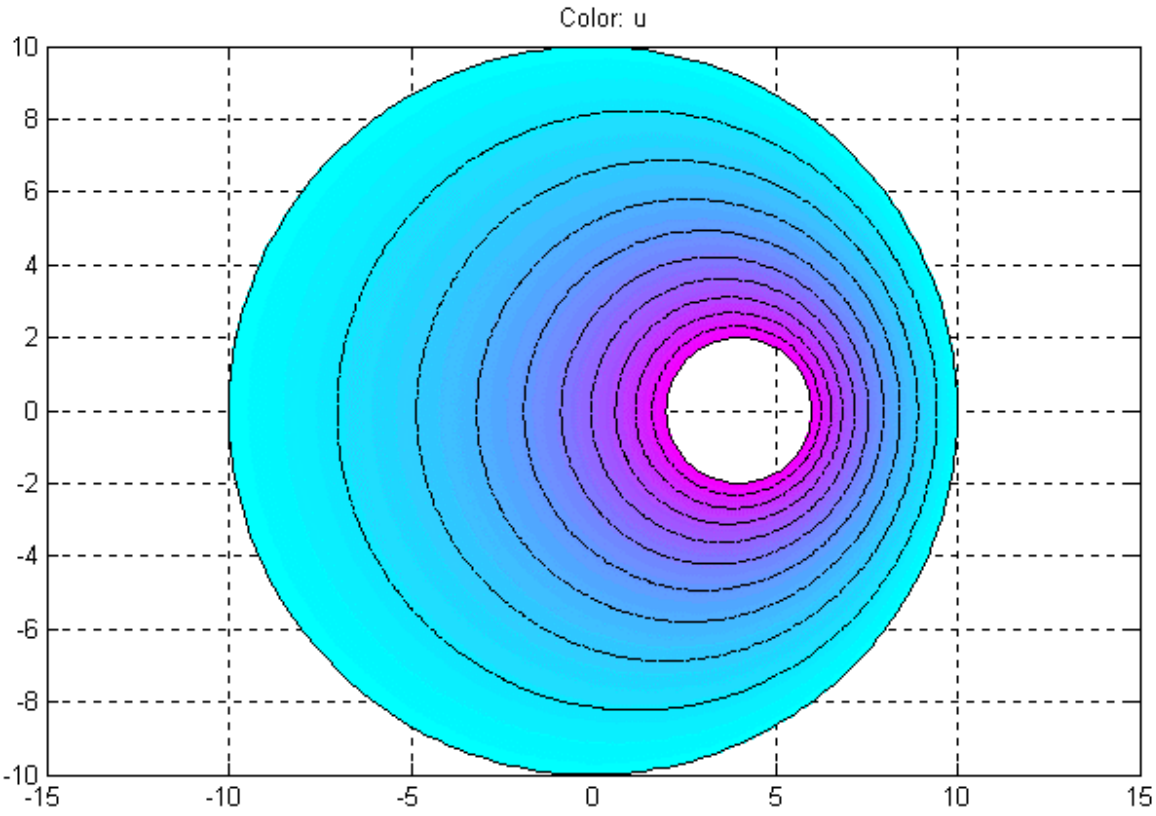


Рис. 7. Численно рассчитанная картина электростатического поля кабеля со смещённой жилой

Моделирование магнитостатического поля цилиндрической катушки

Схематичное изображение осевого сечения цилиндрической катушки без сердечника показано на рис. 8. Будем считать, что ток I течёт перпендикулярно плоскости сечения, и магнитный момент направлен вдоль оси x.

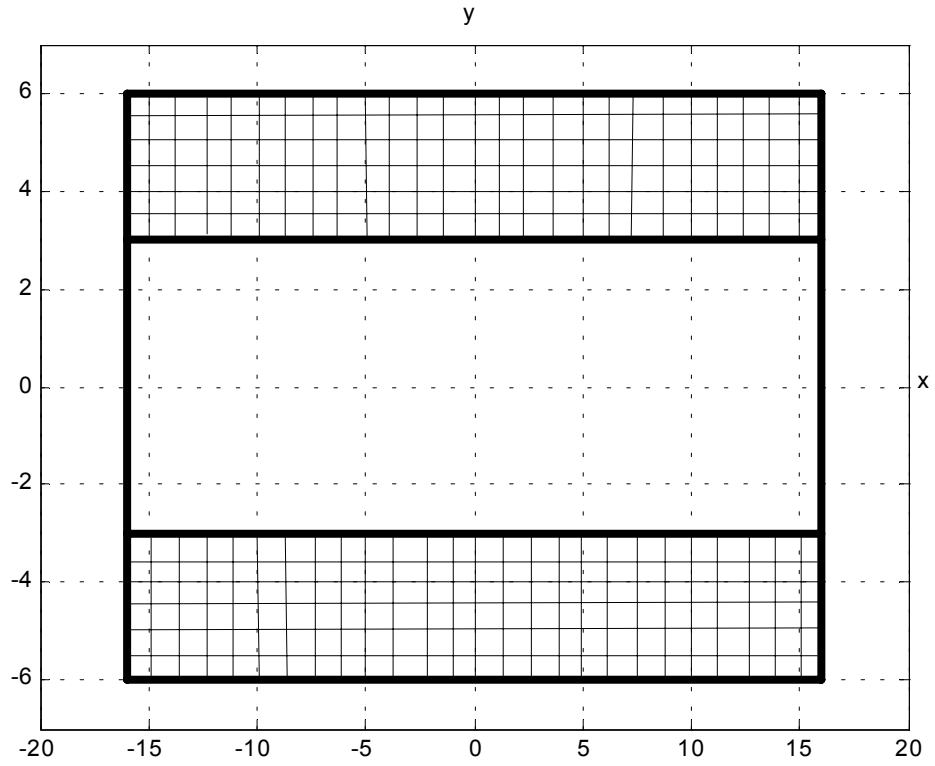


Рис. 8. Осевое сечение цилиндрической катушки

Общее уравнение магнитостатики относительно векторного магнитного потенциала имеет вид:

$$\mathrm{rot}\,(\mu^{-1}\,\mathrm{rot}\,\boldsymbol{A}') = \boldsymbol{\delta} + \mathrm{rot}\,(\mu^{-1}\boldsymbol{M}_r), \qquad (3)$$

где μ – относительная магнитная проницаемость вещества; $\boldsymbol{A}'$– векторный магнитный потенциал, измеряемый в единицах тока; $\boldsymbol{\delta}$ – векторное поле плотности тока; $\boldsymbol{M}_r$ – векторное поле остаточной намагниченности вещества. В рассматриваемом случае $\boldsymbol{M}_r$=0, потому что в анализируемой системе нет постоянных магнитов.

Векторный потенциал и плотность тока можно выразить в цилиндрических координатах. Учитывая, что $\boldsymbol{A}'=\mathbf{1}_\theta \cdot A'$, $\boldsymbol{\delta}=\mathbf{1}_\theta \cdot \delta$, (3) можно преобразовать в двумерную форму:

$$-\frac{\partial}{\partial x}\left(y^{-1}\mu^{-1}\frac{\partial(yA')}{\partial x}\right)-\frac{\partial}{\partial y}\left(y^{-1}\mu^{-1}\frac{\partial(yA')}{\partial y}\right)=\delta \qquad (4)$$

Поскольку магнитостатическое поле цилиндрической катушки симметричное, достаточно его рассчитывать только в первом квадранте системы координат (x≥0, y≥0). Уравнение (4) можно решать непосредственно в PDETool относительно yA'. На рис. 9 показано распределение yA' от катушки длиной l=32см, внутренним радиусом $r_{вн}$=3см, наружным радиусом $r_{н}$=6см, если ток I=2,5А, число витков w=3672. На рис. 10 показано распределение A' от той же катушки.

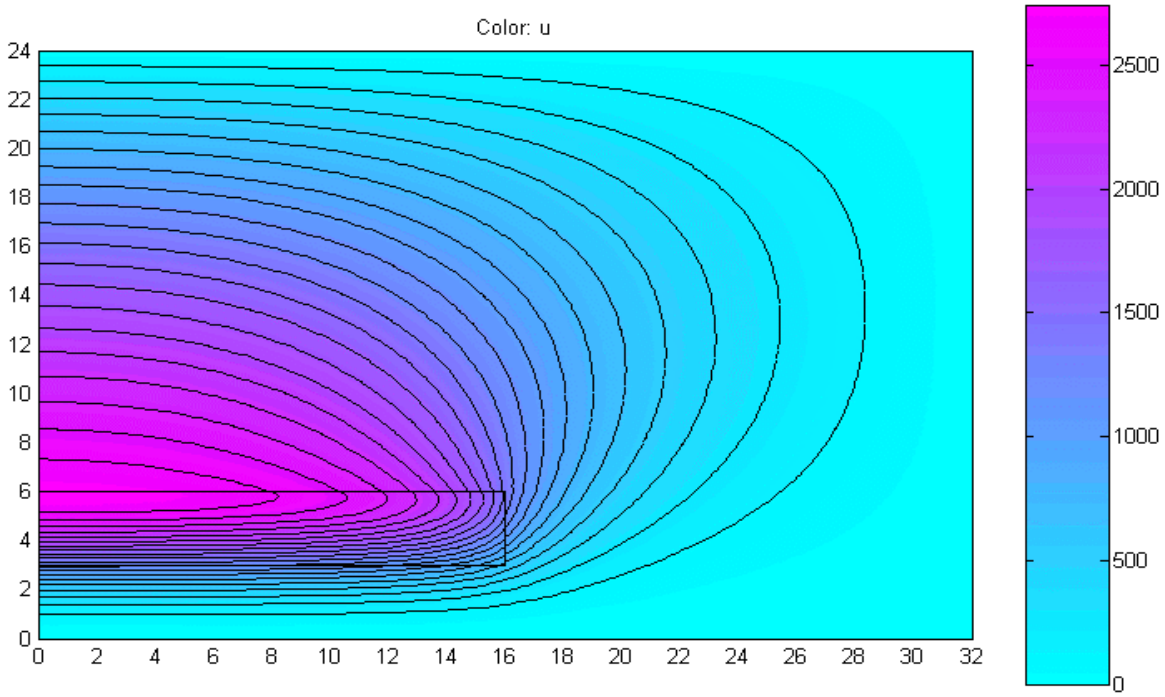


Рис. 9. Распределение yA'

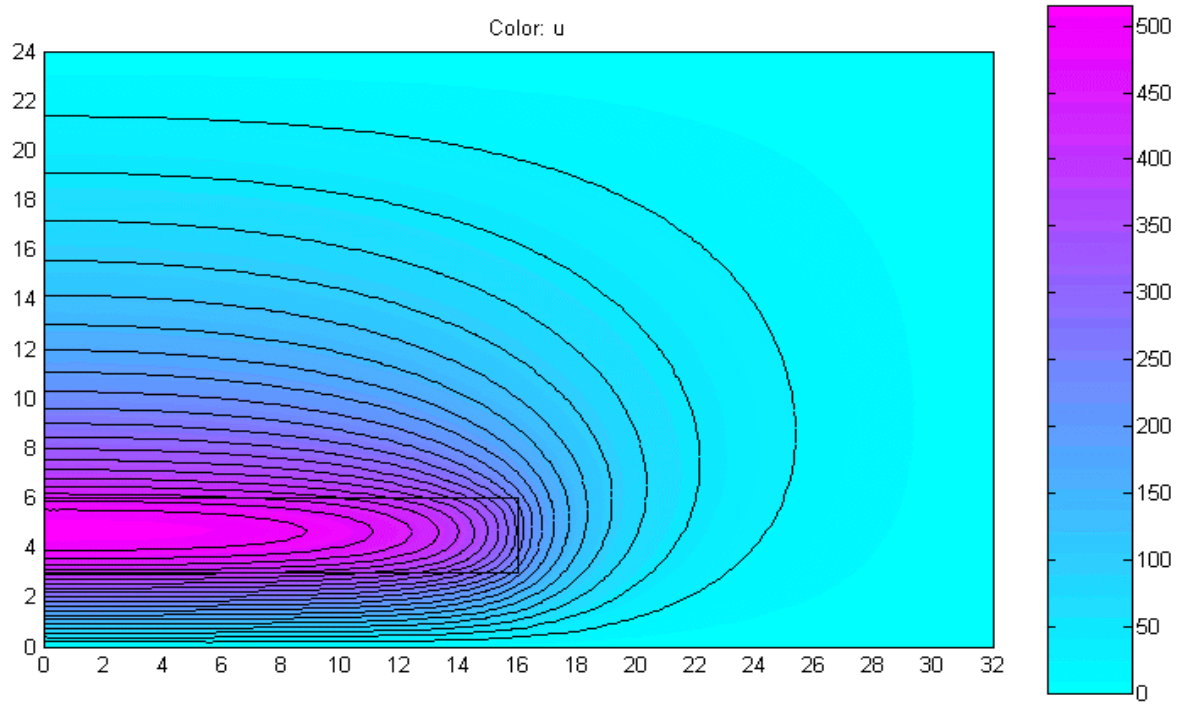


Рис. 10. Распределение A'

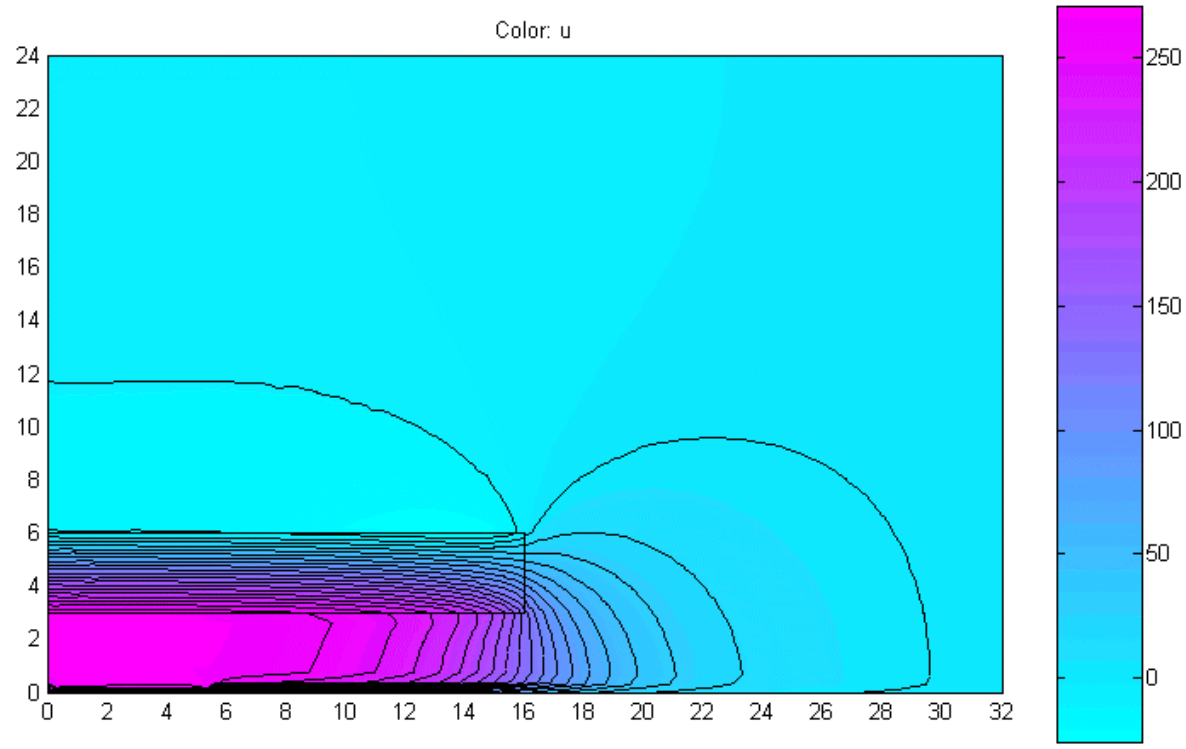


Рис. 11. Распределение x- составляющей напряжённости магнитного поля

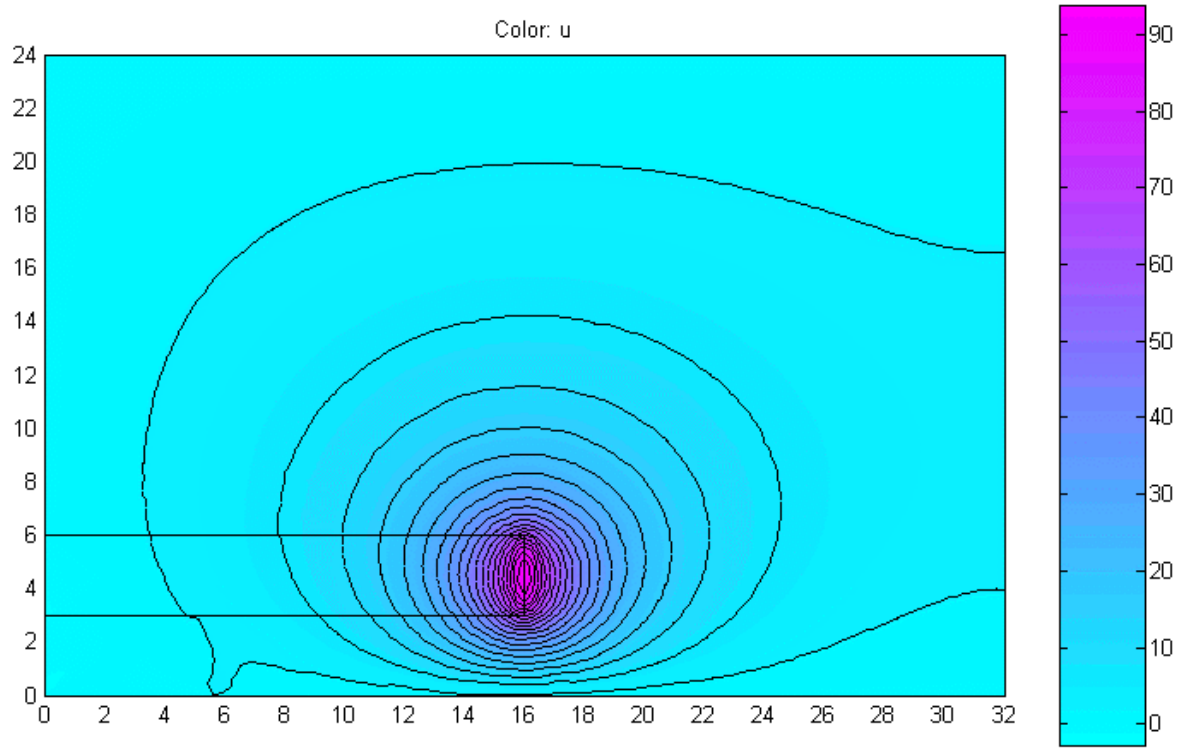


Рис. 12. Распределение y- составляющей напряжённости магнитного поля

На рис. 11, 12 показано распределение вектора напряжённости магнитного поля. Как видно, имеется некоторая численная неустойчивость при решении уравнения (4). Однако решение, данной краевой задачи удобно при вычислении потокосцеплений, а также собственных и взаимных индуктивностей. Например, магнитное потокосцепление в рассчитываемой катушке равно 0.58051Вб, а индуктивность L=0.58051/2.5=0.2322Гн.

Применительно к цилиндрической катушке большей численной устойчивостью обладает решение скалярной краевой задачи магнитостатики. Эта задача базируется на PDE относительно скалярного магнитного потенциала:

$$\mathrm{div}(\mu_a \,\mathrm{grad}\, \varphi_H) = \mathrm{div}(\mu_a \boldsymbol{M}_\phi + \boldsymbol{B}_r)\,, \qquad (5)$$

где φ_H – скалярный потенциал напряжённости магнитного поля (скалярный магнитный потенциал); μ_a – абсолютная магнитная проницаемость вещества; $\boldsymbol{M}_\phi$ – «фиктивная» намагниченность вещества, эквивалентирующая токи; $\boldsymbol{B}_r$ – вектор остаточной магнитной индукции постоянных магнитов; в случае с катушкой $\boldsymbol{B}_r$=0.

Применительно к осесимметричному полю рассматриваемой катушки уравнение (5) в цилиндрических координатах имеет вид:

$$-\frac{\partial}{\partial x}\left(y\mu\frac{\partial\varphi_H}{\partial x}\right)-\frac{\partial}{\partial y}\left(y\mu\frac{\partial\varphi_H}{\partial y}\right)=-\frac{\partial}{\partial x}\left(y\mu M_\phi\right) \qquad (6)$$

Уравнение (6) не может решаться в PDETool, поэтому для моделирования магнитного поля катушки написан специальный вычислительный сценарий. PDETool используется только для прорисовки геометрии и генерации конечноэлементной сетки.

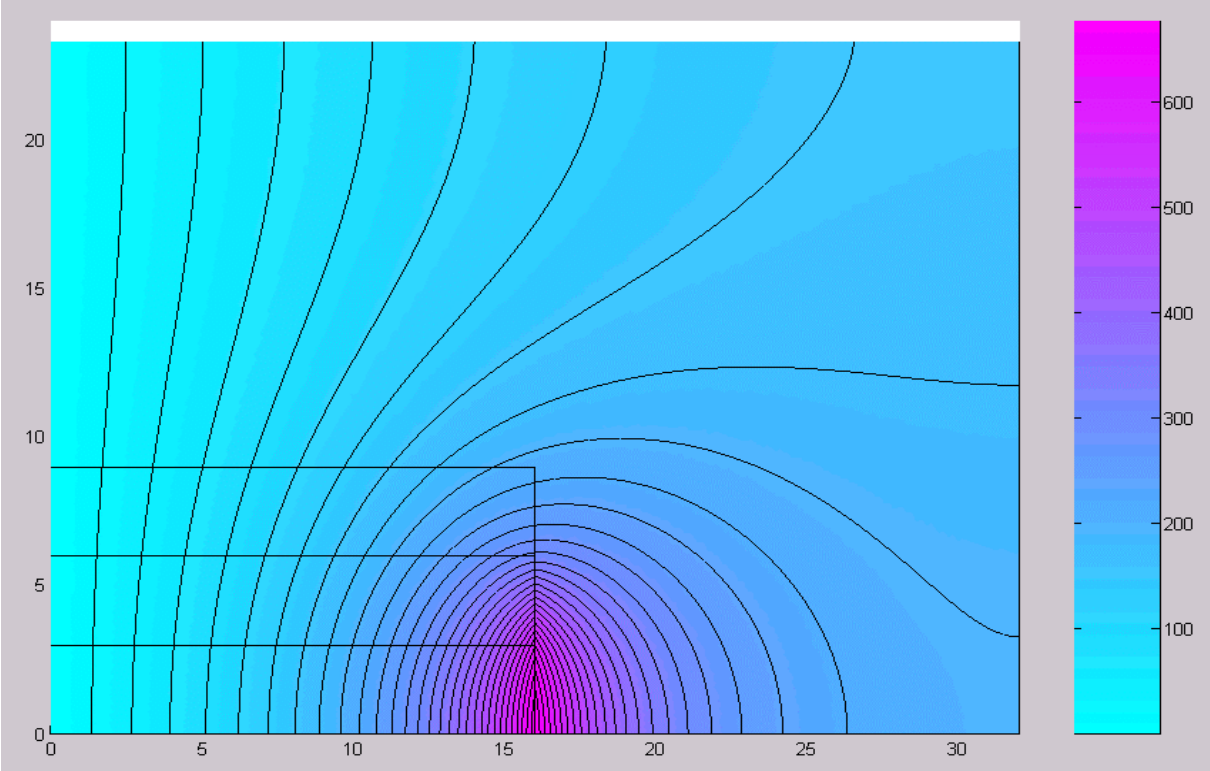


Рис. 13. Распределение скалярного магнитного потенциала

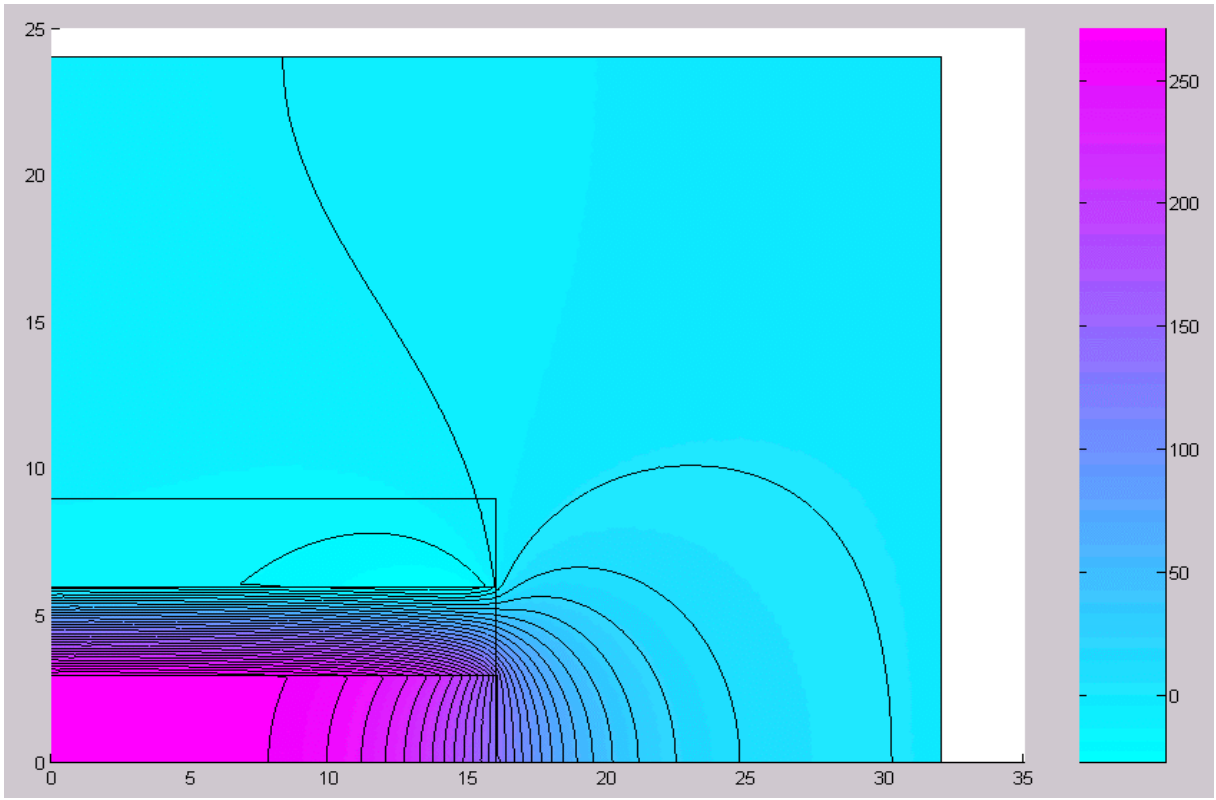


Рис. 14. Распределение x- составляющей напряжённости магнитного поля

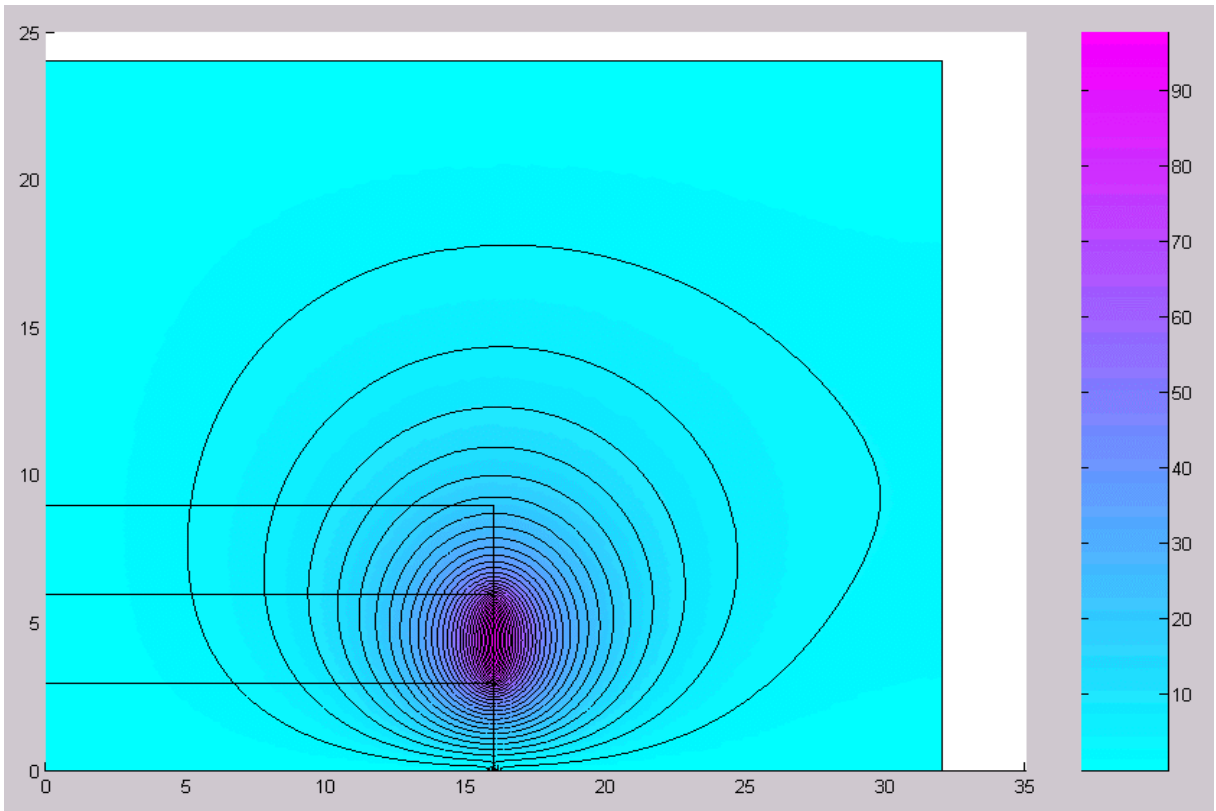


Рис. 15. Распределение y- составляющей напряжённости магнитного поля

На рис. 13, 14, 15 показано распределение магнитостатического поля, рассчитанное путём решения уравнения (6). Эти рисунки показывают, что скалярная краевая задача магнитостатики в случае цилиндрической катушки обладает большей численной устойчивостью, чем векторная краевая задача. Потокосцепления и индуктивности удобнее выражать через векторный потенциал, а магнитные напряжения – через скалярный потенциал. Цилиндрическая катушка является наглядным примером зависимости магнитного напряжения между точками от пути, соединяющего эти точки. Магнитное напряжение между торцевыми точками катушки, лежащими на оси, равно 7824.9A, если путь лежит внутри канала, и 1355.1A, если путь лежит вне канала и вне области обмотки (эти числовые значения получены из анализа распределения скалярного потенциала).

На оси катушки напряжённость магнитного поля можно выразить аналитически:

$$H(x)=\frac{Iw}{2l(r_{\text{н}}-r_{\text{вн}})}\left(\begin{array}{l}(x+l/2)\left(\operatorname{arcsinh}\left(\frac{r_{\text{н}}}{|x+l/2|}\right)-\operatorname{arcsinh}\left(\frac{r_{\text{вн}}}{|x+l/2|}\right)\right)- \\ -(x-l/2)\left(\operatorname{arcsinh}\left(\frac{r_{\text{н}}}{|x-l/2|}\right)-\operatorname{arcsinh}\left(\frac{r_{\text{вн}}}{|x-l/2|}\right)\right)\end{array}\right)$$

График напряжённости на оси катушки показан на рис. 16.

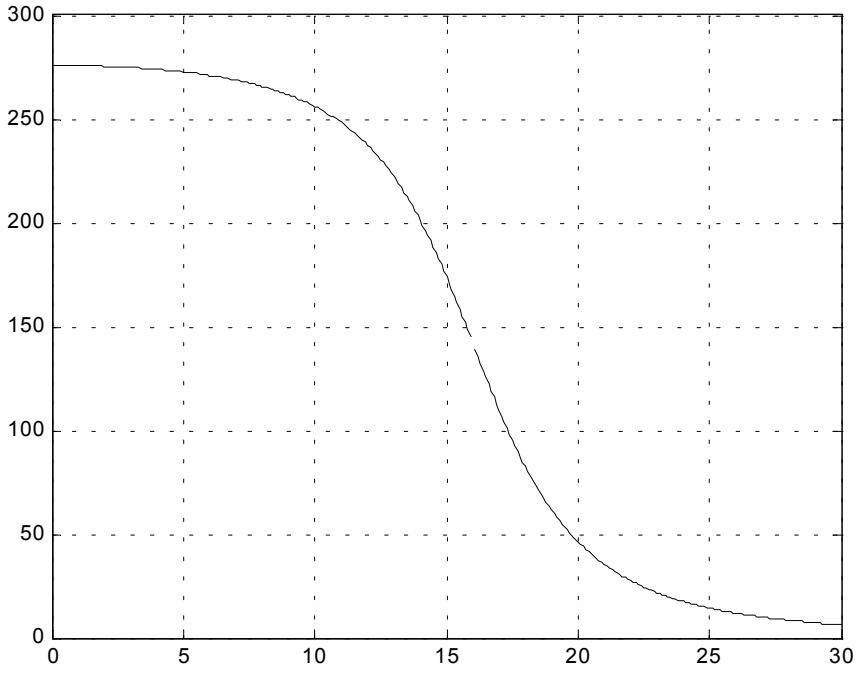


Рис. 16. График напряжённости на оси катушки.

Моделирование цилиндрического электромагнитного экрана

Переменное гармоническое электромагнитное поле в системе с цилиндрическим электромагнитным экраном описывается эллиптическим PDE в комплексной форме:

$$-\mathrm{div}\,(\underline{\mu}^{-1}\ \mathrm{grad}\ \underline{E}) + \left(j\frac{\omega}{c} Z_{\text{в}}\gamma - \frac{\omega^2}{c^2}\varepsilon \right)\underline{E} = 0, \qquad (7)$$

где $\underline{E}$ – комплексное действующее значение z- составляющей напряжённости электрического поля; $c = (\mu_0\,\varepsilon_0)^{-0.5} = 2.9979*10^8$ м/с – скорость света в вакууме; $Z_{\text{в}} = (\mu_0/\varepsilon_0)^{0.5} = 376{,}7$ Ом – волновое сопротивление вакуума; γ – удельная электрическая проводимость вещества. Краевая задача, основанная на уравнении (7), является стандартной для GUI-приложения PDETool, поэтому написание своего сценария не требуется.

Для запуска PDETool и прорисовки геометрии выполним следующую последовательность команд:

```
pdeinit
pderect([-8 8 -6 6],'R1')
pdecirc(0,0,4,'C1')
pdecirc(0,0,3,'C2')
pderect([-1.5 1.5 1 1.5],'R2')
pderect([-1.5 1.5 -1 -1.5],'R3')
```

Это означает, что расчётная область – прямоугольник (16×12)мм; экран с внутренним радиусом 3 мм и наружным радиусом 4 мм; области ‘R2’ и ‘R3’ – сечения измерительной обмотки, в которой наводится ЭДС электромагнитной индукции, подаваемая на электронный вольтметр. Пусть везде μ=1, γ=0, ε=1, частота ω=2*2*pi рад/мс, а в экране γ=5,6*10^4 См/мм (т.е. экран медный). Источником электромагнитного поля являются граничные условия Дирихле: на прямых y=±6мм $\underline{E}$=±10мкВ/мм. На прямых x=±8мм заданы нулевые граничные условия Неймана.

Для вычисления ЭДС в измерительной обмотке экспортируем сетку и решение PDE в рабочую область MATLAB и выполним следующую последовательность команд:

```
ut=pdeintrp(p,t,u);
ar=pdetrg(p,t);
eds=ut(t(4,:)==4)*ar(t(4,:)==4).'/sum(ar(t(4,:)==4));
```

```
eds=eds-ut(t(4,:)==3)*ar(t(4,:)==3).'/sum(ar(t(4,:)==3))
eds =

   1.8332 -   2.3397i

[abs(eds) angle(eds)*180/pi]

ans =

   2.9723  -51.92
```

Приведённые числовые данные означают следующее: действующее значение синусной составляющей ЭДС равно 1.8332мкВ/(мм·виток), косинусной составляющей равно – 2.3397мкВ/(мм·виток), действующее значение ЭДС (вольтметр покажет) 2.9723мкВ/(мм·виток), а её запаздывание по фазе относительно сторонней ЭДС на границе расчётной области составляет 51.92°.

На рис. 17 показано распределение действующего значения синусной составляющей напряжённости электрического поля (а заодно и линий синусной составляющей магнитной индукции). На рис. 18 – то же для косинусной составляющей. На рис. 19 – распределение действующего значения напряжённости электрического поля. Как видно, экранирующее действие основано на поверхностном эффекте.

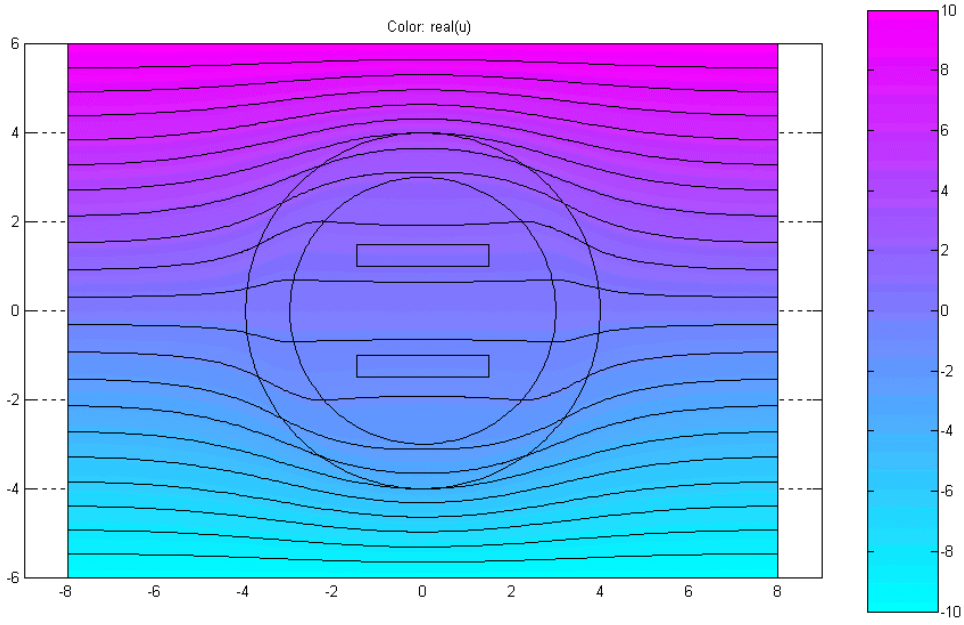


Рис. 17. Распределение синусной составляющей напряжённости поля

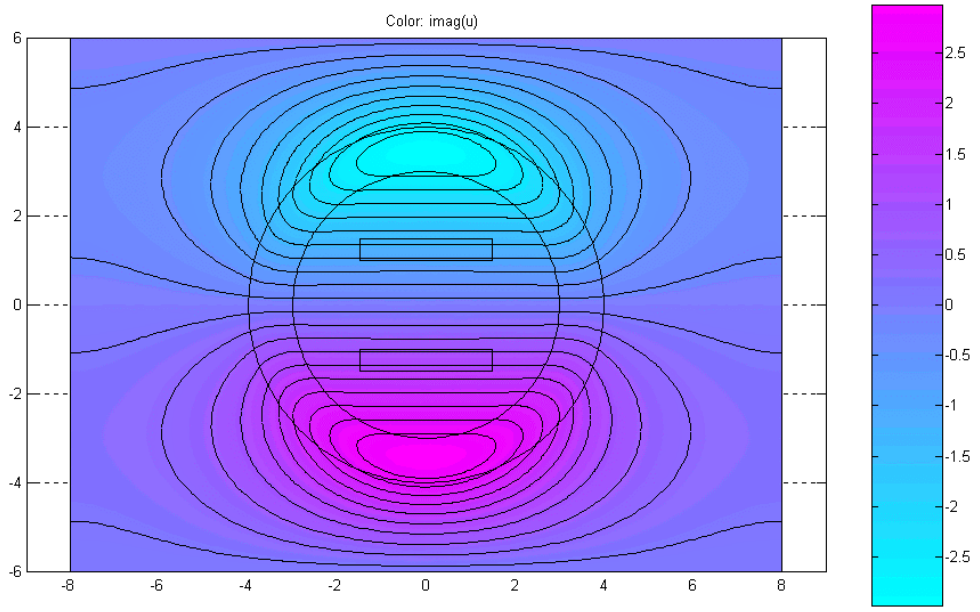


Рис. 18. Распределение косинусной составляющей напряжённости поля

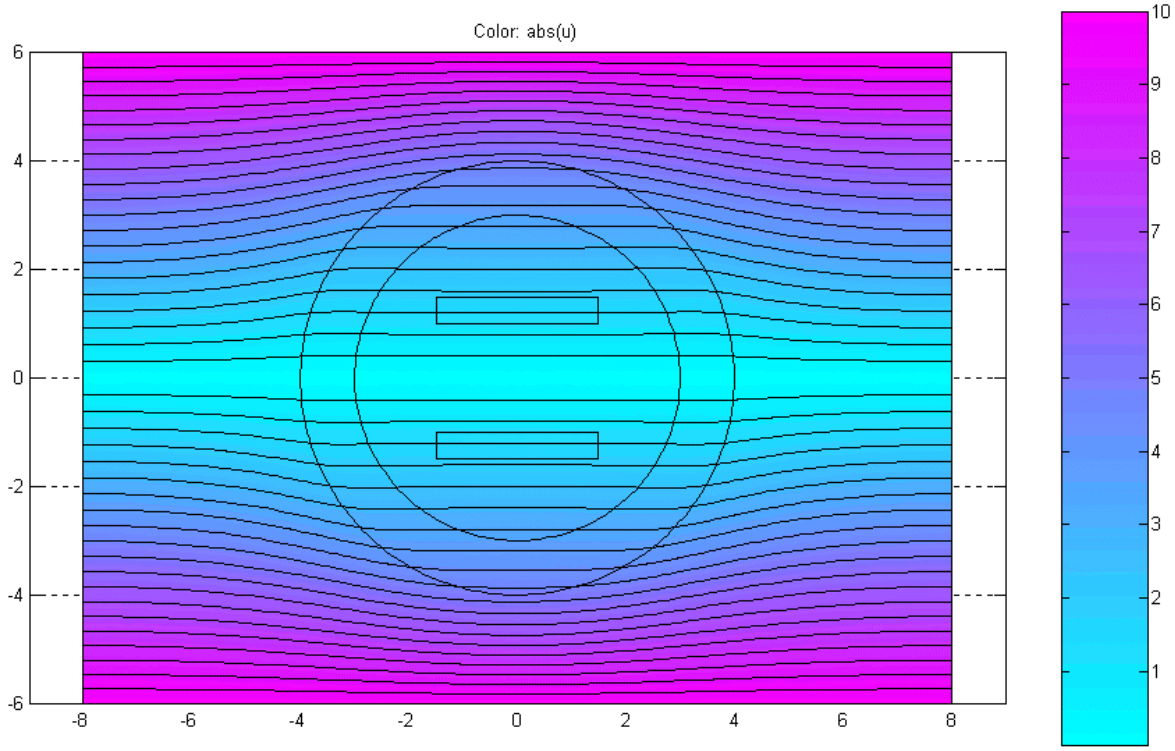


Рис. 18. Распределение действующего значения напряжённости поля

Теперь посчитаем ЭДС в обмотке при отсутствии экрана. Для этого в PDETool решим уравнение (7) при $\gamma=0$ везде и экспортируем решение в рабочую область MATLAB. Затем выполним последовательность команд:

```
ut=pdeintrp(p,t,u);
eds=ut(t(4,:)==4)*ar(t(4,:)==4).'/sum(ar(t(4,:)==4));
```

```
eds=eds-ut(t(4,:)==3)*ar(t(4,:)==3).'/sum(ar(t(4,:)==3))
eds =
     4.1667
kekr=eds/(1.8332-2.3397i)
kekr =
   0.86457 +    1.1034i
[abs(kekr) angle(kekr)*180/pi]
ans =
     1.4018      51.921
```

Приведённые числовые данные означают, что «экспериментальная» эффективность экранирования равна 1.4018.

Теперь для интереса посчитаем эффективность экранирования стального экрана, у которого μ=120, γ=10^4 См/мм, магнитный гистерезис не учитываем.

```
ut=pdeintrp(p,t,u);
eds=ut(t(4,:)==4)*ar(t(4,:)==4).'/sum(ar(t(4,:)==4));
eds=eds-ut(t(4,:)==3)*ar(t(4,:)==3).'/sum(ar(t(4,:)==3))
eds =
 2.4709e-006 -1.9731e-005i
[abs(eds) angle(eds)*180/pi]
ans =
 1.9885e-005     -82.862
4.1667/abs(eds) %          Это эффективность экранирования.
ans =
 2.0953e+005
```

Заключение

Автором разработаны также учебные вычислительные сценарии, использующие только ядро MATLAB, для выполнения расчётного задания: расчёт потенциальных и ёмкостных коэффициентов, частичных ёмкостей воздушной линии с учётом влияния земли; расчёт магнитостатического поля коаксиального кабеля с многослойными жилой и оболочкой; расчёт поля растекания тока заземлителя; расчёт поверхностного эффекта в плоской проводящей пластине.

Предложенные автором технологические приёмы использования PDE Toolbox MATLAB, а также разработанные m-файлы могут применяться не только в учебном процессе, но и для решения инженерных и научных задач.

УДК 330

ВИЗУАЛЬНОЕ МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ В MATLAB

Шампанер Г.М.
Алтайский государственный университет, г.Барнаул
e-mail: gsham@mail.ru

Современное развитие информационных и коммуникационных технологий привело к возможности использования их для преподавания предметов естественно - научного цикла, и в частности физики. Использование новых технологий в образовательном процессе приводит к развитию новых методов и способов подачи учебного материала; структурным изменениям в педагогической системе.

При изучении физических процессов основную роль в понимании результатов играет визуализация, причем от ее качества зависит эффективность работы студента, исследователя. Особенно важную роль визуализация должна играть в тех разделах современной физики, где поведение объекта исследования и его характеристик не всегда очевидно из описывающих какой-либо процесс формул. Модели являются эффективным средством развития познавательной деятельности студентов, позволяют углублять понимание учебного материала.

Проблема получения качественных знаний оказывается особенно важной при изучении студентами курса "Компьютерное моделирование в физике", так как отсутствие наглядного образа часто является проблемой при анализе характеристик различных физических явлений. Эту проблему помогает решить пакет MATLAB, содержащий инструмент визуального моделирования Simulink.

В Алтайском государственном университете на кафедре общей физики в рамках спецкурса ведется курс "Компьютерное моделирование в физике". Целью данного курса является знакомство студентов с некоторыми методами создания и исследования моделей физических процессов. В основе курса лежит изучение пакета MATLAB на практических примерах. Курс состоит из двух блоков: лекционный и лабораторный.

В лекционной части курса рассмотрены общие вопросы моделирования, основы технологии имитационного моделирования, в том числе моделирование случайных факторов; управление модельным временем; моделирование параллельных процессов; обработка и анализ результатов моделирования, теоретический аспект работы с пакетом MATLAB; библиотеки Simulink, основы моделирования в среде Simulink, а также задачи механики и молекулярной физики. Занятия проводятся в виде лекционного и лабораторного практикума, в ходе которого студенты строят математическую модель явления, и представляют ее с помощью, как самого пакета, так и инструмента визуального моделирования Simulink. После построения модели производится задание начальных параметров и условий, затем осуществляется процесс обработки событий объектами построенной модели. Практикум проводится как совокупность занятий по изучению основ физической теории, математических методов, необходимых при моделировании и непосредственной работе в среде MATLAB.