МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

Е.И. КУРКИН, Е.А. КИШОВ

ОСНОВЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ АВИАЦИОННЫХ КОНСТРУКЦИЙ

Рекомендовано редакционно-издательским советом федерального государственного автономного образовательного учреждения высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» в качестве учебного пособия для обучающихся по основным образовательным программам высшего образования по направлениям подготовки 24.03.04, 24.04.04 Авиастроение и специальности 24.05.07 Самолёто- и вертолётостроение

САМАРА
Издательство Самарского университета
2023

УДК 629.7.01(075)+004.9(075)
ББК Ж 2.5-05я7
К932

Рецензенты: канд. техн. наук, доц. Д. С. В д о в и н,
канд. техн. наук, доц. Я. А. Е р и с о в

Куркин, Евгений Игоревич

К932 Основы разработки программного обеспечения для автоматизированного проектирования авиационных конструкций: учебное пособие / *Е.И. Куркин, Е.А. Кишов.* − Самара: Издательство Самарского университета, 2023. – 72 с.: ил.

ISBN 978-5-7883-1965-0

Предназначено для обучающихся по направлениям подготовки 24.03.04, 24.04.04 Авиастроение и специальности 24.05.07 Самолёто- и вертолётостроение и может быть использовано для изучения дисциплин «Технология программирования на языках высокого уровня», «Математическое моделирование технических систем», «Вычислительный эксперимент в авиационной технике», «Численные методы решения инженерных задач». Представлен конспект лекций, позволяющий сформировать у студентов знания о программировании на языках высокого уровня MATLAB и APDL, используемых для автоматизации проектирования и инженерных расчетов с использованием компьютерного моделирования.

Подготовлено на кафедре конструкции и проектирования летательных аппаратов.

УДК 629.7.01(075)+004.9(075)
ББК Ж 2.5-05я7

ISBN 978-5-7883-1965-0

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цель учебного пособия – помочь студентам в изучении основ программирования на языках высокого уровня MATLAB и APDL, используемых для автоматизации инженерных расчетов.

MATLAB (англ. – *Matrix Laboratory*, рус. – матричная лаборатория) является одной из самых наиболее развитых систем программирования для решения математических и инженерных задач [1,2]. В настоящее время она дополнена многими частными приложениями, применяемыми в задачах вычислительной математики, обработки информации, конструировании электронных приборов, экономики и других разделов прикладной науки.

ANSYS Parametric Design Language – параметрический язык проектирования ANSYS – язык программирования, имеющий общие черты с языком программирования Фортран 77 [3,4,5]. Использование языка позволяет наладить взаимодействие между пользователем и системой, не предусмотренное стандартным интерфейсом, например, циклы типа «do», связки вида «если – то – иначе», скалярные, векторные и матричные операции. Все это может быть использовано для написания различных оптимизаторов, получения каких-то данных с модели, приложения нагрузок к каким-то конкретным узлам и многое другое.

В пособии рассматриваются практические и теоретические аспекты методов получения и обработки данных с помощью функций и операторов языков программирования. Учебное пособие включает в себя примеры решения некоторых нестандартных задач, которые можно решить с помощью APDL.

1 ОПИСАНИЕ ОСНОВНЫХ ВОЗМОЖНОСТЕЙ И ИНТЕРФЕЙСА СИСТЕМЫ MATLAB

1.1 Описание возможностей MATLAB

MATLAB начинает свою историю с возникновения более ранних прикладных пакетов LINPACK и EIGPACK, созданных в 1970-е годы в США, которые являются также основой для современных вычислительных пакетов, таких как MathCad, MAPLE и Mathematica. Совершенствование системы MATLAB происходило как в связи с достижениями в вычислительной математике, так и в связи с изменениями в архитектуре персональных компьютеров и развитием общесистемных средств. В процессе совершенствования MATLAB был дополнен целым рядом приложений (toolboxes), которые раздвинули границы его применимости. Они очень важны для большинства пользователей, так как позволяют изучать и применять специализированные методы.

К основным достоинствам MATLAB относятся:

1. Применение языка высокого уровня для научных и инженерных вычислений.

2. Интерактивный интерфейс для проектирования и решения задач.

3. Наличие огромной совокупности инструментов для визуализации данных и создания пользовательских графиков.

4. Дополнительные приложения, которые позволяют пользователю эффективно анализировать и обрабатывать данные, проводить приближения и аппроксимации и решать широкий круг инженерно-технических и научных задач.

5. Наличие инструментов для создания приложений с пользовательским интерфейсом.

6. Возможность применений интерфейсов для C/C++, Java®, .NET, Python® , SQL, Hadoop® и Microsoft® Excel®.

7. Универсальность программы благодаря наличию версий пакета для большинства существующих на данный момент оперативных систем: Unix, Windows, Mac OS X y GNU/Linux.

Согласно мнению экспертов, наиболее важной особенностью MATLAB является его способность к развитию. Это позволяет пользователю создавать свои собственные вычислительные программы и свои собственные приложения. Наиболее важными особенностями MATLAB являются:

1. Написание программы на языке, близком к языку математики.

2. Применение матриц в качестве базового элемента языка, что обеспечивает значительное снижение ресурсов.

3. Реализация комплексной арифметики.

4. Возможность расширения и адаптации языка через файлы script и *.m.

1.2 Основные характеристики интерфейса

Запуск программы MATLAB осуществляется с помощью главного меню Windows или ярлыка, находящегося на рабочем столе. Интерфейс MATLAB многооконный и имеет ряд средств прямого доступа к различным компонентам системы (рисунок 1.1).

По умолчанию интерфейс включает в себя три окна: окно Command Window (Командное окно) в правой части Рабочего стола, окна Current Folder (Текущий каталог) и Workspace (Рабочая область) в верхней левой части и окно Command History (История команд) в нижней левой части. Они обеспечивают оперативный контроль за состоянием системы. Выводимые на экран окна интерфейса MATLAB могут быть включены или отключены из пункта меню View. Вся работа организуется через командное окно (Command Window), которое появляется при запуске программы. В процессе работы данные располагаются в памяти (Workspace) в

виде матриц. Для просмотра значения любой переменной из текущего рабочего пространства системы достаточно набрать ее имя и нажать клавишу Enter.

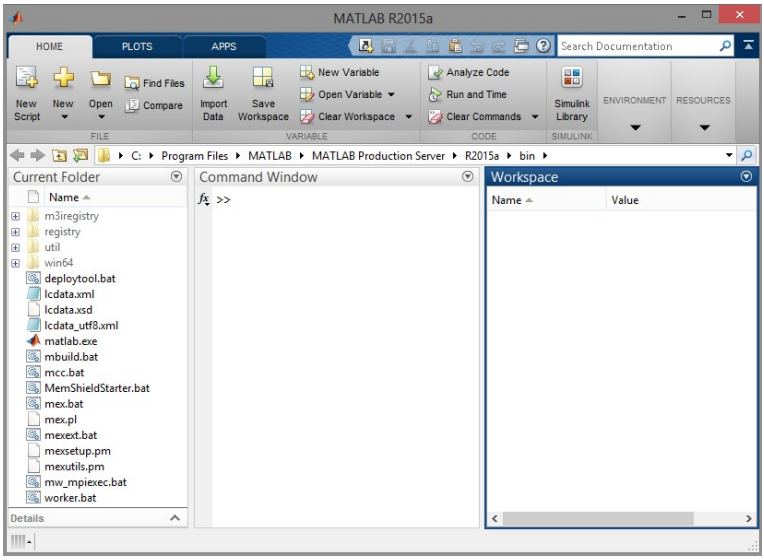

Рисунок 1.1 – Главный интерфейс пакета MATLAB

Следует обратить внимание на следующие кнопки панели инструментов:

1. **New M-file** − выводит пустое окно редактора m-файлов.
2. **Open file** − открывает окно для загрузки файлов MATLAB.
3. **Simulink** − открывает окно браузера библиотек Simulink.
4. **Help** − открывает окно справки.

Эти функции дублируются в очень простом меню системы MATLAB.

Пользователю доступно работать непосредственно в Command Windows, при этом каждая строка является определенным оператором, который автоматически выполняется по нажатию Enter. Такой подход рекомендуется, когда требуется несложное

вычисление. Другой подход состоит в том, что пользователь сначала пишет код в новом файле Script и запускает его с помощью команды Run.

1.3 Виды переменных и арифметические операции

Решение любой задачи или создание программы, как правило, начинается с задания переменных и способа представления данных. Следовательно, чтобы правильно организовать описание данных программы, необходимо знать, как задавать переменные в MATLAB и какие виды переменных возможны.

Самый простой и наиболее распространенный тип данных – это число. В MATLAB число хранится в переменной, которая имеет определенное имя, например: $a = 2$ (задает переменную с именем a и присваивает ей значение 2). По умолчанию переменная является вещественной (тип double), т.е. может принимать дробные значения, например: $a = -6.9$ (задает значение переменной a равное -6,9). Изменить тип переменной можно, указав тип числа с помощью соответствующего ключевого слова, например: $a = \text{int}15(3)$ (выполняет присваивание числа 3 как целочисленного 15-битового значения).

Обращаем внимание на то, что при задании переменных должно соблюдаться правило определения имен, которое состоит в следующем. Имена переменных могут задаваться только латинскими буквами, цифрами и символом ‘_’. Причем первый символ в имени должен соответствовать букве латинского алфавита. Также существуют имена переменных, которые уже зарезервированы для определенных случаев, например: ans – результат последней операции; eps – машинная точность; pi – число π; end – наибольшее значение индекса размерности массива и т.д.

Типы данных, доступных в MATLAB, представлены в таблице 1.1.

Таблица 1.1 – Основные типы данных в MATLAB

Вид	Описание
double	вещественный, 64 бит
single	вещественный, 32 бит
int8	знаковый целочисленный, 8 бит
int16	знаковый целочисленный, 16 бит
int32	знаковый целочисленный, 32 бит
int64	знаковый целочисленный, 64 бит
uint8	беззнаковый целочисленный, 8 бит
uint16	беззнаковый целочисленный, 16 бит
uint32	беззнаковый целочисленный, 32 бит
uint64	беззнаковый целочисленный, 64 бит

Следует отметить, что в MATLAB по умолчанию используется тип double, который имеет наибольшую точность представления вещественного числа и является потому универсальным типом. Выбор одного или другого типа зависит от желаемой точности и также от допустимых компьютерных ресурсов. Также необходимо отметить, что вид переменной относится именно к хранению данных, а для форматов вывода на экране переменных применяются команды, представленные в таблице 1.2.

Таблица 1.2 – Форматы вывода на экран

Формат	Представление
short	Число отображается с 4 цифрами после десятичной точки или в формате short e.
short e	Число в экспоненциальной форме с мантиссой из 5 цифр и показателем из 3 цифр.
rat	Представление в виде рационального дробного числа.
long	Число с 16 десятичными цифрами.
long e	Число в экспоненциальной форме с мантиссой из 16 цифр и показателем из 3 цифр.
hex	Число в шестнадцатеричной форме.

В окне Workplace существует так называемый строчный редактор, в котором с помощью применения клавиш Up и Down (стрелки курсора «Вверх» и «Вниз») пользователь может вернуться к использованию ранее введённых строк или команд.

Как и в других математических пакетах, в MATLAB существуют определённые знаки, с помощью которых указываются элементарные арифметические операции. Эти знаки представлены в таблице 1.3.

Таблица 1.3 – Главные арифметические операторы

Знак	**Операция**
+, -	Сложение, вычитание
*	Умножение
/	Деление слева направо
\	Деление справа налево
^	Возведение в степень

Использование MATLAB в режиме калькулятора может происходить путем простой записи в командную строку последовательности арифметических действий с числами (рисунок 1.2).

Стандартные функции MATLAB принимают в качестве аргумента одно или два действительных (x, y) или одно комплексное (z) число (таблица 1.4).

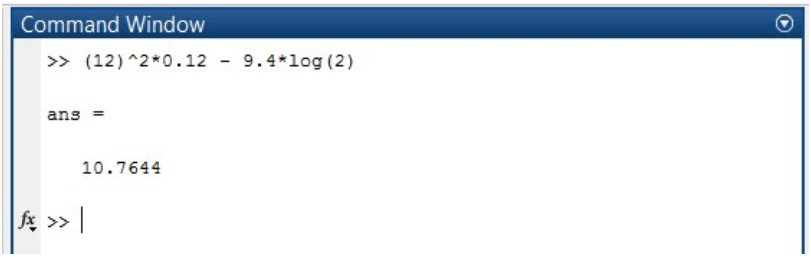

Рисунок 1.2 – Использование MATLAB в режиме калькулятора

Таблица 1.4 – Элементарные алгебраические функции

Функция	Описание
abs(*z*), **abs**(*x*),	Вычисление модуля комплексного числа *z* или абсолютного значения действительного числа *x*
angle(*z*)	Вычисление аргумента *z*
sqrt(*z*), **sqrt**(*x*)	Вычисление квадратного корня чисел *z* и *x*
real(*z*)	Вычисление действительной части комплексного числа *z*
imag(*z*)	Вычисление мнимой части комплексного числа *z*
round(*x*)	Округление до целого
fix(*x*)	Округление до ближайшего целого в сторону нуля
rem(*x*, *y*)	Вычисление остатка от деления *x* на *y*
exp(*z*)	Вычисление *e* в степени *x*
log(*z*)	Вычисление натурального логарифма числа *x*
log10(*z*)	Вычисление десятичного логарифма числа *x*

1.4 Массивы в MATLAB

Наборы чисел в программировании принято называть массивами. Всему массиву присваивается одно имя, а доступ к отдельным элементам массива осуществляется по целочисленному индексу, то есть по номеру элемента в массиве. Массивы бывают одномерными, когда используется единственный индекс (номер), а могут быть и многомерными (в частности – двумерными).

Для задания одномерного массива, состоящего из нескольких чисел (вещественных или комплексных), используется операция конкатенации с помощью квадратных скобок – [ ]. Например, следующее выражение формирует переменную с именем a1, являющуюся одномерным массивом из трёх элементов (вещественных чисел):

a1 = [1 2 3].

Объединяемые в массив элементы должны отделяться друг от друга либо пробелом, либо запятой, например:

a1 = [1, 2, 3].

Для доступа к индивидуальному элементу одномерного массива нужно после его имени указать в круглых скобках индекс (номер) этого элемента. Например, третий элемент массива a1 обозначается как a1(3), первый элемент – как a1(1), второй элемент – как a1(2).

Если требуется изменить третий элемент уже сформированного выше операцией конкатенации массива a1, то можно применить операцию присваивания:

a1(3) = 789.

Количество элементов в одномерном массиве всегда можно узнать с помощью функции length:

length(a1)

ans = 3.

Один из способов создания многомерных массивов вытекает из его представления как совокупности 2-мерных массивов, размещаемых на новых страницах (слоях). Он состоит в том, чтобы просто добавлять новые размерности для формирования нужных страниц (3-, 4-, 5-ю и т. д.).

Например:

A = [5 7 8; 0 1 9; 4 3 6].

Для облегчения работы пользователя пакет MATLAB содержит ряд основных матриц и массивов:

1) ***eye*** – создание единичной матрицы:

– *I = eye(n)* – *I* содержит квадратную единичную матрицу размера *n*;

– *I = eye(m,n)* – *I* содержит прямоугольную матрицу размера *m* на *n* с единицами на главной диагонали.

2) ***linspace*** – генерация вектора, значения элементов которого изменяются с постоянным шагом:

– *v = linspace (a, b)* – в вектор *v* заносится 100 значений от *a* до *b*;

– *v = linspace(a,b,n)* – в вектор *v* заносится *n* значений от *a* до *b*.

3) ***ones*** – создание массива, элементы которого являются единицами:

– *A = ones(n)* – *A* содержит квадратную матрицу из единиц размера *n*;

– *A = ones(m,n)* – *A* содержит прямоугольную матрицу размера *m* на *n*, состоящую из единиц;

– *A = ones(m,n,k)* – A содержит массив трех измерений размера *m* на *n* на *k*, состоящий из единиц. Допускается создание массивов большего числа измерений.

4) ***zeros*** – создание массивов, состоящих из нулей. Использование аналогично *ones*.

2 СИМВОЛЬНЫЕ И МАТРИЧНЫЕ ОПЕРАЦИИ В MATLAB

2.1 Символьные вычисления и алгебраические команды в MATLAB

Встроенный пакет символьных вычислений и операций Symbolic Math Toolbox предоставляет собой совокупность операторов для решения, задания и преобразования символических математических уравнений [6,7]. Набор инструментов включает в себя функции из общих математических областей, таких как линейная алгебра, алгебраические и обыкновенные дифференциальные уравнения и другие. Symbolic Math Toolbox может выполнять вычисления либо аналитически, либо с использованием заданной точности.

Под символьным объектом в системе MATLAB понимается переменная, предназначенная для символьных преобразований. Объявление такой переменной осуществляется функцией *sym* либо *syms*. Функция *sym* используется при объявлении какой-либо одной символьной переменной. Пример объявления переменных *x, y* представлен на рисунке 2.1.

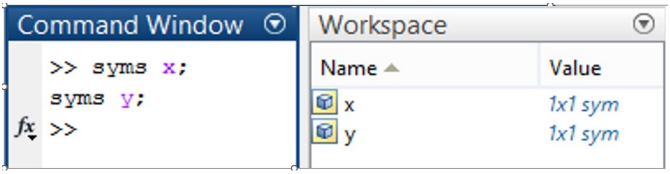

Рисунок 2.1 – Объявления символьных переменных на разных строках

При правильном объявлении переменной, она автоматически проявляется в окне Workspace в качестве сохраненной переменной. Подобным образом возможно задавать несколько переменных на одной строке (рисунок 2.2).

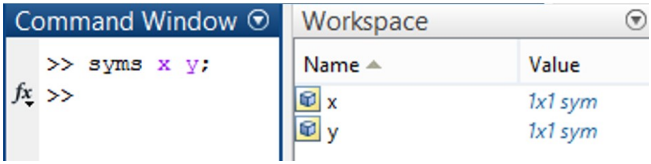

Рисунок 2.2 – Объявления символьных переменных на одной строке

Также возможно объявление переменной, принимая некоторые допущения. Например, можно предположить, что некоторая переменная только принимает целые значения, добавляя после её имени слово «integer». Для того чтобы проверить, какие допущения приняты для переменной, используется оператор *assumptions* (рисунок 2.3).

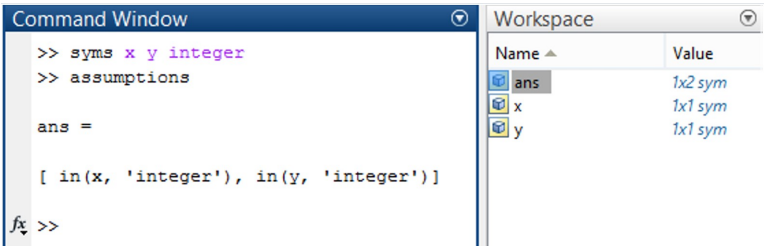

Рисунок 2.3 – Проверка принятых ранее допущений

Для снятия допущений применяются операторы *assume* и *clear* рисунок 2.4.

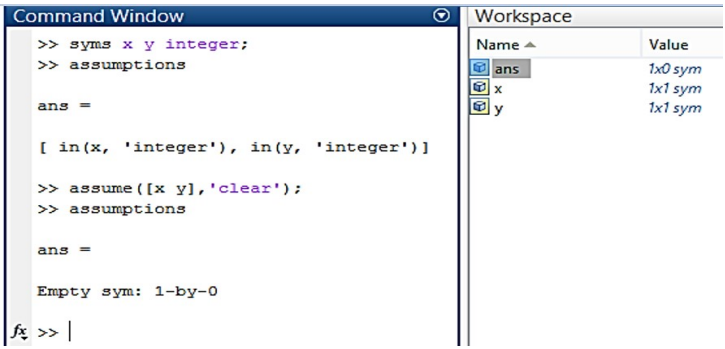

Рисунок 2.4 – Снятие раннее принятых допущений

Помимо задания переменных в MATLAB возможно объявлять функцию из символических переменных. Например:

```
>> f = x^2 + sin(y).
```

Если необходимо поставить числовые значения в символическую функцию, можно воспользоваться оператором *subs*:

```
>> subs(f,{x,y},{1,pi/2}).
```

Подобным образом можно вычислить производные (diff(f,x)), интегралы (int(f,x)) и пределы (limit(f,x,0)) в аналитической форме.

Одной из самых важных возможностей MATLAB являются преобразование, сокращение и комбинация алгебраических выражений с помощью операторов символьного вычисления пакета Symbolic Math Toolbox. Самым популярным оператором является команда *simplify*. Оператор *simplify* позволяет сокращать математические выражения. Оператор работает с различными типами символических выражений, таких как полиномы, выражения с тригонометрическими, логарифмическими, и специальными функциями.

Например, сокращение дроби $\frac{x^3-1}{x-1}$ показано на рисунке 2.5.

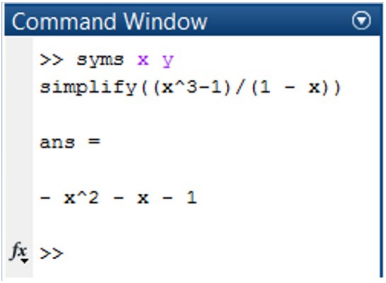

Рисунок 2.5 – Сокращение символьных выражений в MATLAB

Другие элементарные операторы преобразования символьных выражений в MATLAB представлены в таблице 2.1.

Таблица 2.1 – Элементарные операторы символьных выражений в MATLAB

Оператор	Описание
combine	Объединение членов одинаковой алгебраической структуры.
expand	Расширения выражения
factor	Разложение выражения на множители
collect	Объединение выражений одинаковой степени
children	Разделение многочлена на простейшие алгебраические структуры
rewrite	Выражение многочлена через определенные функции

2.2 Решение алгебраических уравнений

Многие математические задачи требуют решения громоздких уравнений, которые практически невозможно решить вручную. Пакет Symbolic Math Toolbox позволяет пользователю решить алгебраическое уравнение (при выполнении определенных условий и существовании решения) с помощью оператора *solve*. В простейшей форме можно записать уравнение в скобках и применить оператор *solve*. Например:

```
>> solve('x-3=0')
ans = 3
```

Однако для более сложных выражений рекомендуется написание уравнений (и при необходимости также допущений) на разных строках. Необходимо обратить внимание на то, что объявление уравнения осуществляется с помощью двойного знака равенства. Например:

```
>> syms x;
>> eqn = sin(x) == 1;
>> solx = solve(eqn,x)
solx = pi/2
```

Подобным образом можно решить уравнения с несколькими переменными, предполагая, что остальные величины являются постоянными. Например:

```
>> syms a b c x
>> sol = solve(a*x^2 + b*x + c == 0)
>>  sola = solve(a*x^2 + b*x + c == 0, a)
sol =
 -(b + (b^2 – 4*a*c)^(1/2))/(2*a)
 -(b – (b^2 – 4*a*c)^(1/2))/(2*a)
sola =
-(c + b*x)/x^2
```

В случае тригонометрических или других видов уравнений, возможны не одно, а множество решений, которые правильно удовлетворяют исходному уравнению. Для таких случаев MATLAB имеет дополнительный оператор «Return Conditions», определяющий условия, при которых данное решение является справедливым. Пример применения данного оператора представлен на рисунок 2.6.

Обычно при решении сложных математических уравнений символьный решатель не может найти точное аналитическое решение этого уравнения, тогда выдается предупреждение об этом, и программа предлагает выполнить численное решение. При использовании этой функции необходимо помнить, что, во-первых, для нелинейных алгебраических уравнений нахождение численного решения может занять много времени и, во-вторых, система показывает только первое найденного решение, в то время как остальные пренебрегаются (рисунок 2.7).

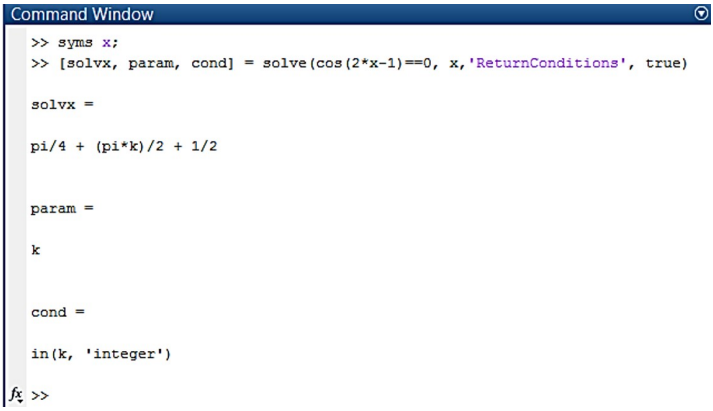

Рисунок 2.6 – Применение оператора *solve* при задании дополнительных условий

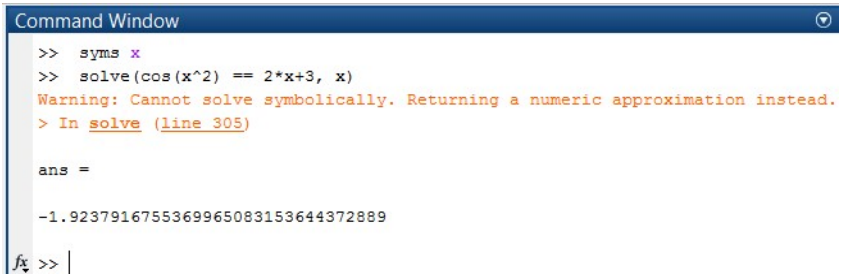

Рисунок 2.7 – Численное решение уравнения в MATLAB

Для полной оценки всех решений необходимо построить графики функций, входящих в уравнение, и дальше указать решателю, в каких интервалах следует искать решение. Например, если из графика видно, что одно из решений находится около точки -4, то можно указать:

```
>> vpasolve(200*sin(x) == x^3 – 1, x, -4)
ans = -3.000995467708643067992657292 4945
```

Как и для обычных алгебраических уравнений, пакет MATLAB способен решать системы уравнений при указании определенных условий, аналитически или численно и при других

заданных пользователем параметрах. Пример решения системы уравнений представлен на рисунке 2.8.

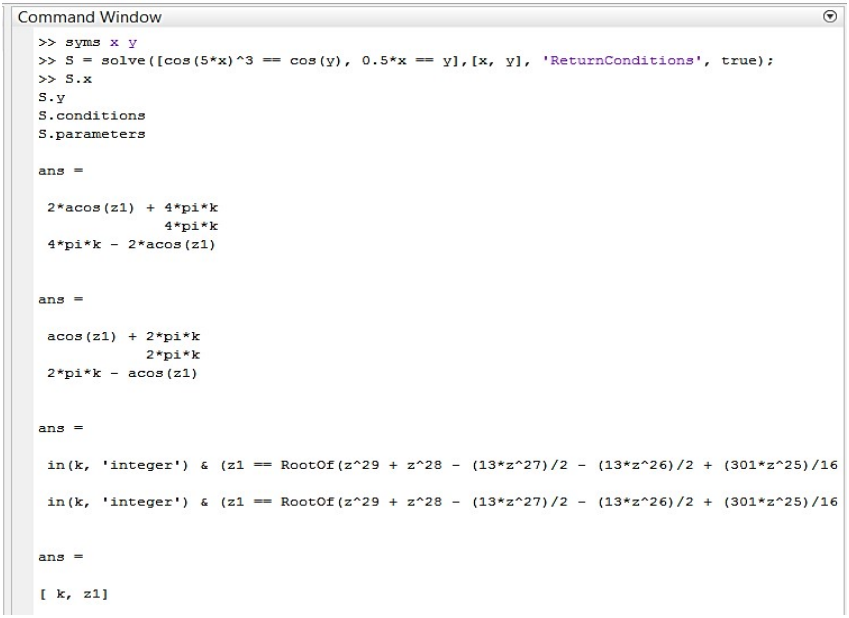

Рисунок 2.8 – Решение системы уравнений

2.3 Основные матричные операции

Основные матричные операции в MATLAB относятся к поэлементному преобразованию и матричным действиям над матрицами.

Для поэлементного преобразования матрицы справедливы все указанные ранее алгебраические функции. После применения некоторой функции формируется матрица того же размера, что и заданная, каждый элемент которой вычисляется как функция от соответствующего элемента предыдущей матрицы. Помимо этого, в MATLAB возможно вычисление поэлементного умножения матриц одинакового размера (написав команду «.*» между именами перемножаемых матриц), поэлементного деления (написав коман-

ду «./» и «.\»), поэлементного возведения в степень (команда «.^»), когда каждый элемент первой матрицы возводится в степень, равную значению соответствующего элемента второй матрицы.

Примеры применения некоторых операторов поэлементного преобразования матриц представлены на рисунке 2.9.

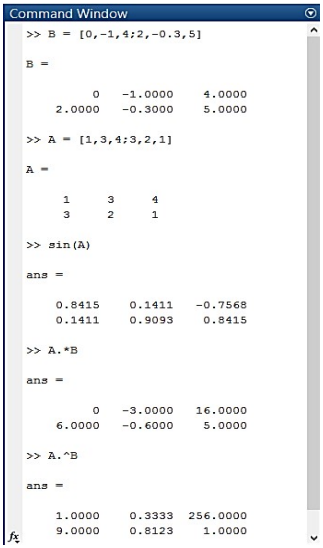

Рисунок 2.9 – Поэлементное преобразование матриц

Матричными действиями над матрицами являются такие операции, которые применяются в матричных исчислениях и не содержат неопределенностей или противоречий. Базовые действия с матрицами (сложение, вычитание, транспонирование, умножение матрицы на число, умножение матрицы на матрицу, возведение матрицы в целую степень) осуществляются с помощью обычных команд арифметических операций при выполнении следующих условий:

1. При сложении или вычитании, матрицы должны иметь одинаковые размеры.

2. При умножении матриц количество столбцов первой матрицы должно совпадать с количеством строк второй матрицы.

Примеры применения некоторых операторов матричных действий представлены на рисунке 2.10.

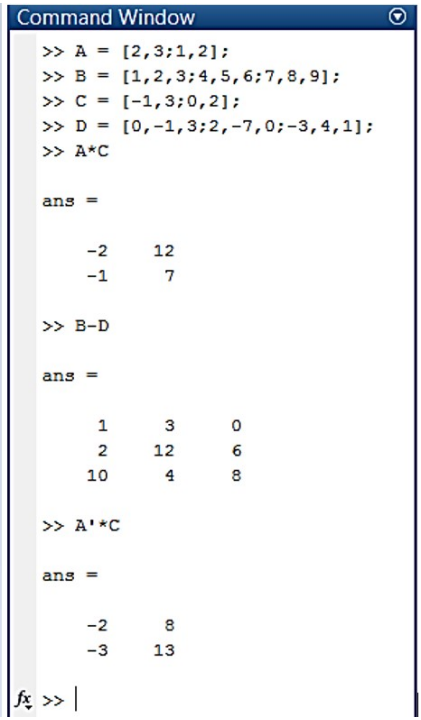

Рисунок 2.10 – Матричные действия над матрицами

Функция inv(A) вычисляет матрицу, обратную заданной матрице A. Исходная матрица A должна быть квадратной, а ее определитель не должен равняться нулю. Помимо этого, MATLAB также содержит операторы для вычисления определителя (det(A)), ранга (rank(A)) и других численных характеристик матриц. Примеры этих операций представлены на рисунке 2.11.

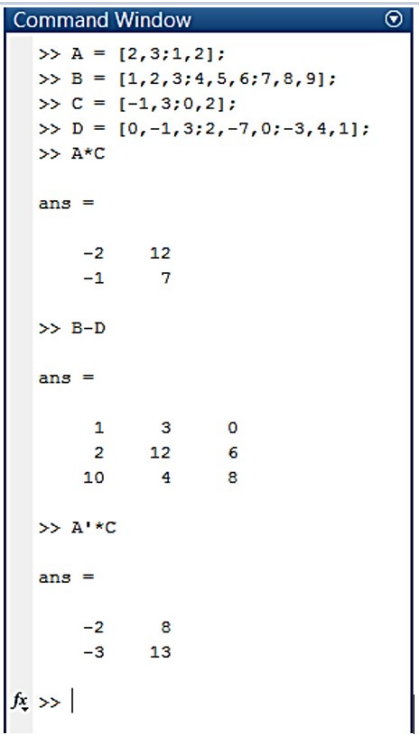

Рисунок 2.11 – Вычисление обратной матрицы
и численных характеристик матрицы

Следует отметить, что матрицы могут содержать символьные переменные – в этом случае над ними можно производить всю совокупность матричных операций в аналитической форме с помощью пакета Symbolic Math Toolbox.

3 ИМПОРТ И ЭКСПОРТ ДАННЫХ В СИСТЕМЕ MATLAB. ГРАФИЧЕСКОЕ ОТОБРАЖЕНИЕ ДАННЫХ

Первым шагом обработки данных любого типа является их импорт в некоторый вычислительный пакет. Иногда из-за большого количества существующих форматов, методов и инструментов, проблема импорта данных может вызвать определенные затруднения. Эта лекция посвящена описанию самых простых методов импорта и экспорта файлов в пакете MATLAB. Также в качестве дополнительного материала студенту предлагается ознакомиться с некоторыми способами построения двумерных графиков функций.

3.1 Импорт и экспорт файлов

Для считывания данных из файла в MATLAB существует несколько способов, которые можно разделить на две группы: с использованием функций MATLAB, или с использованием приложений с графическим интерфейсом. Например, пусть имеется текстовый файл с данными:

«Flow Time»	«cd»
1.50000e-03	1.05571e+00
3.00000e-03	6.59052e-01
4.50000e-03	3.80718e-01
6.00000e-03	1.90133e-01
7.50000e-03	7.00324e-02
9.00000e-03	7.92450e-03
1.05000e-02	-1.09808e-02
1.20000e-02	-3.10253e-03
1.35000e-02	1.68206e-02

Для импорта столбцов файла в векторы можно воспользоваться инструментом Import Data, для чего в меню Home основного окна MATLAB следует выбрать пункт Import Data и в появившем-

ся диалоговом окне открытия файла указать нужный файл. Нажимав кнопку Open, проявляется окно инструмента ImportData, в котором пользователь может выбрать: какой вид разделения столбцов нужен, количество пропущенных первых текстовых строк при считывании данных из файла и так далее (рисунок 3.1).

После нажатия на кнопку Import Selection в окне Workspace окажутся новые массивы (или новые вектора) с выбранным именем.

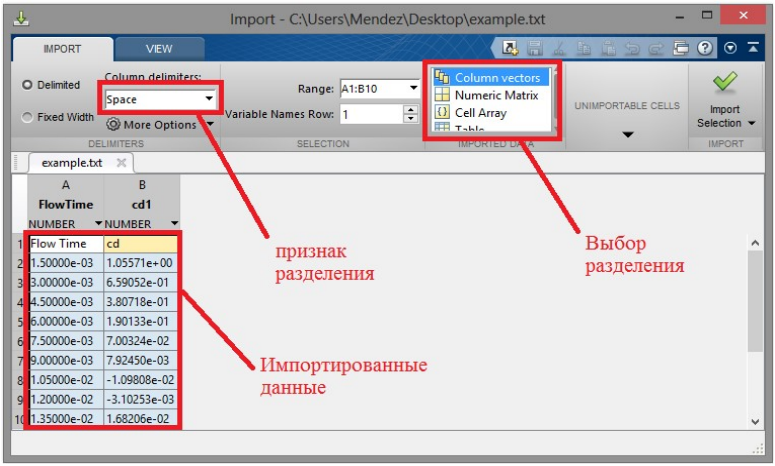

Рисунок 3.1 – Считывание файлов с помощью приложения ImportData

Если данные находятся в книге Excel, содержащей один лист, то Мастер импорта также способен создать массив, в который будут записаны эти данные. Если между столбцами данных на листе Excel есть пустые столбцы, то в MATLAB получится массив, содержащий NaN (не числа – Not a Number) в ячейках этих столбцов. Их можно убрать с помощью ручного выделения данных.

Другой метод считывания файлов в MATLAB осуществляется с помощью текстового оператора *importdata* (или csvread, dlmread, textscan, readtable, xlsread для специальных файлов). Для чтения числовых данных из текстовых файлов, имеющих табличную структуру, также подходит функция *load*. Символы-разделители (пробел, точка, точка с запятой, табуляция) интерпретируются как разделители элементов строки. Следует предостеречь пользователя от чтения файлов, имеющих заголовки, при помощи оператора load, так как MATLAB не способен интерпретировать два типа переменных (текстового и числового) из одного и того же файла. Это действие можно осуществить с помощью *importdata* – при этом возникнет новый структурированный массив, который состоит из текстовых и числовых столбцов.

При чтении текстового файла оператором *load* автоматически показываются считанные данные и в результате создается двумерный массив. Например:

```
>> A = load('example.txt')
A =
0.0015    1.0557
0.0030    0.6591
0.0045    0.3807
```

Для записи содержимого столбцов в векторы *x* и *y* можно воспользоваться простыми командами

```
>> x=A(:,1);
>> y=A(:,2)
```

Для считывания данных из Excel в MATLAB есть специальный текстовый оператор *xlsread*, который позволяет выбрать несколько способов. Необходимо отметить, что при чтении файла Excel оператор *xlsread* считывает данные только с первого листа книги. Для считывания данных из книги Excel не только с первого, а с произвольного листа, достаточно указать номер листа или его имя (в апострофах) в качестве второго входного аргумента. Например:

```
>>A=xlsread('example1.xls', 2)
```

При необходимости возможно также указание диапазона данных, которые требуется считать в качестве третьего аргумента оператора *xlsread.* Например:

```
>>A=xlsread('example1.xls',2, 'B2:C4')
```

Экспорт данных из системы MATLAB осуществляется с помощью совокупности функций, которые связаны с определенным типом файла. Выбор того или иного варианта зависит от дальнейшего использования экспортированного файла. В таблице 3.1 представлены самые популярные функции для экспортирования данных из MATLAB.

Например:

```
>>csvwrite('export.txt',A)
```

создаёт текстовый файл, который содержит массив A.

Таблица 3.1 Функции экспортирования файлов в MATLAB

Функция	Описание
save	Создает файл формате ASCII со всеми переменными, содержащимися в Workspace.
csvwrite	Создает файл с запятыми в качестве разделителей.
dlmwrite	Создает файл с числами в качестве разделителей.
xlswrite	Создает книгу Excel

3.2 Команды создания графиков в MATLAB и задания их параметров

Вывод графиков в системе MATLAB является очень простой и удобной операцией, которая осуществляется с помощью функции *plot.* Она обеспечивает построение графиков на экране компьютера. Весь графический вывод в системе MATLAB поступает в одно или несколько графических окон.

Общая форма обращения к функции такова:

```
>> plot(x1, y1, s1, x2, y2, s2,...)
```

где x1, y1 – заданные векторы, элементами которых являются массивы значений аргумента (x1) и функции (y1), отвечающие первой кривой графика;

x2, y2 – массивы значений аргумента и функции второй кривой и т.д.;

переменные s1, s2, ... являются символьными (их указание не является обязательным).

С помощью функции *plot* легко построить график функции (рисунок 3.2).

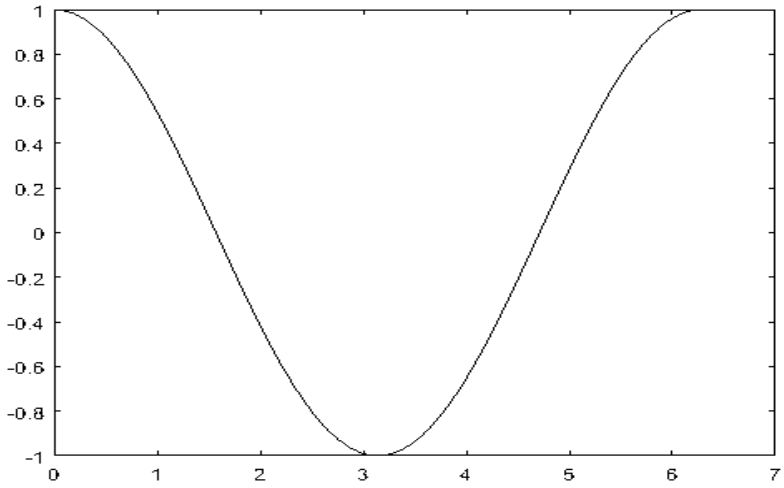

Рисунок 3.2 – Простой график, построенный с помощью *plot*

Например:

```
>> x = 0:pi/50:2*pi;
>> y = cos(x);
>> plot(x,y)
```

Функции *xlabel*, *ylabel* отвечают за добавление подписей к осям абсцисс и ординат соответственно, а *title* – определяет заголовок. Для предыдущего примера результат работы этих функции имеет вид:

```
xlabel('Time')
ylabel('cos(t)')
title('Plot of the function cosine')
```

Результат показан на рисунке 3.3.

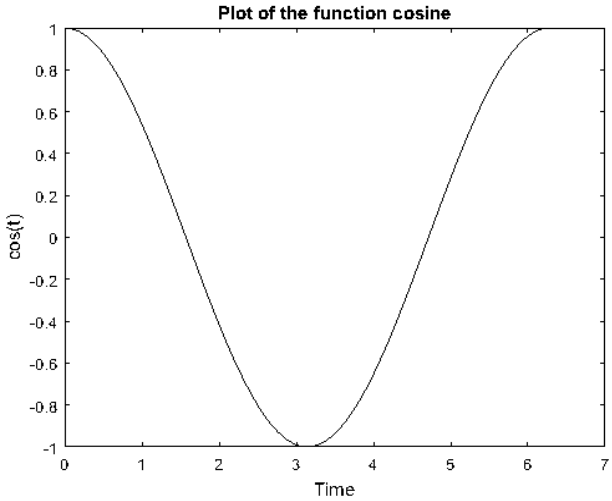

Рисунок 3.3 – Добавление заголовка и подписей к осям

Если пользователю необходимо построить несколько графиков в одной и той же системе координат, то следует использовать команду *hold on*, включающую режим сохранения предыдущего графического результата. Выйти из режима hold on можно с помощью команды hold off. Например:

```
x = 0 : pi/50 : 2*pi;
y1 = cos(x);
y2 = sin(x);
plot(x, y1)
hold on
plot(x, y2)
```

Другой способ вывода нескольких графиков в одном окне – применить функцию *plot* с несколькими парами параметров x, y. Причем, рекомендуется добавить пояснение графиков с помощью команды legend. Например, следующий код создает графики, изображенные на рисунке 3.4:

```
x = linspace(-pi,pi);
y1 = sin(x);
y2 = sin(2*x);
y3 = sin(3*x);
y4 = sin(4*x);
plot(x,y1,x,y2,x,y3,x,y4)
legend('sin(x)','sin(2x)','sin(3x)','sin(4x)',2)
```

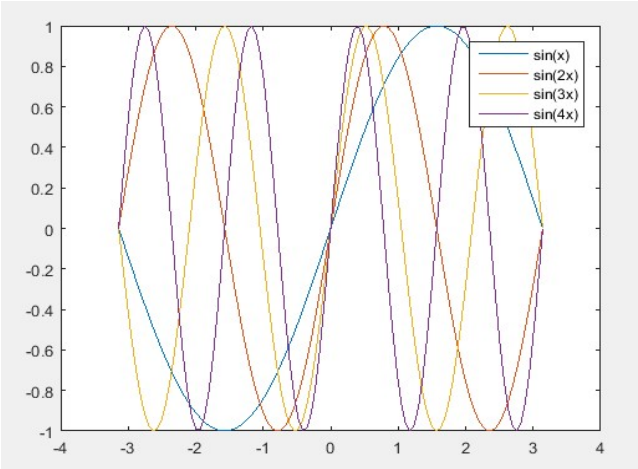

Рисунок 3.4 – Добавление легенды к графикам

Как было показано выше, аргумент *s* определяет стиль графика. Если аргумент *s* не указан, то тип линии по умолчанию – отрезок прямой, тип точки – пиксель, а цвет устанавливается в такой очередности: синий, зеленый, красный, голубой, фиолетовый, желтый, черный и белый – в зависимости от того, какая по очереди линия выводится на график. Например, обращение вида:

```
>> plot(x1, y1, s1, x2, y2,...)
```

приведет к построению графика, в котором первая кривая будет линией из отрезков прямых синего цвета, вторая кривая – такого же типа зеленой линией и т.д.

Стиль описывает от 1 до 4 символов, обозначающих цвет, стиль линии и тип маркера:

1. Цвета: **b, g, r, c, m, y, k** (которые соответствуют синему, зеленому, красному, бирюзовому, лиловому, желтому и черному цветам).

2. Стили линий: −, −−, :, − **(**которые соответствуют сплошной линии, штрих-линии, пунктирной линии, штрих-пунктирной линии и отсутствию линии);

3. Тип маркера: **+, o, *, x , s, d, ^, v, >, ^, p, h.**

На рисунке 3.5 представлены все типы маркеров, доступных для построения графиков.

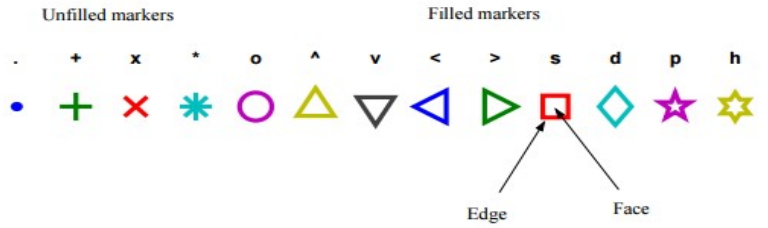

Рисунок 3.5 – Типы маркеров

Можно рассматривать несколько графиков с разными стилями. Например, графики, представленные на рисунке 3.6 строятся с помощью записи:

```
x = linspace(-pi, pi, 6);
y = sin(x);
x2 = linspace(-pi, pi, 200);
y2 = sin(x2);
plot(x, y,'-d',x2,y2,'+m')
```

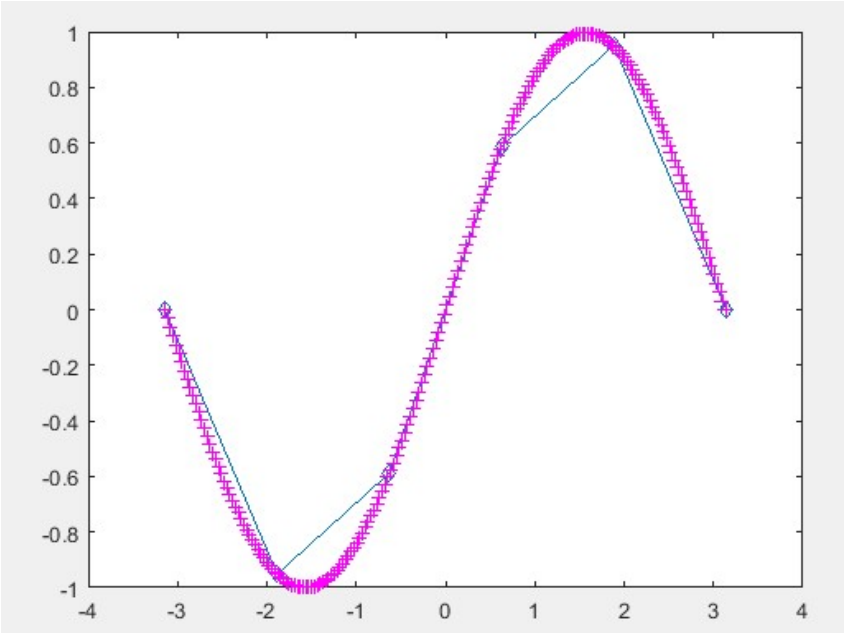

Рисунок 3.6 – Применение разных стилей графиков

Подобным образом можно построить графики параметрических функций. Например, график, представленный на рисунке 3.7, строится с помощью программы:

```
t = linspace(-10, 10);
x = sin(t).*(e.^cos(t)-2.*cos(4*t)-(sin(t/12)).^5);
y = cos(t).*(e.^cos(t)-2.*cos(4*t)-(sin(t/12)).^5);
figure
plot(x,y)
axis equal
```

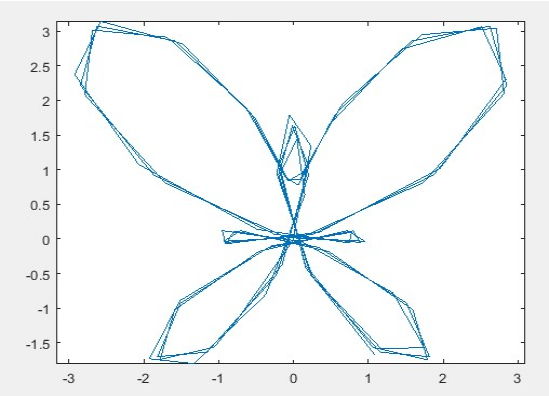

Рисунок 3.7 – Параметрический график построен в MATLAB

Для построения графиков в полярных координатах используется команда *polar*. Результат применения этой команды представлен на рисунке 3.8:

```
theta = 0:0.01:2*pi;
rho = 2.*sin(5*theta).*cos(2*theta);
polar(theta,rho)
```

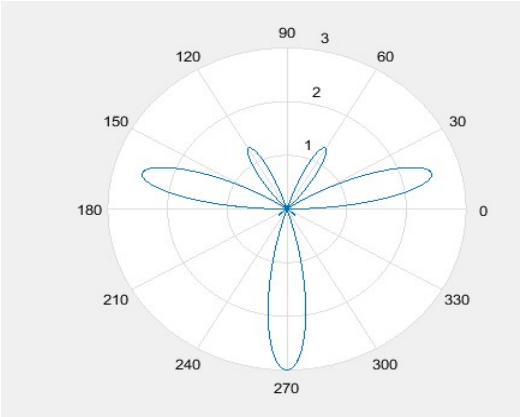

Рисунок 3.8 – График, построенный в полярных координатах

4 АНАЛИЗ И ОБРАБОТКА ДАННЫХ С ПОМОЩЬЮ СИСТЕМЫ MATLAB

При экспериментальных исследованиях часто встречается задача обработки результатов и представления некой зависимости, заданной отдельными точками, в виде гладкой функции. В этом случае целесообразно решить задачу аппроксимации исходных данных, которая сводится к определению параметров некоторой полуэмпирической модели на основе заданного множества экспериментальных данных.

Данная лекция посвящена основным методам обработки результатов с помощью системы MATLAB.

4.1 Основные операции и статическая обработка данных

Одним из простейших методов анализа данных, содержащихся в некотором массиве, является поиск его элементов с максимальным и минимальным значениями. В системе MATLAB существуют следующие функции для нахождения минимальных и максимальных элементов массива [8]:

1) max(A) − возвращает наибольший элемент, если A − вектор; или возвращает вектор-строку, содержащую максимальные элементы каждого столбца, если A − матрица;

2) max(A.B) − возвращает массив того же размера, что A и B, каждый элемент которого есть максимальный из соответствующих элементов этих массивов;

3) max(A.[ ],dim) − возвращает наибольшие элементы по столбцам или по строкам матрицы в зависимости от значения скаляра dim. Например, max(A,[ ],1) возвращает максимальные элементы каждого столбца матрицы A;

4) [C.I] =max(A) − кроме максимальных значений возвращает вектор индексов I этих элементов;

5) min(A) − возвращает минимальный элемент, если A − вектор; или возвращает вектор-строку, содержащую минимальные элементы каждого столбца, если A − матрица;

6) min(A.B) − возвращает массив того же размера, что A и B, каждый элемент которого есть минимальный из соответствующих элементов этих массивов;

7) min(A,[ ],dim) − возвращает наименьший элемент по столбцам или по строкам матрицы в зависимости от значения скаляра dim. Например, max(A,[ ],1) возвращает минимальные элементы каждого столбца матрицы A;

8) [C,I] = min(A) − кроме минимальных значений возвращает вектор индексов этих элементов.

Работа указанных функций базируется на сравнении численных значений элементов массива A, что и обеспечивает высокую скорость выполнения операций.

Кроме выше указанных функций, существуют скалярные операторы для обработки данных массива. К ним относятся:

1) SUM, CUMSUM − суммирование элементов массива,

2) PROD, CUMPROD − произведение элементов массива,

3) SORT − сортировка элементов массива по возрастанию,

4) MEDIAN − определение срединных значений (медиан) элементов массива,

5) MEAN − определение средних значений элементов массива,

6) STD − определение стандартных отклонений элементов массива,

7) COV − определение ковариационной матрицы элементов массива,

8) CORRCOEF − определение коэффициентов корреляции элементов массива.

4.2 Аппроксимация данных полиномом и с помощью приложения CurveFitting

Как и в случае импорта файлов, существует несколько способов для аппроксимации, которые можно разделить на две группы: с использованием функций MATLAB, или приложений с графическим интерфейсом (в данном случае CurveFitting).

К функциям аппроксимации данных относятся следующие операторы:

1) POLYFIT − аппроксимация данных полиномом;

2) INTERPFT − аппроксимация периодической функции на основе быстрого преобразования Фурье;

3) ICUBIC − кубическая интерполяция функции одной переменной;

4) SPLINE, PPVAL, MKPP, UNMKPP − интерполяция функции одной переменной кубическим сплайном;

5) INTERP1 − одномерная табличная интерполяция;

6) INTERP2 − двумерная табличная интерполяция;

7) GRIDDATA – двумерная табличная интерполяция на неравномерной сетке.

В качестве примера рассмотрим первый из методов. Пусть имеется массив данных, полученный экспериментально. Предположим, что зависимость можно приблизительно описать с помощью следующего полинома:

$$p(x) = p_1 x^n + p_2 x^{n-2} + p_3 x^{n-3} + \ldots p_{n-1} x + p_n .$$

Коэффициенты аппроксимирующего полинома степени n, наилучшим образом приближающего данные эксперимента в смысле наименьшего квадратичного отклонения в узлах, находятся с помощью функции:

P = polyfit (X, Y, N).

Для вычисления значений полинома можно использовать функцию:

$$Y = \text{polyval}\ (\ P, X\ ).$$

где вектор X показывает интервал табличных значений и может даже выходить за его пределы. Тем самым интерполяционный полином будет использован также для экстраполяции данных.

Например, пусть проводится эксперимент по исследованию закона Ома для нелинейного элемента (например, электронная лампа) и получены определенные данные зависимости силы тока от напряжения. После импортирования файла в MATLAB, можно написать следующий код:

```
p = polyfit(V,I,3);
p2 = polyfit(V,I,2);
p3 = polyfit(V,I,1);
y1 = polyval(p,V);
y2 = polyval(p2,V);
y3 = polyval(p3,V);
figure
plot(V,I,'o',V,y1,'-m',V,y2,':b',V,y3)
xlabel('Voltage');
ylabel('Current');
legend('Experimental data','Third-order','Second-order','First-
    order)','Location','northwest')
```

Результат показан на рисунке 4.1. Как видно из рисунка, существенным аспектом при полиномиальной аппроксимации является выбор степени полинома.

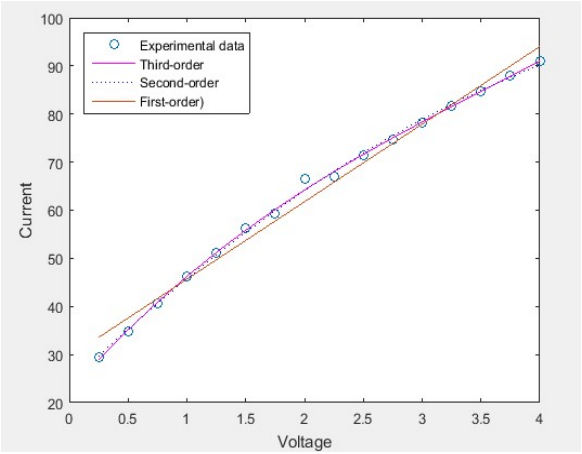

Рисунок 4.1 – Результаты аппроксимации данных полиномами разной степени

Curve Fitting Toolbox содержит наиболее широко используемые методы для подгонки кривых и поверхностей к данным, в том числе линейной и нелинейной регрессии, сплайнов и интерполяции. Все его алгоритмы могут быть реализованы через определенные функции или с помощью интерфейса приложения (рисунок 4.2).

По умолчанию интерфейс включает в себя 3 части:

1. Меню для выбора типа аппроксимирующего уравнения позволяет пользователю «подгонять» разные виды уравнения, чтобы найти наилучшее. В пакете, доступны linear (линейная), Polynomal (степенная), power (экспоненциальная), Rational (рациональная), Smoothing Spline (сплайн), Sum of sine (сумма синусов) и Weibull модели.

2. Меню выбора позволяет пользователю указать данные из ранее импортированных, для которых будет проводиться аппроксимация.

3. Окно результатов показывает полученные аппроксимацией коэффициенты уравнения.

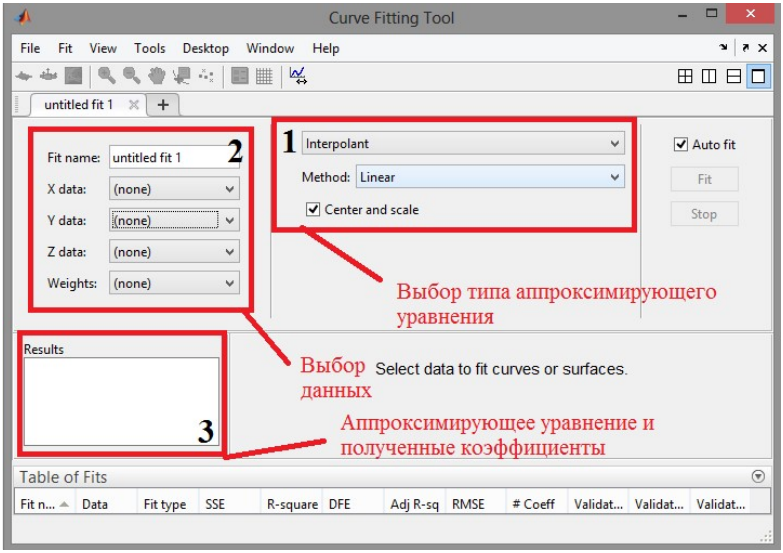

Рисунок 4.2 – Интерфейс приложения Curve Fitting

CurveFitting также имеет возможности по проведению нескольких аппроксимаций одновременно и выполнению затем их сравнения, которое осуществляется с помощью открытия нового окна (File > New Fit) и активирования опции Residuals Plot, показывающей значения разниц между данными и точками, полученными аппроксимацией.

Например, проведём с помощью CurveFitting аппроксимацию тех же самых данных исследования закона Ома для нелинейного элемента. Выбираем в качестве аппроксимирующих уравнений полином второго порядка и экспоненциальную функцию. Результаты аппроксимации показаны на рисунке 4.3.

Выбор той или иной аппроксимации можно объективно сделать с помощью статистических значений погрешностей, которые показаны в окне результатов. Описание этих статистических параметров выходит за рамки данного пособия.

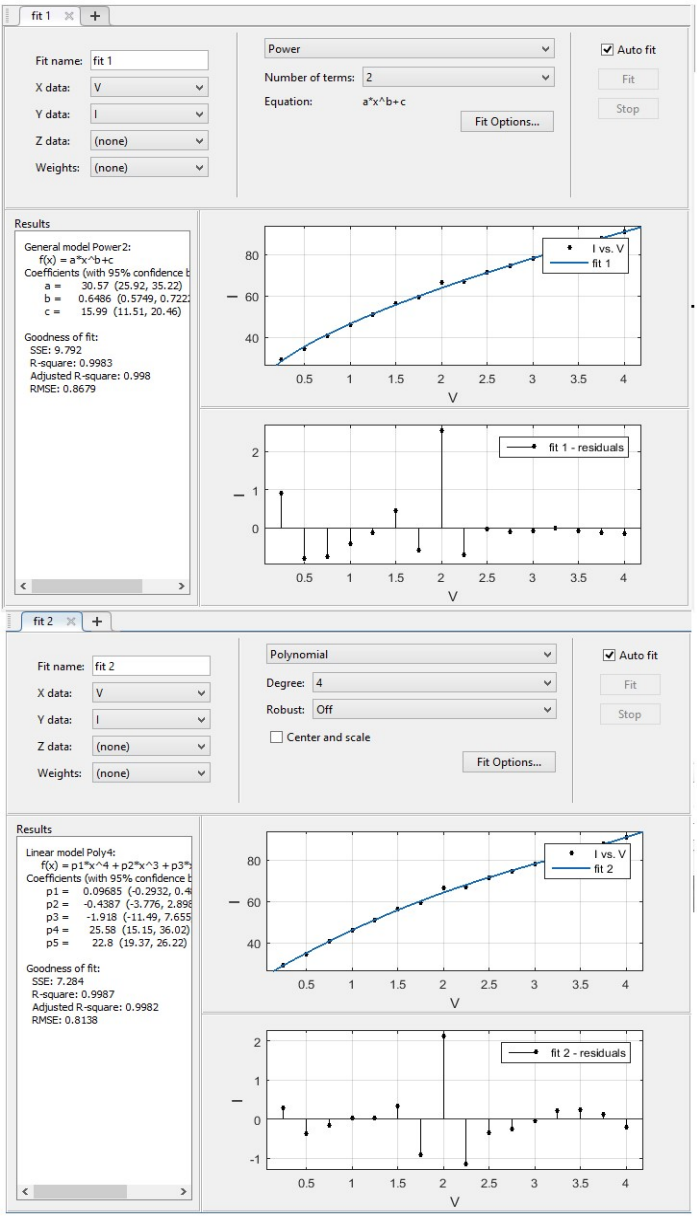

Рисунок 4.3 – Аппроксимация данных с помощью Curve Fittting

5 ОПИСАНИЕ ОСНОВНЫХ ВОЗМОЖНОСТЕЙ И ПРОСТЕЙШИХ ОПЕРАЦИЙ ANSYS APDL

5.1 Описание возможностей ANSYS APDL

По многим аспектам моделирования решения в ANSYS Workbench система принимает автоматически и для «ручного» управления процессом используется, как правило, APDL [9,10,11]. Иногда бывает необходимо получить какие-либо величины при решении задачи для каждого элемента или внести изменения в свойства материала в ходе решения задачи. Стандартными средствами это сделать невозможно либо очень сложно. Однако это необходимо при написании оптимизаторов или при необходимости задать какой-либо сложный случай нагружения.

Входной файл для решателя ANSYS пишется на APDL, в чем можно убедиться, если открыть файл ds.dat, который находится в директории решателя (папка MECH). В нем полностью описаны все шаги работы решателя: описание узлов и элементов, граничные условия, настройки расчёта и требуемые результаты расчёта.

5.2 Основные расчетные процессоры ANSYS

Для удобства пользования ANSYS имеет графический интерфейс пользователя, предоставляющий быстрый доступ к различным функциям, командам, а также к обширной HELP-системе.

Работа программы ANSYS организована в двух уровнях:

– начальный уровень (Begin level);

– процессорный уровень.

Работа программы ANSYS начинается с начального уровня (Begin level). На этом уровне доступны команды работы с файлами (сохранение, удаление, переименование и т.д.).

На втором уровне доступны несколько процессоров. Каждый из них предоставляет доступ к различным функциям и командам.

Список наиболее часто используемых процессоров и задач, с помощью них решаемых, приведен ниже.

К основным процессорам относятся:

1. Препроцессор /prep7 – в препроцессоре создается геометрическая и расчетная модель, прикладываются нагрузки. Здесь же можно изменить некоторые параметры исследуемой модели (предусмотренные типом элемента) во время расчета, такие как плотность, объём, толщина и другие параметры элементов.

2. Решатель /solution – в нем указывают опции расчета и проводится решение.

3. Постпроцессор /post1 – основной постпроцессор, в котором можно узнать результаты анализа всей модели или ее частей за один шаг нагружения (если их несколько). В этом модуле можно получить различные таблицы, в которых записана вся информация о модели во время анализа. К этим данным можно отнести объем элементов, различные напряжения, деформации и многое другое.

4. Постпроцессор /post26 – «временной» постпроцессор, в котором можно создавать различные графики, отображающие результаты анализа в зависимости от времени или друг от друга.

Всего процессоров около 10, однако не все они задокументированы.

Важно учитывать, что большая часть APDL-команд работает только в одном процессоре, за рядом нескольких исключений. Так, применив команду, относящуюся к /prep7, в/ post1, получим ошибку (рисунок 5.1).

Здесь видно, что программа выполнила расчет, но команды могли быть проигнорированы, как не относящиеся к prep7. На практике такое сообщение не всегда обозначает, что программа не работает или работает неверно. Как правило, warning (предупреждение) лишь показывает, что где-то возможна ошибка. Поэтому после проведения расчета полезно проверить лог решателя (отчёт).

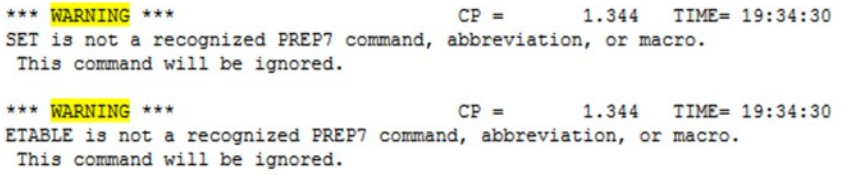

Рисунок 5.1 – Сообщение об ошибке

5.3 Пользовательские переменные и арифметические операции

APDL позволяет использовать стандартные арифметические действия (сложение, вычитание, умножение и деление) и создавать переменные и константы (не путать с реальными константами элементов) без каких либо предварительных действий. Данные переменные не относятся к какому-то конкретному типу расчетных модулей и могут задавятся в любой части кода.

Это выглядит примерно следующим образом:

B=1
C=2
K=B+C

Также существует команда Set, позволяющая задавать значения конкретным переменным. Команда Set имеет следующий синтаксис:

*Set,par,value

где *set – объявление о начале команды;

Par – имя переменной, для которой задается значение;

Value – значение или цепочка значений.

Пример:

*set,a,1

Здесь переменной a присваивается значение 1 (аналог a =1).

Также, если a – элемент массива, то команда set может присвоить значения его элементам следующим образом:

*set,a(1),1,3

Получим, что a(1) = 1, a(2) = 3.

5.4 Массивы в APDL. Типы массивов

ANSYS позволяет использовать несколько типов массивов [12,13]. В них можно хранить самую различную информацию, но чаще всего в них необходимо хранить различные параметры, соответствующие элементу, такие как напряжения, плотности и другие.

Для использования массива его необходимо объявить с помощью команды *dim.

Рассмотрим команду подробнее:

*dim, Par, Type, IMAX, JMAX, KMAX,

где *dim – объявляет о начале массива;

Par – переменная массива;

Type – тип массива;

IMAX, JMAX, KMAX – отвечают за размерность массива.

Одномерный массив a состоящий из ne-элементов: *dim,a,array,ne.

Рассмотрим типы массивов подробнее:

1. Array – обычный 1-, 2- или 3-мерный массив. Его размерность зависит от параметров: IMAX, JMAX, KMAX. По умолчанию JMAX= KMAX = 1, что соответствует одномерному массиву.

2. Arr4, Arr5 – то же, что и Array, но 4- и 5-мерные соответственно.

3. Char – символьный массив.

4. TAB4, TAB5 – тоже, что и Arr4, Arr5.

5. Table – табличный массив, используется для заполнения таблиц какими-либо значениями, извлеченными из элемента. Данная команда позволяет заполнять 1-, 2- или 3-мерные таблицы.

6. String – строковый массив, может быть заполнен только по IMAX.

5.5 Вывод результатов в файл, созданный пользователем

Иногда бывает необходимо получить какие-либо параметры, взятые с исследуемой модели, но это невозможно сделать стандартными средствами – например, нам может понадобиться оценить массу каждого конкретного элемента, рассчитать и выдать какой-либо коэффициент.

Для вывода в файл требуется первым делом создать его:

```
*cfopen,C:\ .....\имя_файла,dat
```

Вместо многоточия указывается путь к файлу. Проще всего это сделать следующим образом – создать файл с нужным именем в нужной папке, после чего скопировать путь к нему из Проводника. Файл для записи создается автоматически, так что создавать его заранее не обязательно. Для записи переменой в файл нужно записать команду *vwrite и переменную:

```
*vwrite,a
('a=',e12.4)
```

В скобках ниже в кавычках указан текст, а через запятую – форма вывода (в данном случае e12 обозначает экспоненциальную форму вывода, 4 – число знаков после запятой). Также этой командой можно записывать просто текст в файл. В конце программы необходимо закрыть файл командой *cfclos.

6 СБОР ДАННЫХ О МОДЕЛИ, ИСПОЛЬЗУЕМОЙ В ЗАДАЧЕ. ЦИКЛЫ. КОМАНДА ETABLE

6.1 Сбор данных о модели, используемой в задаче. Команды *get и *vget

При решении задачи нам необходимы данные для анализа. Некоторые из этих данных можно получить напрямую средствами, предоставляемыми модулем. Но при решении какой-либо нестандартной задачи очень часто возникает ситуация, когда стандартных средств бывает недостаточно. Тогда можно воспользоваться некоторыми из рассмотренных ниже команд.

Первые из этих команд – команды *get и *vget.

Это основные команды, с помощью которых можно получать самую разную информацию о задаче.

Эти команды очень похожи, разница состоит в том, что *vget извлекает значение и записывает их в массив параметров, а * get записывает его в качестве скалярного параметра.

Рассмотрим пример использования:

```
*vget,el,elem,,elist,,
*get,n,parm,el,dim,1
```

С помощью этих команд мы получили номера всех элементов массива, что необходимо для дальнейших расчетов, так как данные чаще всего извлекаются поэлементно, для чего нам требуется знать количество элементов.

Фактически, *vget записывает номера всех элементов в массив el, а *get определяет количество элементов в массиве el и записывает его в переменную n.

Другой способ воспользоваться напрямую командой get:

```
*get,n,elem,0,count
```

Здесь в n записывается число элементов в выбранном наборе.

Примечание: существуют и другие способы определения числа элементов.

Структура команд очень похожа, поэтому рассмотрим только *get:

*Get, Par, Entity, ENTNUM, Item1,

где Par – переменная, которой будет присваиваться значение извлекаемого параметра;

Entity – ключ, который определяет, что мы ищем. Это может быть элемент, узел и много другое, полный список с описанием можно найти в Help. Этот же параметр определяет, к какому процессору относится команда.

Рассмотрим несколько ключей подробнее:

1) Elem – ключ, позволяющий получить информацию о элементе. Здесь есть площадь элемента, номер материала, сечения и др.;

2) Shell – отвечает за листовые тела, здесь есть различные параметры, вроде жесткостей, толщин и прочее;

3) Node – отвечает за узлы (их номера, приложенные к ним силы и закрепления);

4) Mat – отвечает за используемый материал;

5) ENTNUM – номер элемента, в котором извлекается параметр;

6) Item1 – пометка к извлекаемому параметру, может отсутствовать, в зависимости от типа извлекаемого параметра. Он зависит от параметра Entity.

Рассмотрим несколько примеров применения *get. Данной командой можно получить значение площади в элементе elk:

*get,a,elem,elk,area

Данной командой можно получить значение объёма в элементе elk:

*get,v,elem,elk,volu

Не все данные можно собрать с помощью команды *get напрямую. Так, например, для получения напряжений с элементов необходимо предварительно создать таблицу элементных результатов (etable) – и уже с неё считывать данные командой *get.

Эта работа выполняется в постпроцессоре /post1.

6.2 Команда *etable

Как было написано выше, данная команда относится к постпроцессору. Поэтому для ее использования необходимо его открыть и закрыть после использования:

/post1
……
Finish

Рассмотрим структуру команды etable: Etable, Lab, Item, Comp, Option.

Подробнее о командах:

1) Etable – объявляет о выполнении команды;

2) Lab – таблицы с элементом (подробнее чуть ниже);

3) Item – основной ключ (фактический аналог enitity);

4) Comp – компонент item, зависит от него (подробнее чуть ниже);

5) Option – опции таблицы. Существуют 3 опции: AVG (средний) – значение по умолчанию; Min, Max – хранит минимальное или максимальное значение. На практике чаще всего писать не нужно, так как значения по умолчанию обычно бывает достаточно.

Рассмотрим подробнее параметр Item:

1) PRES – параметр, показывающий давление. Не имеет компонента.

2) S – выдаст напряжения, причем обладает несколькими компонентами, которые позволяют оценить различные виды на-

пряжений или их части. Например, comp = XY позволит оценить напряжения именно в плоскости XY.

Вообще существует более 30 различных значений Item, полный список которых можно найти в Help.

Рассмотрим пример использования:

```
/post1
 set,vol  – имя переменной таблицы
    etable,seqv_%vol%,volu
finish
```

Эта команда записывает в таблицу seqv_%vol% все значения объемов выбранной области (о выбранных областях чуть ниже).

Теперь их можно использовать с помощью команды get.

Важно – выше был пример получения объёма элементов напрямую с помощью *get. Разницы в данном случае нет, оба способа верные. Но следует учесть, что в постпроцессоре будет получен объем элемента после деформаций, а в препроцессоре – до. Для некоторых задач, например, оптимизации, это принципиально, так как он может меняться в результате нагрузок.

6.3 Циклы

ANSYS позволяет использовать цикл do и связку if-then. Цикл do относится к командам, которые работают в нескольких процессорах (по крайней мере, в /prep7 и /post1).

Сам цикл выполняет простую функцию – выполняет какое-либо действие определённое число раз.

Так как большую часть данных придется получать поэлементно, то без использования циклов обойтись очень сложно. Так же цикл

может быть использован для присвоения элементам определенных свойств. Вообще говоря, циклы – достаточно мощный инструмент, позволяющий решать множество самых разнообразных задач.

Рассмотрим реализацию циклов в APDL:

Чтобы объявить цикл, необходимо написать следующую команду:

*DO, Par, IVAL, FVAL, INC

Пояснения:

1) *DO – объявляет о начале цикла;

2) Par – перебираемая переменная, она задается пользователем исходя из задачи;

3) IVAL – отмечает, с какого числа мы начинаем перебирать Par (чаще всего это 1);

4) FVAL – отмечает число, по достижении которого цикл прекращается (чаще всего это номер последнего элемента или узла);

5) INC – команда, увеличивающая Par (по умолчанию на 1). Можно не указывать, если устраивает увеличение по умолчанию на 1.

Завершается цикл командой:

*ENDDO

Рассмотрим пример использования цикла для получения объёма всех элементов тела:

```
/prep7
*do,k,1,n
elk=el(k)
*get,V,elem,elk,volu
*enddo
Finish
```

где n – количество элементов в модели.

Получить распределение объёмов можно и стандартными средствами ANSYS в графическом виде, что гораздо удобнее. Но для решения нестандартных задач, это необходимо.

6.4 Связка IF-THEN

ANSYS позволяет использовать очень полезный инструмент – связку IF-THEN, которая позволяет выполнять проверку различных условий. Особенно эта связка хороша при использовании совместно с циклами

Механизм действия очень простой – если значение параметра удовлетворяет определенному условию, то будет выполнено одно действие, если же нет, то другое.

Рассмотрим команду, позволяющую использовать IF:

*IF, VAL1, Oper1, VAL2, Base1, VAL3, Oper2, VAL4, Base2

где *IF ─ объявляет начало команды;

Val1─ переменная, которая сравнивается с Val2;

Oper1 ─ операция сравнения.

Рассмотрим подробнее.

Вместо oper1 записывается одна из приведенных ниже команд:

EQ – =
NE – ≠
LT – <
GT – >
LE – ≤
GE – ≥

Base1 – описывает действие, предпринимаемое в случае, если команда выполняется. Важно учесть, что для Base1 – это всегда then, для последующих (если они есть) возможен вариант else.

Рассмотрим простой пример применения If:

```
*if, a,gt,b, then
min=b
*elseif,a,lt,b,then
min=a
```

Здесь определяется наименьшее из чисел a и b. Пример достаточно примитивный, но с его помощью хорошо видна структура связки. Важно учесть, что *elseif без *if работать не будет.

7 ТИПЫ ЭЛЕМЕНТОВ, ИСПОЛЬЗУЕМЫЕ В ANSYS, И ИХ СВОЙСТВА

ANSYS использует для решения задач механики метод конечных элементов [14-20], состоящий в разбиении исследуемого объекта на большое число элементов конечного (не бесконечно малого) объёма. Рассмотрим эти элементы подробнее.

ANSYS поддерживает более 200 типов конечных элементов, которые позволяют моделировать различные объекты. При этом разные классы элементов имеют различные подтипы, позволяющие наиболее точно смоделировать исследуемый объект.

Также следует отметить, что элементы одного класса, могут иметь совершенно разные свойства (наиболее «наглядными» отличиями являются различные степени свободы и число узлов в элементе).

В модуле Workbench тип элемента назначается автоматически, программа сама подбирает наиболее подходящий (с ее точки зрения), что иногда затрудняет решение задачи. Чуть позже мы рассмотрим, как задать тип элемента вручную, если по каким-то причинам не подходит элемент, назначенный автоматически.

Важно: прежде, чем менять тип элемента, необходимо подробно изучить его свойства. Так же следует помнить, что в новых версиях программы добавляют новые типы элементов, заменяющие старые. При этом поддержка старых элементов, как правило, остается, однако они могут не документироваться. В этом случае можно попытаться найти описание элемента в специальной литературе (как правило, с пакетом Ansys предоставляется достаточно много документации, в которой можно найти практически любую информацию о пакете и работе в нем.)

7.1 Основные типы элементов. Присвоение определенного типа элемента. Команда ET

Итак, рассмотрим несколько классов элементов, которые могут пригодиться для дальнейшей работы. К этим классам элементов можно отнести твердотельные (Solid), оболочечные (shell), массовые (mass) и балочные (beam).

Твердотельными элементами можно моделировать практически любые тела, толстостенные конструкции.

Оболочечными телами обычно моделируются различные тонкостенные конструкции, например обшивка самолета.

Массовые элементы позволяют создавать точечные массы, которые могут моделировать различные тяжелые, но небольшие предметы либо тела, имеющие массу, но которые по каким-либо причинам нельзя или неудобно отображать твердотельной моделью. Например, ими можно задать нагрузку, сообщаемую двигателем на крыло самолета при предварительных расчетах.

Балочные элементы позволят смоделировать различные силовые элементы, стержни и прочее.

Рассмотрим несколько видов твердотельных элементов. Основным элементом (который чаще всего выбирается по умолчанию для твердотельных моделей), является solid186. Solid 186 – 20-узловой конечный 3D-элемент, имеющий вид, представленный на рисунке 7.1.

Буквы от A до J – узлы элемента. Данный элемент может принимать следующие формы (рисунок 7.2).

Для этого элемента можно назначить множество различных свойств, от плотности и типа материала до температуры в каждом отдельном узле. Этот элемент можно считать одним из самых гибких в плане настроек и свойств (поэтому он часто назначается по умолчанию).

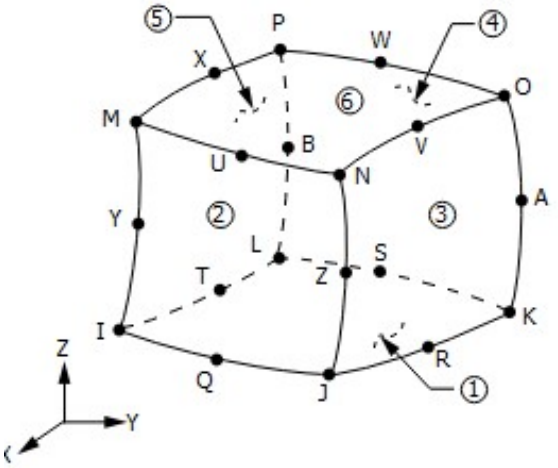

Рисунок 7.1 – Элемент Solid186

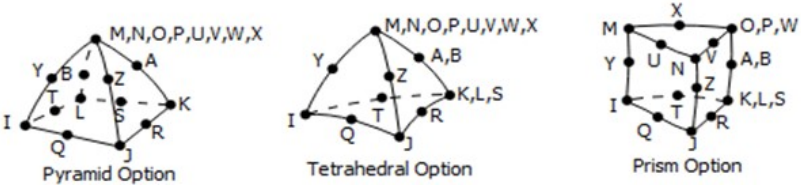

Рисунок 7.2 – Различные формы объёмного элемента

Элемент Solid236 – так же обладает теми же 20 узлами, что и Solid186, однако он может моделировать электромагнитные и магнитные поля, соответственно, обладает магнитными и электрическими степенями свободы.

Рассмотрим оболочечные элементы. Shell181 ─ основной оболочечный элемент, обычно назначаемый по умолчанию. Он обладает 4-мя узлами (I,J,K,L), расположенными по углам (рисунок 7.3).

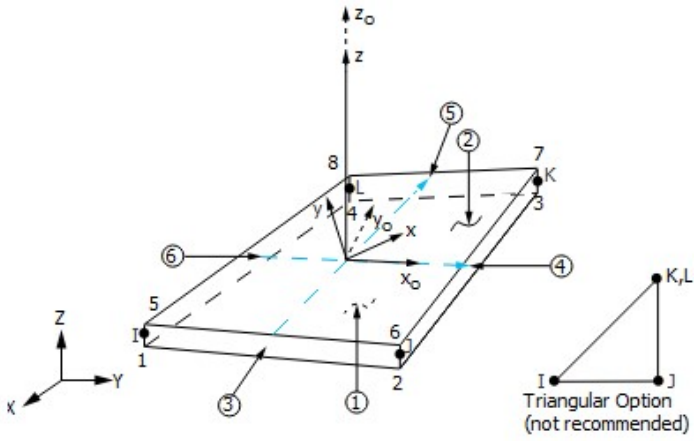

Рисунок 7.3 – Элемент Shell181

Элемент может выражаться в треугольник, в этом случае узлы K и L совпадают.

Узел I можно назвать основным (так как толщина, заданная в нем, определяет толщину элемента (если не указаны толщины других узлов).

Свойства, которые можно назначить те же, что и для solid (к ним относятся свойства материала).

Балочные элементы Beam161, Beam188, Beam189 могут быть использованы для создания стержней. Отличаются друг от друга количеством узлов и степенями свободы.

Элемент MASS21 – массовый элемент, который является фактически точкой.

Для того, чтобы назначить тип элемента вручную используется команда ET.

Рассмотрим ее структуру:

ET, ITYPE, Ename, KOP1

где ET – объявляет о команде;

ITYPE – номер типа элемента, задающийся пользователем (может быть использован для дальнейших ссылок на него). Нумерация с начинается с 1 (по умолчанию);

Ename – имя типа элемента (например, SHELL181);

KOP1 – KEYOPT. Обозначает свойства элемента. Для каждого элемента имеется свой набор свойств. Всего может быть до 6 свойств KEYOPT. Если не изменять их, то в команде их писать не нужно.

Пример:

ET,1,Shell181

7.2 Определение свойств элемента. Команды R, MP

Все элементы имеют определенные свойства. Часть их определяется свойствами материала (плотность, модуль упругости), а часть – при помощи так называемых реальных констант, отвечающих за различные геометрические характеристики.

Рассмотрим команды, позволяющие задать эти свойства.

Первыми рассмотрим свойства материла. За это отвечает команда MP.

Рассмотрим ее структуру:

MP, Lab, MAT, C0

где MP – объявляет команду;

Lab – название присваиваемого свойства;

MAT – отвечает за связь номера материал с элементом. По умолчанию пишется MAT (если используется 1 материал, если больше, то номер элемента, заданный ET);

C0 – задаваемое значение. Можно указать просто число (важно соблюсти размерность), можно переменную, вычисленную заранее.

Пример применения (присваивает плотность элементу типа 1):

ET,1,Shell181
mp,dens,1,0.284

Lab для разных типов элементов свой, полный список можно найти в описании материала. Вообще, можно задать модули упругости, различные температурные коэффициенты и много другое.

Другой способ редактирования – использование реальных констант.

Это достаточно специфический термин Ansys, обозначающий какую-либо характеристику элемента. При этом важно помнить, что не у всех элементов есть реальные константы. Например, solid186 не имеет реальных констант. Узнать, если ли у элемента реальные константы, можно из его описания.

Команда работает очень просто:

R, NSET, R1

где R – объявляет команду;

Nset – номер реальной константы;

R1– значение.

Данная команда чаще всего относится к балочным элементам и некоторым устаревшим оболочечным, поэтому применять ее следует с осторожностью.

7.3 Изменение свойств элемента. Команды Emodif, Secdata и Sectype

Иногда нам будет требоваться изменить свойства элемента или материала прямо во время задачи, что особенно важно при оптимизации различных конструкций. Осуществить это можно с помощью команды Emodif.

Команда работает следующим образом:

EMODIF, IEL, STLOC, I1

где IEL – номер модифицируемого элемента или узла;

STLOC –модифицируемый параметр. Это может быть MAT, TYPE, REAL, или SECNUM, то есть материал, тип элемента, реальная константа или номер сечения. Последнее рассмотрим чуть более подробно ниже;

I1 – присваиваемое значение. Может быть несколько значений, в зависимости от STLOC.

Этой командой мы присваиваем элементу или, в данном случае, всей модели тип элемента shell163:

emodif,1,type,Shell163

Но этой командой будет модифицирована вся модель. Если же нам нужно придать уникальное свойство каждому элементу, необходимо присвоить каждому элементу уникальное сечение. Для этого служит команда:

SecType, SECID, Type

где SecType – объявляет о команде;

SECID – уникальный номер сечения;

Type – тип элемента в сечении.

Для присвоения каждому элементу своего сечения можно воспользоваться циклом по элементам, как это рассматривалось в предыдущих лекциях.

Еще одна команда, позволяющая менять некоторые геометрические параметры модели – secdata. Для каждого типа элемента она имеет свой набор параметров, мы рассмотрим пример с shell181: Secdata,2 – установит толщину листа в 2 мм.

Может быть несколько параметров, необходимо уточнять для каждого типа элемента.

С помощью этой команды можно присвоить каждому элементу свою толщину, для этого можно воспользоваться циклом. Вообще говоря, все поэлементные действия выполняются с использованием циклов.

8 ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ APDL ДЛЯ РЕШЕНИЯ ЗАДАЧ. МАКРОСЫ

В данной лекции мы рассмотрим некоторые нестандартные задачи, которые можно решить с помощью APDL. К таким задачам можно отнести все, что нельзя увидеть или задать напрямую, используя инструменты, предоставляемые Ansys. Это связано с тем, что некоторые команды были убраны из интерфейса Workbench, так как многое в нем автоматизировано (например, выбор типа конечного элемента в Ansys Mechanical APDL необходимо задавать вручную). Это удобно, особенно для неподготовленного пользователя, однако лишает определенной гибкости при решении задачи.

Решить эту проблему можно с помощью APDL-команд.

8.1 Совместное использование команд из различных процессоров

Часто приходится сталкиваться с тем, что необходимо получить какие-либо данные из постпроцессора для того, чтобы использовать их при решении, которое находится в препроцессоре.

Например, если нам нужно использовать в препроцессоре значения напряжений, возникающих в элементе в результате нагрузки. Для того чтобы их получить, мы воспользуемся командой Etable. Однако если записать ее в препроцессор, мы получим сообщение вида, показанного на рисунке 8.1.

```
*** WARNING ***                         CP =       0.891   TIME= 17:59:04
ETABLE is not a recognized PREP7 command, abbreviation, or macro.
 This command will be ignored.
```

Рисунок 8.1 – Сообщение об ошибке в логе

В данном сообщении говорится, что команда etable не относится к препроцессору и может быть проигнорирована. На практике это означает, что произойти может что угодно. Иногда программа просто игнорирует команду и проводит решение задачи без ее учета, иногда просто останавливается и выдает сообщение о ошибке, иногда считает нормально.

Вообще говоря, желательно проверять лог на предмет ошибок после каждого решения, но об этом чуть ниже.

Итак, нам нужно получить значения напряжений. Очевидно, что эти значения мы можем получить только после нагружения, следовательно, нужно использовать постпроцессор /post1.

Для этого запишем следующие команды:

```
/solu
Solve
finish
/post1
etable,stress_table,s,eqv
finish
```

Таким образом, мы получим значения напряжений в таблице stress_table.

Важно: /solu – запускает решатель для того, чтобы Ansys корректно провел расчёт. Также важно закрывать процессор после использования командой finish.

Теперь возникает вопрос, как их использовать в препроцессоре. Существует несколько путей:

1) Можно записать их в заранее созданный массив (*dim). Для этого потребуется использовать цикл. Плюсом данного метода является то, что в дальнейшем этот массив можно будет использовать в любом месте программы, независимо от процессора. Выглядеть это будет примерно следующим образом:

```
/post1
  etable,stress_table,s,eqv
  *do,i,1,ne
          *get,elem_stress(i),elem,i,etab,stress_table
  *enddo
Finish
```

3) Можно воспользоваться командой *get прямо в препроцессоре:

```
/post1
etable, stress_table,s,eqv
finish
Теперь запустим препроцессор:
/prep7
*do,k,1,ne
elk=el(k)
*get,esk,elem,elk,etab,stress_table
….
Finish
```

Данный способ удобен тем, что не требует дополнительных массивов. Кроме того, им удобнее пользоваться при итерационных процессах, когда в ходе решения необходимо получить значение напряжений несколько раз.

В целом, оба варианта имеют место быть и какой из них применять – выбор пользователя.

Теперь рассмотрим несколько практических задач.

8.2 Создание оболочечного тела с переменной толщиной, определяемой уравнением

Возможно, возникнет ситуация, когда понадобится создать оболочечное тело переменной толщины, причем изменяющейся по определенному закону.

Сделать это встроенным редактором очень сложно (если вообще возможно), импортировать созданный внешним графическиv редактором не всегда возможно. Кроме того, следует помнить, что импортированное тело не всегда переносится с адекватным сохранением геометрии.

В этом может помочь команда RThick.

Рассмотрим пример:

Пусть толщина определяется уравнением:

$$T = 0,5 + 0,2x + 0,02y^2 .$$

Тогда программа будет выглядеть следующим образом (рисунок 8.2).

```
*GET,MXNODE,NODE,,NUM,MAXD
*DIM,THICK,,MXNODE
*DO,NODE,1,MXNODE
   *IF,NSEL(NODE),EQ,1,THEN
      THICK(node) = 0.5 + 0.2*NX(NODE) + 0.02*NY(NODE)**2
   *ENDIF
*ENDDO
```

Рисунок 8.2 – Алгоритм определения толщины

Здесь используется свойство оболочечных элементов – толщина задается в узле. Рассмотрим алгоритм подробнее:

1) Получаем максимальный номер узла.

2) Перебираем узлы, присваивая каждому узлу, имеющему номер 1 (эквивалентно узлу I, т.е. основному).

После чего используем команду RThick:

Rthick,Thick(1),1,2,3,4
Структура команды проста:
Rthick – запуск команды
Thick(1) – значение толщины
1,2,3.4 – узлы I,J,K,L

Отметим, что будет достаточно назначить толщину для узла I, как для основного.

После выполнения мы получим тело, представленное на рисунке 8.3.

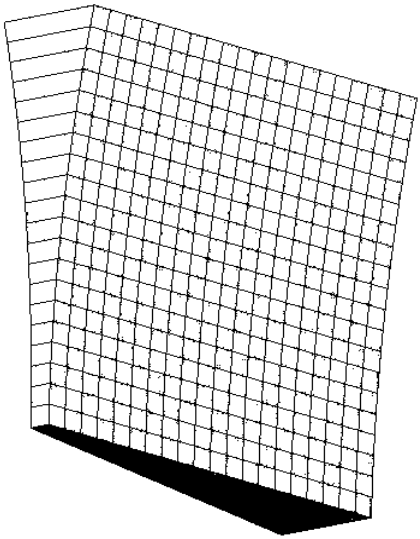

Рисунок 8.3 – Оболочечное тело переменной толщины

8.3 Создание и использование макросов

ANSYS позволяет использовать макросы, то есть заранее написанный код.

Это удобно тем, что позволяет использовать чужой код неподготовленному пользователю или записать для себя часто используемые команды в файл, чтобы вызывать их по необходимости.

Создать макрос очень просто. Для этого нужно создать файл блокнота, записать на нем команды на языке APDL и сохранить его, задав разрешение .macro.

То есть файл должен называться примерно так: Macr.macro.

Для его вызова выполняется команда *use:

*use,/myaccount/macros/Mac.macro

где *use объявляет о запуске макроса, затем следует путь к макросу.

8.4 Использование ANSYS для решения задач оптимизации

Одной из самых интересных возможностей ANSYS является возможность решения пользовательских оптимизационных задач [21]. Фактически, говоря «оптимизация» мы имеем ввиду минимизацию некоторого критерия при выполнении заданных ограничений. Для авиационных конструкций в качестве такого критерия зачастую выступает масса, а в качестве ограничений – требования прочности.

Вообще говоря, ANSYS имеет собственные инструменты для проведения оптимизации, как требующие знания APDL, так и позволяющие обойтись без него. К сожалению, они не всегда эффективны.

Оптимизация с помощью APDL-команд будет рассмотрена в лабораторной работе «оптимизация плоской пластинки».

В заключение хотелось бы привести несколько советов.

1) Описание абсолютно всех команд можно найти в разделе Help, встроенном в ANSYS. Там же можно найти некоторые примеры использования APDL-команд.

2) Если APDL-команды по каким-либо причинам не выполняются, то можно их исключить из решения (ПКМ на commands (APDL)) и выбрать suppressed) и проверить, будет ли ANSYS работать без них. Нередко ошибки возникают из-за неверного закрепления, неверно приложенной нагрузки, некорректной модели и много другого, с APDL не связанного.

3) После решения всегда полезно проверить лог на ошибки (рисунок 8.4).

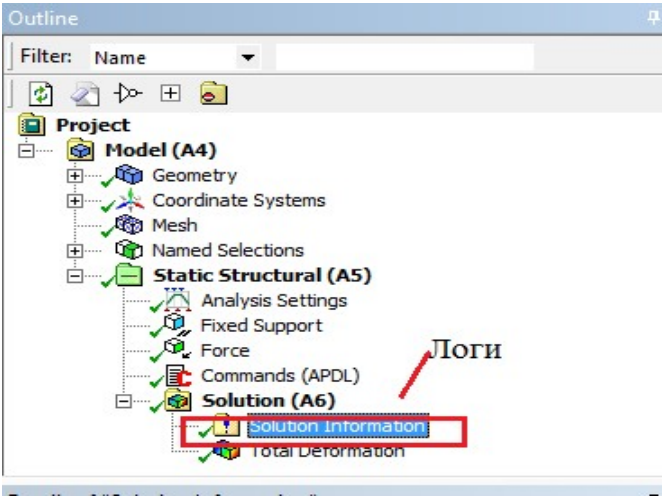

Рисунок 8.4 – Местоположение лога в дереве проекта

4) Полезно изучить файл ds.dat, находящийся в директории решателя. В нем описаны все шаги работы программы, от открытия решателя и до его закрытия. Из этого файла можно узнать много любопытных команд и синтаксис языка APDL в целом.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Гилат, А. MATLAB. Теория и практика / А. Гилат; пер. с англ. Н.К. Смоленцев. – М: ДМК Пресс, 5-е издание, 2016. – 416 с.
2. Дьяконов, В.П. MATLAB 6/6.1/6.5 + Simulink 4/5 в математике и моделировании. Полное руководство пользователя / В.П. Дьяконов. – М: СОЛОН-Пресс, 2009. – 576 с.
3. Клебанов, Я.В. Использование программного комплекса ANSYS в учебном процессе / Я.В. Клебанов, А.Н. Давыдов, В.Л. Папировский. – Самара: Самарский государственный технический университет. – Сайт cadfem.ru.
4. Басов, К.А. ANSYS в примерах и задачах / К.А. Басов; под общ. ред. Д.Г. Красковского // М: Компьютер пресс, 2002. – 224 с.
5. Чигарев, А.В. ANSYS для инженеров: справочное пособие / А.В. Чигарев, А.С. Кравчук, А.Ф. Смалюк. – М: Машиностроение-1, 2004. – 512 с.
6. Ревинская, О.Г. Символьные вычисления в MATLAB: учебное пособие для вузов / О.Г. Ревинская. – Санкт-Петербург: Лань, 2-е издание, 2023. – 528 с.
7. Рябикова Т.В. Решение задач математического анализа с использованием MATLAB: учебно-методическое пособие / Т.В. Рябикова, Л.Ю. Уразаева. – М: ФЛИНТА, 2021. – 108 с.
8. Сирота, А.А. Методы и алгоритмы анализа данных и их моделирование в MATLAB: учебное пособие / А.А. Сирота. – СПб.: БХВ-Петербург, 2016. – 384 с.
9. Морозов, Е.М. ANSYS в руках инженера: Механика разрушения / Е.М. Морозов, А.Ю. Муйземнек, А.С. Шадский – М: ЛЕНАНД, 2-е издание, 2010. – 456 с.
10. Басов К.А. ANSYS: справочник пользователя / К.А. Басов – М: ДМК Пресс. – 640 с.

11. Жидков, А.В. Применение системы ANSYS к решению задач геометрического и конечно-элементного моделирования. Учебно-методический материал / А.В. Жидков. – Нижний Новгород: Нижегородский государственный университет им. Н.И. Лобачевского, 2006. – 115 с.

12. Каплун, А.Б. ANSYS в руках инженера: практическое руководство / А.Б. Каплун, Е.М. Морозов, М.А. Олферьева – М: Едиториал УРСС, 2003. – 272 с.

13. Чигарев, А.В. ANSYS для инженеров: справочное пособие / А.В. Чигарев, А.С. Кравчук, А.Ф. Смалюк. – М: Машиностроение-1, 2004. – 512 с.

14. Бате, К.Д. Численные методы анализа и метод конечных элементов / К.Д. Бате, Е.Л. Вилсон; пер. с англ. – М: Стройиздат, 1982. – 448 с.

15. Галлагер, Р. Метод конечных элементов. Основы / Р. Галлагер; пер. с англ. – М: Мир, 1984. – 428 с.

16. Вержбицкий, В.М. Основы численных методов: учебник для вузов / В.М. Вержбицкий. – М: Высш. шк., 2005. – 840 с.

17. Норри, Д. Введение в метод конечных элементов / Д. Норри, Ж. Де Фриз; пер. с англ. – М: Мир, 1981. – 304 с.

18. Сегерлинд Л. Применение метода конечных элементов / Л. Сегерлинд; пер. с англ. – М: Мир, 1979. – 392 с.

19. Зенкевич, О.К. Метод конечных элементов в технике / О.К. Зенкевич; пер. с англ. – М: Мир, 1975. – 541 с.

20. Стренг, Г. Теория метода конечных элементов / Г. Стренг, Дж. Фикс; пер. с англ. – М: Мир, 1977. – 349 с.

21. Christensen, P. W. An introduction to structural optimization / P. W. Christensen, A. Klarbring. – Berlin: Springer, 2009. – 214 p.

Учебное издание

Куркин Евгений Игоревич,
Кишов Евгений Алексеевич

ОСНОВЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ АВИАЦИОННЫХ КОНСТРУКЦИЙ

Учебное пособие

Редакционно-издательская обработка
И.И. Спиридоновой

Подписано в печать 01.11.2023. Формат 60×84 1/16.
Бумага офсетная. Печ. л. 4,5.
Тираж 120 экз. (1-й з-д 1-27). Заказ №

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)
443086, САМАРА, МОСКОВСКОЕ ШОССЕ, 34.

Издательство Самарского университета.
443086, Самара, Московское шоссе, 34.