# ВВОДНОЕ ЛАБОРАТОРНОЕ ЗАНЯТИЕ

# ОСВОЕНИЕ ИНТЕРФЕЙСА MATLAB

Цель работы

Получение общего представления о математическом пакете Matlab – особенностей интерфейса, функциональных основных возможностей, формирования навыков практической работы в среде Matlab, математических вычислений, моделирования, разработки приложений и анализа данных.

Разбор примеров

Была запущена программа MATLAB. В её командное окно была введена команда *help why*, выводящая справку о команде *why* (Рисунок 1). Альтернативой данному методу поиска справочной информации является команда doc, открывающая окно документации.

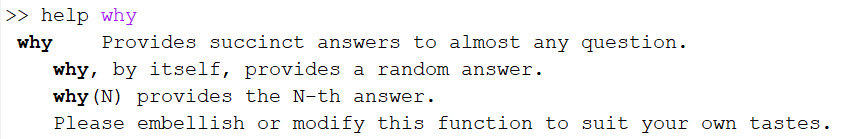


Рисунок 1 – Выполнение команды *help*

Затем были записаны операнды с явным и неявным присвоением переменных среды (Рисунок 2).

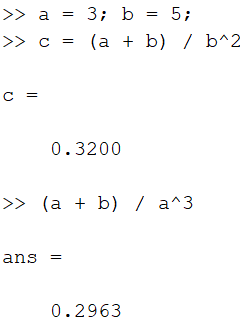


Рисунок 2 – Запись операторов

В переменную *a* поочерёдно были записаны вектор и матрица чисел (Рисунок 3).

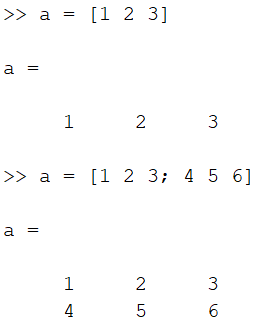


Рисунок 3 – Задание вектора и матрицы

Затем через специальную конструкцию был задан новый вектор чисел *x* (Рисунок 4). Затем в результате операции над ним был создан ещё один вектор *y*.

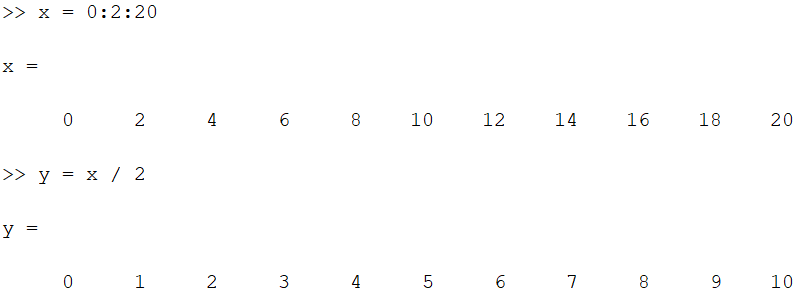


Рисунок 4 – Задание и вычисление вектора

Также было опробовано задание векторов с помощью различных функций, таких как *linspace* (Рисунок 5).

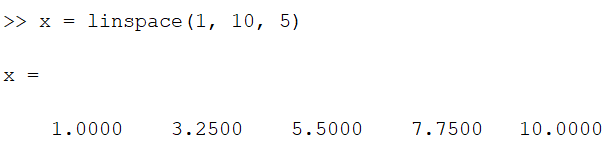


Рисунок 5 – Задание вектора функцией

Была успешно протестирована работа функций *rand* и *randi* для создания матриц случайных чисел n-ного порядка (Рисунок 6, Рисунок 7).

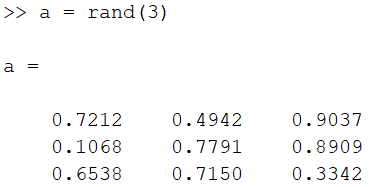


Рисунок 6 – Использование функции *rand*

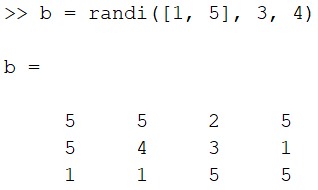


Рисунок 7 – Использование функции *randi*

Далее были протестированы различные способы обращения к элементам матрицы. Для этого в созданной ранее матрице *b* были измены элемент, столбец и строка (Рисунок 8).

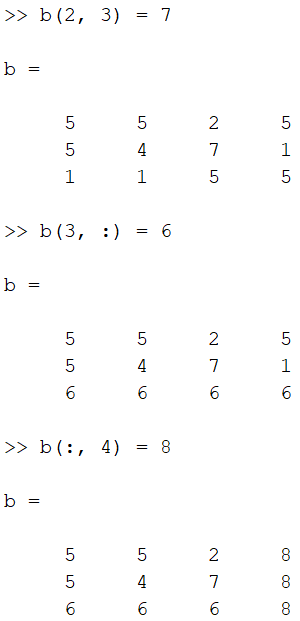


Рисунок 8 – Изменение фрагментов матрицы

Из матрицы был удалён первый столбец путём присвоения ему значения пустого массива (Рисунок 9).

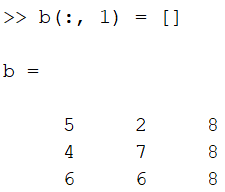


Рисунок 9 – Удаление столбца

С помощью конструкции с шагом был заполнен вектор *x*. На основе него с помощью операндов был построен вектор *y* (Рисунок 10). Так был получен вектор y, состоящий из значений описанной функции *y(x)*.

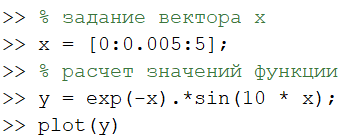


Рисунок 10 – Заполнение вектора *y* по вектору *x*

Через функцию *plot* вызвано графическое окно, где был построен график значений вектора *y* на основе их порядкового номера (Рисунок 11).

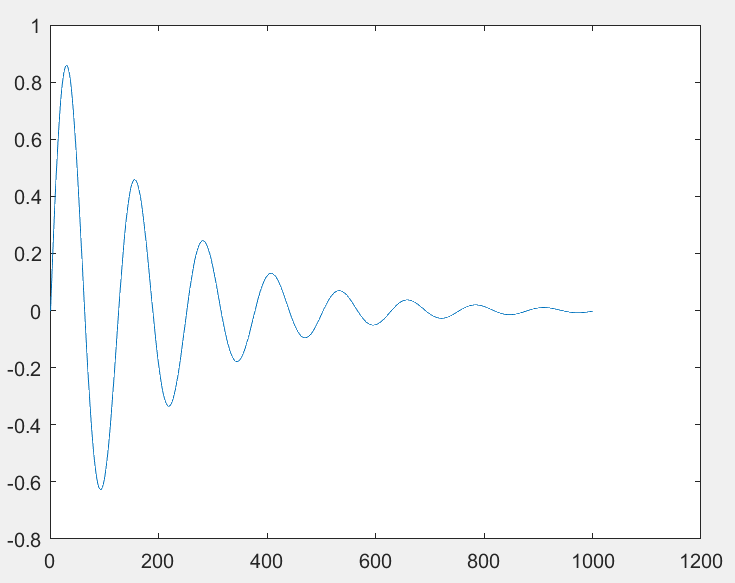


Рисунок 11 – Графическое окно

Через команду *plot(x, y)* был построен график значений вектора *y* относительно вектора *x*, т.е. график функции *y(x)* (Рисунок 12).

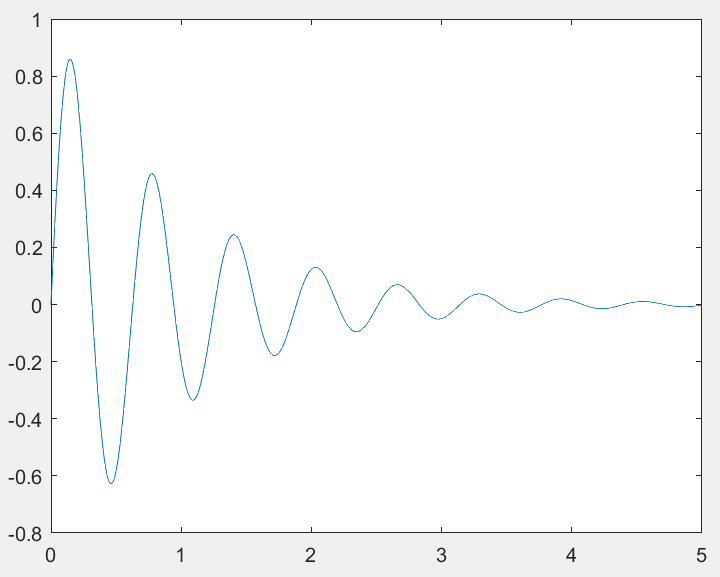


Рисунок 12 – График функции *y(x)*

Далее был написан ряд команд, заполняющий два новых вектора на основе вектора *x* (Рисунок 13). Таким образом были смоделированы функции *y1(x)* и *y2(x)*.

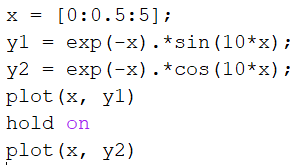


Рисунок 13 – Заполнение векторов *y1* и *y2*

С помощью поочерёдных вызовов команды *plot*, связанных командой *hold on*, графики функций *y1(x)* и *y2(x)* были построены в одном графическом окне (Рисунок 14).

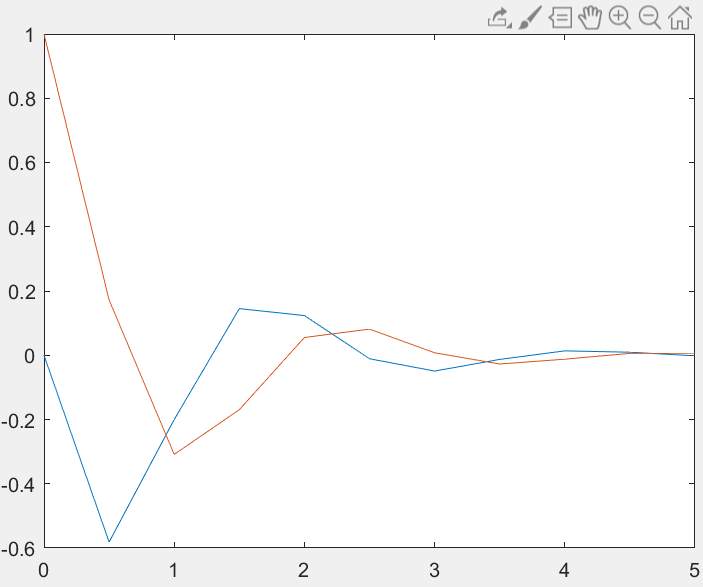


Рисунок 14 – Графики функций *y1(x)* и *y2(x)*

Похожий результат был получен путём вызова команды *plot* с дополнительными аргументами, влияющими на цвет, тип линии и метки графика. Помимо этого, было продемонстрировано действие команд *title*, *xlabel*, *ylabel* и *grid on* (Рисунок 15).

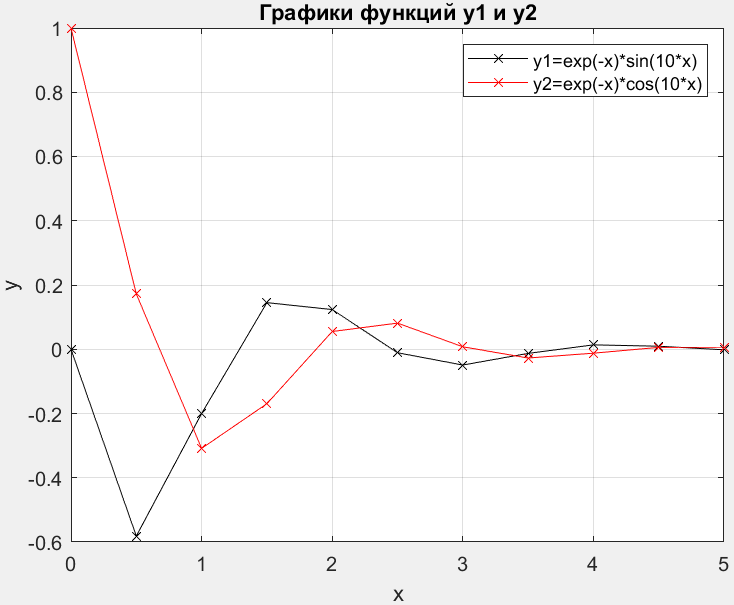


Рисунок 15 – Результат построения графиков c дополнительными настройками

Далее была продемонстрирована работа команды *subplot* (Рисунок 16). В результате было открыто графическое окно с четырьмя графиками функций, где каждая расположена в своём подокне (Рисунок 17).

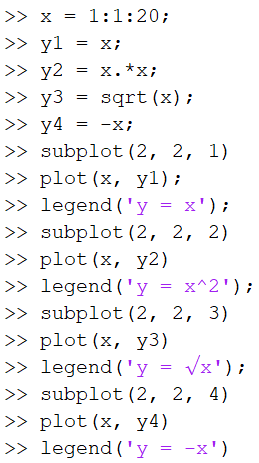


Рисунок 16 – Создание векторов и вызов графического окна

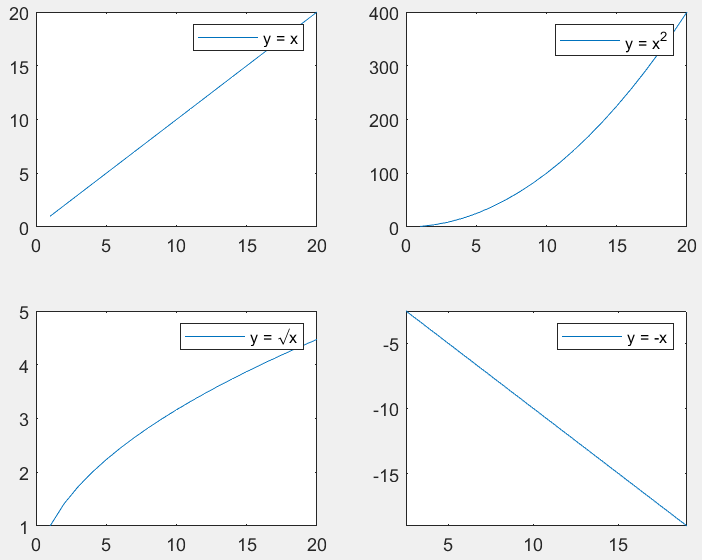


Рисунок 17 – Результат действия команды *subplot*

Был написан файл с процедурой, которая по переданному параметру *x* возвращает два значения *y* (Рисунок 18).

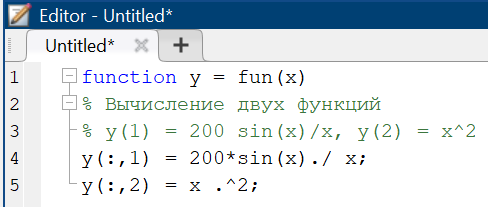


Рисунок 18 – Создание процедуры

Написанная процедура была вызвана в командном окне (Рисунок 19). Также был построен график возвращаемых ею значений в зависимости от значения параметра на интервале [-20;20] (Рисунок 20).

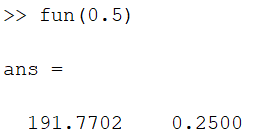


Рисунок 19 – Вызов процедуры

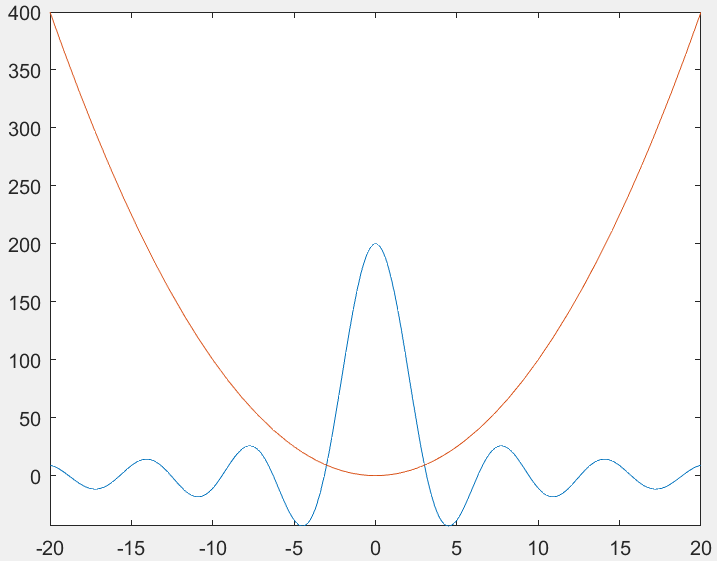


Рисунок 20 – Построение графика для процедуры

Описание библиотек

В каталоге matlab/toolbox/matlab можно найти множество библиотек, содержащих справочную информацию об операторах, функциях и конструкциях языка системы Matlab.

Библиотека **ops** содержит данные обо всех операторах системы Matlab: арифметических, логических, побитовых, операторах отношения, операторах над множествами, специальных символах.

В ней расположены библиотеки **relop** и **slash**. В первой находятся данные о логических операторах и операторах отношения, во второй − данные об операторах, отвечающих за матричное деление (*/*, *\*, *./*, *.\*).

Библиотека **elfun** содержит данные об элементарных математических функциях, установленных в Matlab: тригонометрических, экспоненциальных, комплексных, функциях округления и остатка.

Библиотека **specfun** содержит данные о специальных математических функциях Matlab, которые являются решениями дифференциальных уравнений специального вида или обозначениями некоторых видов интегралов. По своей специфике они аналогичны элементарным функциям.

В каталоге graph2d можно найти библиотеку **plot**, в которой содержится информация об одноимённой команде. С помощью неё можно строить графики по векторам и матрицам, а также настраивать их цвет, тип линии и др.

Задание по варианту

Необходимо вычислить значения функции на отрезке [; ] с шагом (Таблица 1) и построить её график, настроив его оформление.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант |  |  |  |  |
| 2 |  | 2.06 | 8.1 | 1.2 |

Таблица 1 – Вариант задания

Для этого был создан вектор *x*, содержащий точки на отрезке [; ] с шагом . Затем по нему с помощью различных операторов был построен вектор *y*, содержащий значения функции в данных точках (Рисунок 21).

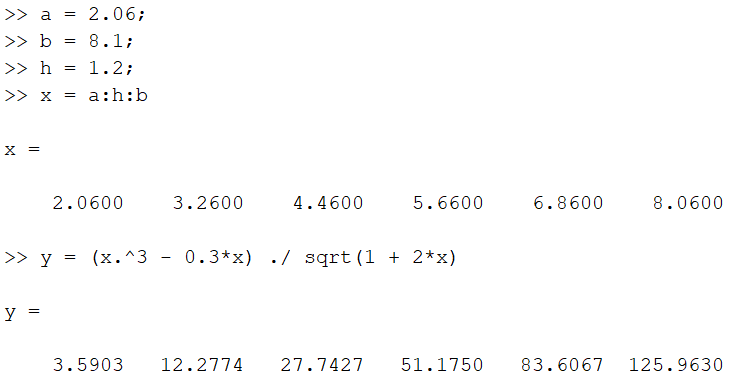


Рисунок 21 – Поиск значений функции на отрезке

Далее был построен график функции . Для этого вектора *x* и *y* были перезаписаны с меньшим шагом для большей плавности и точности графика (Рисунок 22).

Также для графика функции было настроено оформление (цвет, тип линии), добавлены подписи осей и название графика.

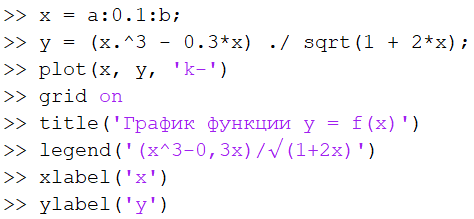


Рисунок 22 – Построение графика функции

В результате график функции был отображён в графическом окне (Рисунок 23).

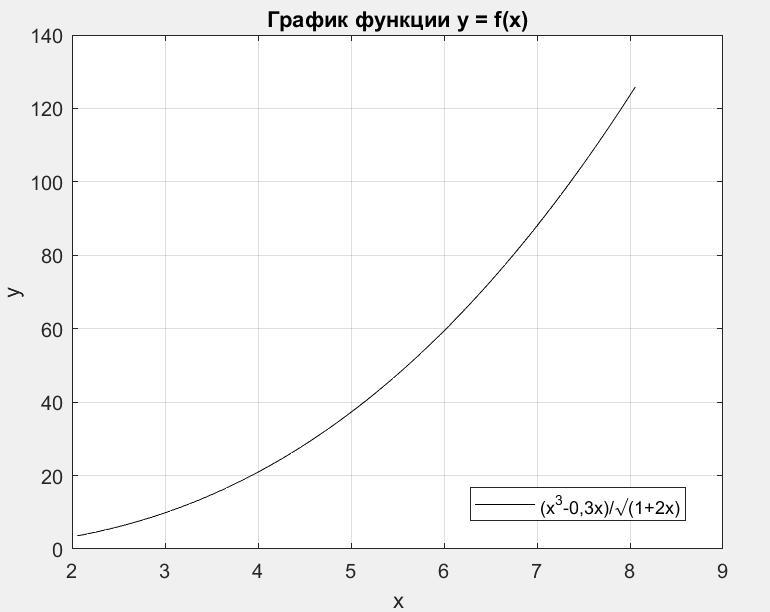


Рисунок 23 – График функции

Контрольные вопросы

1. Что такое Matlab?

MATLAB − высокоуровневый интерпретируемый язык программирования, а также пакет прикладных программ и интегрированная среда для разработки, выполнения инженерных и математических расчётов, работы с матричными базами данных и др.

1. Каким образом можно получить справку о той или иной команде?

Обратиться к справочной системе можно с помощью команды *help* (выводит на экран список каталогов) или *doc* (открывает окно документации). Чтобы просмотреть справку конкретной функции необходимо дописать после команды её название.

1. Как вводятся матрицы и векторы в языке Matlab? Какими функциями можно формировать специальные матрицы и векторы в языке Matlab?

Для ввода векторов и матриц используются квадратные скобки []. Разделительными символами для вектора служат пробел и запятая, для строк матрицы – точка с запятой.

Задать вектор можно конструкцией: [начальное значение : [шаг] : конечное значение]. По умолчанию шаг равен единице.

Также для создания матриц используются различные функции: *linspace*, *ones* (все единицы), *zeros* (все нули), *eye* (единичная матрица), *magic* (магический квадрат), *rand* (случайные числа).

1. Какие функции Matlab осуществляют вывод графиков на экран?

*plot* – график в линейном масштабе;

*loglog* – график в логарифмическом масштабе;

*semilogx*, *semilogy* – график в полулогарифмическом масштабе;

*polar* – график в полярных координатах.

1. Какими функциями обеспечивается снабжение графика координатными линиями и надписями?

*title* – название графика;

*legend* – подпись графика функции;

*xlines* – подпись горизонтальной оси;

*ylines* – подпись вертикальной оси;

*grid on* – линии разметки координатной плоскости;

1. Можно ли построить несколько графиков в одной системе координат и в одном графическом окне?

Можно, для этого нужно либо указать в функции *plot* дополнительные аргументы для второго графика, либо использовать команду *hold on* для размещения графиков в одной плоскости.

1. Как построить несколько отдельных графиков в одном графическом окне в разных графических полях?

Необходимо перед созданием каждого нового графика использовать команду *subplot(m, n, p)*. Значение *m* указывает, на сколько частей разбивается окно по горизонтали, *n* – по вертикали, а *p* – номер подокна, куда будет выводиться очередной график.

Вывод

В ходе работы были исследованы основные компоненты и возможности математического пакета MATLAB, а именно устройство интерфейса, наиболее важные операторы и функции, справочная система, построение и обработка векторов и матриц, методы визуализации данных в виде графиков и их настройка, написание скриптов и обращение к ним.