

Методическое пособие для первокурсников ФФПФ МФТИ
по курсу «Проектная деятельность»
для проектов «Анализ колебаний солнечных корональных петель» и
«Моделирование движения заряженных частиц в магнитном и
электрическом полях»

Введение в программирование на языке МАТЛАВ

Февраль 2019

Содержание

I	Основные понятия	2
1	Рабочая среда и настройка	2
2	Использование MATLAB как калькулятора	3
3	Основы работы с графикой, скрипты	5
4	Циклы и условия	9
II	Первый блок заданий	10
5	Задачи, обязательные для всех	10
6	Задачи повышенного уровня	12
III	Ещё немного о работе с графикой и выводом данных	13
IV	Второй блок заданий	13

Часть I

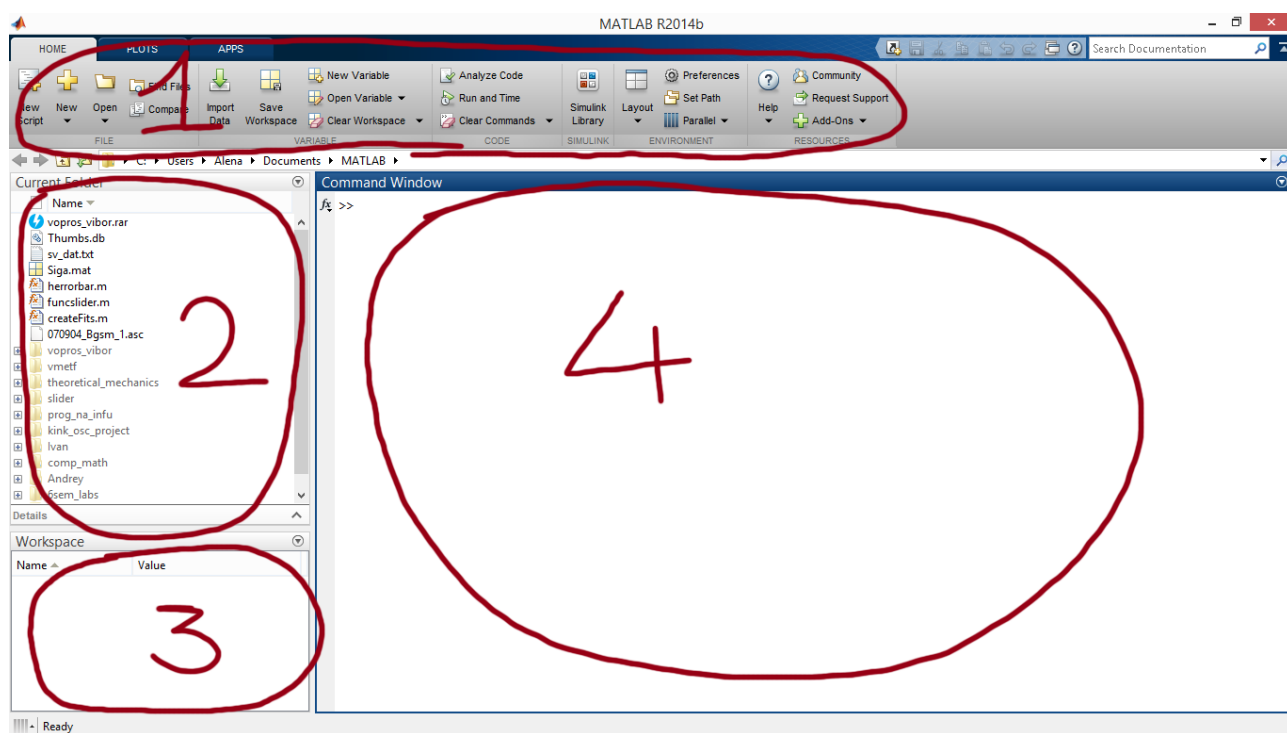
Основные понятия

Примечание: на всех скринах показана моя рабочая среда MATLAB R2014b, однако по сути своей более поздние версии почти не отличаются от этой.

1. Рабочая среда и настройка

Основную информацию по MATLAB можно посмотреть в приложенной к моему курсу книжке INTRODUCTION TO MATLAB FOR ENGINEERING STUDENTS, которую можно найти на GITHUB в папке с материалами для вашего проекта. Она написана на английском языке, но он там достаточно тривиальный, и я надеюсь, с обращением к этой книге в непонятных ситуациях проблем не возникнет. **Приступим.**

На скрине ниже можно увидеть, как выглядит окно программы, в которой мы с вами будем работать:



Что здесь есть что? Цифрой 1 обозначена панель инструментов: здесь нам понадобится сначала одна кнопка: **Preferences**, здесь можно многое под себя кастомизировать, для меня лично удобной функцией является настройка шрифтов во вкладке **Fonts**.

Цифрой 2 обозначен перечень файлов в папке, с которой сейчас работает MATLAB. Здесь можно соответственно бродить между папками, открывать файлы и создавать новые райт-кликом.

Цифрой 3 обозначен перечень ваших переменных в памяти, с которыми вы работаете и их значения.

Цифрой 4 обозначено командное окно, в котором можно как писать всякие короткие команды (не создавая для этого отдельных скриптов), так и сюда выводятся результаты выполнения ваших скриптов, если есть что вывести.

2. Использование MATLAB как калькулятора

Нажми на кнопку – получишь результат,
И твоя мечта осуществится.
Нажми на кнопку, но что же ты не рад.
Тебе больше не к чему стремиться.
песня группы "Технология"

Для знакомства с MATLAB начнём с простого: научимся оперировать с цифрами. Самое простое задание: попросим нашу программу сложить два числа. Для этого в командном окне напишем:

```
> > 3 + 2
```

И нажмём **Enter**. Вследствие чего получим ответ:

```
ans =  
5
```

Также в **Workspace** появилась переменная **ans** со значением 5. Если мы ещё раз таким же образом сложим другие два числа, то их сумма запишется в переменную **ans** поверх предыдущего значения. То есть это стандартная переменная, вводимая MATLAB, если у нас не указано, куда класть результат вычислений. Теперь, допустим, мы хотим указать, в какие переменные класть наш результат. Напишем следующее:

```
> > a = 6 + 3;  
b = 6 - 3
```

На выходе получаем:

```
b =  
3
```

Почему вывелась только вторая переменная? Дело в том, что символ ; ставит запрет на вывод результата действия в командное окно. Результат просто пишется в переменную, которую мы указали, но не выводится для нас. Это удобно использовать для промежуточных результатов в расчетах, которые нам не особенно важны, ведь зачем захламлять окно выхода? Всё должно быть чётко и понятно. Мы научились мало-мальски работать с числами, но MATLAB славится простотой работы с матрицами, давайте также постараемся понять, в чём заключается эта простота, и, как вы поймёте дальше, удобность для непосредственного решения задач.

Для задания матрицы со натуральными значениями от 1 до 10 запишем:

```
> > A = [1 2 3 4 5 6 7 8 9 10]
```

Здесь пробелы выступают в качестве разделителя элементов нашей числовой строки, но для этого также можно использовать и запятые, если кому так удобнее. На выходе получаем матрицу 1x10. Обратите внимание на **Workspace**. Наша переменная **a** осталась, но также и появилась переменная **A**. МАТЛАВ чувствителен к регистру, имейте это ввиду. Теперь зададим другую матрицу 1x10, чтобы сложить её с уже у нас имеющейся. Пусть теперь это будут числа от -1 до -10. Быстро, а не как в прошлый раз, такую матрицу можно задать через команду:

```
> > B = [-1:-1:-10]
```

Где первое и последнее числа определяют интервал в котором будут находиться наши значения, а число посередине – это шаг. Таким образом мы получаем матрицу, где значения – это числа от -1 до -10 с шагом -1. Сложение двух матриц задаётся простейшей командой

```
> > C = A + B
```

Получили, как и ожидали, строку из десяти нулей:

```
C =  
    0    0    0    0    0    0    0    0    0    0
```

А теперь вспомните, как вы раньше писали для такого действия циклы. Правда же удобно?

Теперь очистим наш **Workspace**, для чего введём команду **clear** или райткликом на воркспейсе выберем **Clear Workspace**. Командное окно чистится командой **cls** или же тем же райткликом по нему и выбором **Clear Command Window**.

Зададим четыре произвольных числа:

```
> > a = 10*rand;  
b = 10*rand;  
c = 10*rand;  
d = 10*rand;
```

Здесь **rand** выдаёт нам случайное число из [0,1]. Соответственно **10*rand** выдаст случайное число из [0,1]. Также матрицу мы можем задавать не только непосредственно вписывая туда числа, но и вписывая туда переменные:

```
> > A = [a b; c d]  
A =  
    6.3236    0.9754  
    2.7850    5.4688
```

Знак **;** используется для раздела строчек в матрице. Также квадратную матрицу случайных чисел можно задавать и таким образом, где 2 – размерность матрицы:

```
> > B = rand(2)  
B =  
    0.9575    0.1576  
    0.9649    0.9706
```

Также для наглядности умножения матриц зададим матрицу, состоящую из единиц:

```
> > C = [1 1;1 1]
C =
     1     1
     1     1
```

Я думаю, вы уже додумались, что перемножение матриц также задаётся простейшей командой

```
> D1 = A*C
D1 =
     7.2990     7.2990
     8.2538     8.2538
```

А сейчас приготовьтесь, поэлементное перемножение матриц:

```
> D2 = A.*C
D2 =
     6.3236     0.9754
     2.7850     5.4688
```

Как вы поняли, получилась в точности матрица **A**, что и должно было получиться при поэлементном умножении на матрицу из единиц. Таким образом, ставя точку перед знаком операции, мы говорим, что хотим, чтобы эта операция была выполнена поэлементно.

Краткий экскурс в примитивные операции закончен, больше функций вы можете посмотреть в файле `MATLABREF.PDF`, который также находится в папке с вашим проектом. Советую всегда иметь этот важный файл под рукой при работе в случае чего.

3. Основы работы с графикой, скрипты

Человек любит картиночки
Пухов А.А.

Очень важный раздел, в котором я постараюсь максимально ёмко и понятно описать принципы работы с графикой в `MATLAB`, а также научить правилам хорошего тона в оформлении ваших картинок.

Зачем нам вообще здесь нужна графика? Во-первых, чтобы самому понимать, что же происходит в задаче, которую вы решаете (например вы посчитали, как у вас размывается волновой фронт со временем при движении какого-либо возмущения в среде, но вы получили для каждого набора времён только какие-то числа. Теперь нужно это нарисовать, чтобы стало понятно, что же творится!), во-вторых, чтобы объяснить другим, что же происходит в вашей задаче. Это задача не всегда тривиальная, так как задача может быть вполне сложной, вы могли решать её полгода, и все эти полгода работать ради пары картиночек, но зато очень ёмких. И чтобы презентовать результаты своей работы, эти картиночки должны выглядеть красиво и читабельно.

Начнём с простых вещей: нарисует график синусоиды, для этого зададим вектор `x` икс:

```
x = [0:50];
```

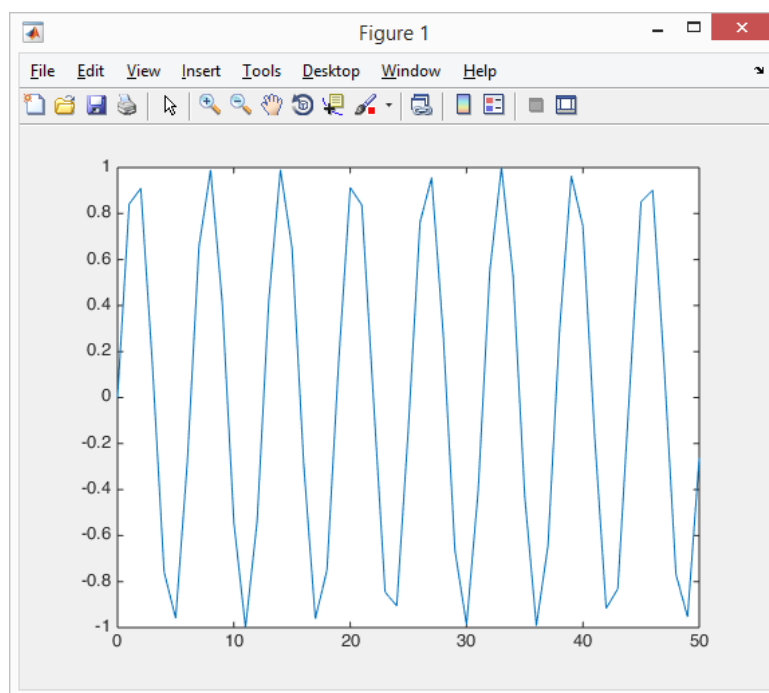
и зададим синусоиду:

```
y = sin(x);
```

Получили строку каких-то чисел, из которых не совсем очевидно, что все они – значения функции `sin(x)`. Чтобы это понять, выполним следующую команду, предназначение которой вполне очевидно:

```
> > plot(x,y)
```

Нам вылезло такое окошко:



Как мы видим, это немного угловатый, но всё же синус. Закроем это окошко и зададим по оси `x` больше точек, чтобы наш синус был более плавным:

```
> > x1 = [0:0.1:50];
```

```
y1 = sin(x1);
```

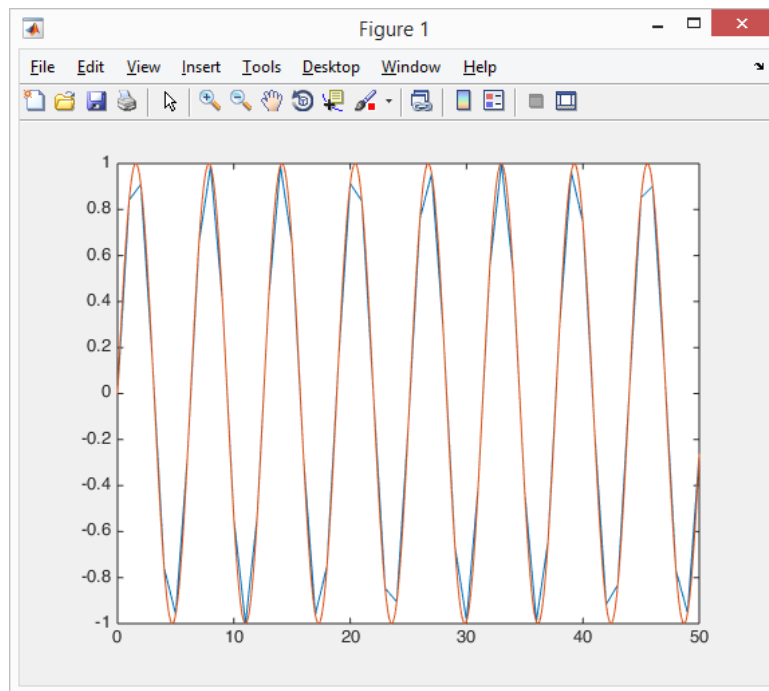
И начертим наш "угловатый" синус, а поверх него более "плавный":

```
> > plot(x,y)
```

```
hold on
```

```
plot(x1,y1)
```

Команда `hold on` удерживает активное графическое окно и не создаёт нового. По умолчанию команда `plot` создаёт новое графическое окно (`figure`), в чём вы можете убедиться, убрав из кода `hold on`, каждый из графиков будет в отдельном окошке. Получили такой график:



С помощью иконок с лупами можно приближать/отдалять наш график (при приближении даблклик возвращает исходный масштаб), с помощью иконки с рукой – двигать.

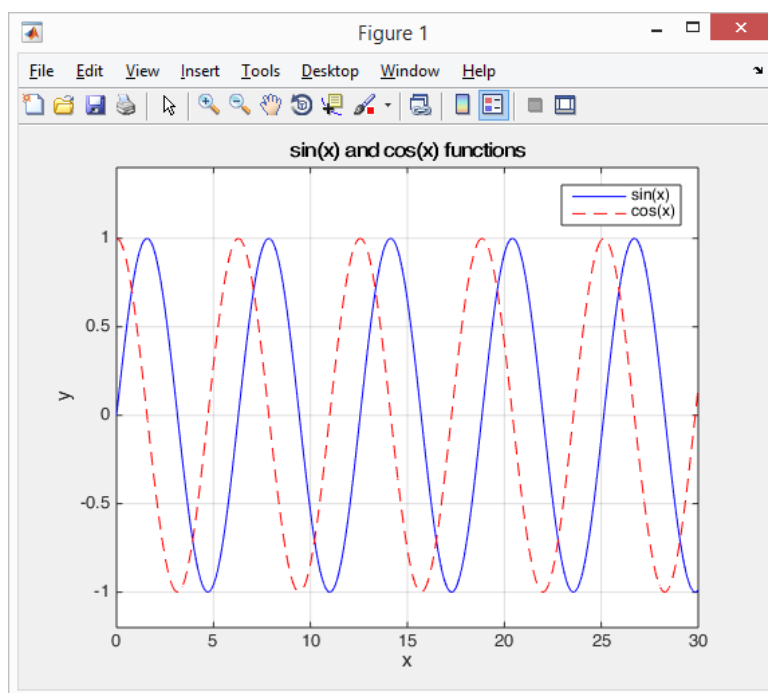
Но всё же наш график выглядит достаточно сыро, не так ли? Для этого создадим новый скрипт. Скрипты в MATLAB – это, по сути своей, маленькие автономные программки. В командном окне работать неудобно, код нужно постоянно править, да и он не занимает две строчки, как если нужно использовать MATLAB исключительно как калькулятор для домашки по линейной алгебре.

В разделе **Current Folder** заходим в папку, в которой будем работать, и создаём там скрипт (**New File > Script**). Получили файл с расширением .м, теперь открываем его. Получили пустой файл, в котором будем писать код нашей программы построения красивой синусоиды. Итак, пишем внутрь нашего скрипта следующий код:

```
clear;
x = [0:0.1:30];
y1 = sin(x);
h1 = plot(x,y1,'b') %b - обозначение для синего цвета линии
grid on %включение сетки
axis([0 30 -1.2 1.4]) %пределы по оси Ox от 0 до 30 и по оси Oy от -1.2 до 1.4
hold on
y2 = cos(x);
h2 = plot(x,y2,'r-') %r - обозначение для красного цвета линии,
      два дефиса - пунктирная линия
legend([h1,h2],'sin(x)','cos(x)') %легенда для графиков h1 - sin(x), h2 - cos(x)
title 'sin(x) and cos(x) functions'
xlabel 'x'
ylabel 'y'
```

Этот скрипт можно найти в папке с материалами для вашего проекта под именем EXAMPLE.м. Комментарии в коде выделяются символом %. На выходе при выполнении скрипта (**F5**)

получаем следующее:



Этот график выглядит уже заметно лучше, тут наглядно понятно, что есть что. Как сохранить этот график, а не делать скрины? Для этого выбираем в верхнем левом углу нашего окна с графиком **File > Save As...**

При выборе расширения сохраняемого графика, вы увидите множество вариантов. Какой же выбрать?

Во-первых, нужно определиться, какой тип графики вам нужен: растровый или векторный. Что это значит? Растровый формат графики – всем нам знакомые расширения типа .JPEG. Они хранят информацию в так называемых пикселах, соответственно при увеличении картинки в какой-то момент мы начнём видеть эти пикселы – квадратики, из которых состоят изображения. Качество картинки тем лучше, чем выше её разрешение (кол-во пикселов). Такая графика подходит, если нужно, например, распечатать полученный график и вклеить в отчёт по лабе для кафедры общей физики. Векторная же графика построена на так называемых кривых, которые при масштабировании не теряют в качестве. Такой тип графики используется, например, в научных статьях. Также, с картинками, сохранёнными в таком формате, можно работать в редакторах для векторной графики, например в ADOBE ILLUSTRATOR. Работу с векторной графикой можно обсудить со мной лично (для самых заинтересованных).

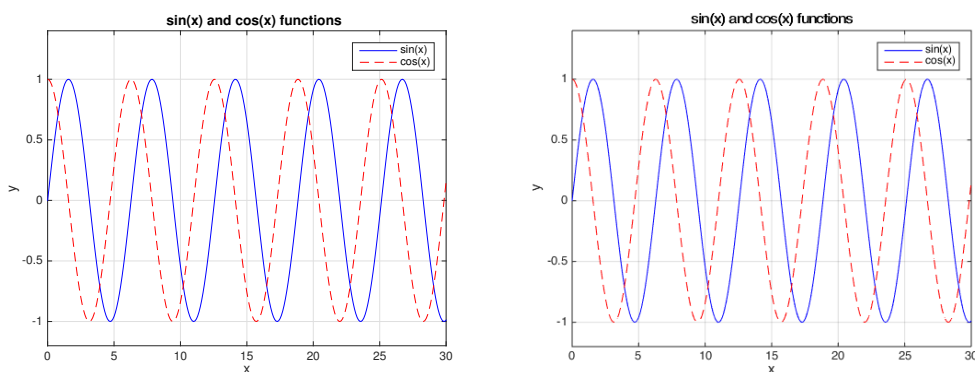
Во-вторых, для каждого типа графики существуют свои форматы: для растровой это, например .JPEG, .PNG, .BMP и другие; для векторной – .EPS, .PDF, .SVG... Обычно, когда речь идёт о векторной графике, я использую первые два формата: .EPS прекрасно подходит для дальнейшей работы с изображением в редакторах (а это нередко требуется), а график, сохранённый в .PDF в свою очередь удобно отправить кому-то для демонстрации.

Что касается растровой графики, по моему опыту, самый оптимальный формат сохранения – это .PNG. Так как .BMP достаточно много весит в силу специфики записи данных,

а .JPEG, сохранённый из MATLAB обычно выглядит как-то так:



А если серьёзно, то разницу между векторной и растровой графикой можно увидеть, приблизив два графика ниже:



Слева представлен график, сохранённый в формате .EPS, справа – в формате .PNG.

Также вы могли заметить, что самый первый формат, в котором MATLAB предлагает вам сохранить ваш график – это стандартный для него формат .FIG. Тогда у вас сохраняется график, при вызове которого снова открывается окно просмотра и все соответствующие его функции, о которых мы поговорим чуть позже.

Вернёмся теперь к нашему коду построения синусоиды и косинусоиды. После строчек с построением графиков функцией `plot` в коде не стоит `;`. Если посмотреть в **Command Window**, то там можно увидеть свойства наших графиков, которые помогут вам разобраться, как выполнять некоторые задачки повышенного уровня, и вообще крайне полезны для дальнейшей вашей работы.

4. Циклы и условия

С работой с циклами и условиями вы уже знакомы по другим языкам программирования. В MATLAB так же присутствуют конструкции типа `for`, `if`, `while`, `break`... Перечень можно посмотреть в уже упоминавшемся здесь MATLABREF.PDF в папке с вашим проектом.

Логические условия (например для конструкции `if ('условие')`) можно составлять как на основе `&&` и `||` – логических «и» и «или», так и сравнений `>`, `<`, `==`, `=`. Также допускается использование логических условий и сравнений, записанных в виде `.AND.`, `.OR.`, `.LEQ.`, `.NEQ.` и т.д. (аналогия с языком FORTRAN).

Часть II

Первый блок заданий

Правила оформления заданий для сдачи: задачи разделены по нескольким скриптам. Внутри каждого скрипта должен быть полный код, при запуске которого в воркспейсе выводятся **ТОЛЬКО** требуемые результаты. Промежуточные результаты выводиться не должны. Все скрипты должны быть названы именно так, как требуется. Задания повышенного уровня не являются обязательными, если вы не претендуете на окончательную оценку "отлично" по этому предмету. Если всё же вы желаете получить такую оценку, прошу выполнять эти задания, так как они требуют не очень долгой, но всё же самостоятельной работы по поиску информации о некоторых функциях и приёмах.

Все скрипты, а так же выходные файлы, которые получаются в некоторых задачах, должны быть объединены в один архив и присланы мне на электронную почту с темой "N-ый (первый в случае первого) блок заданий по проектной деятельности". Название архива должно начинаться с вашей фамилии латиницей, далее нижнее подчёркивание и цифра, означающая блок задания (1 в случае первого). Дальнейшее название файла может быть любым.

До дедлайна по сдаче можно обращаться непосредственно ко мне Вконтакте по поводу возникших вопросов. Задания за вас делать я, конечно, не буду, но постараюсь объяснить возможные непонятные моменты. Задания должны быть сданы до дедлайна, задания сданные по неуважительной причине после дедлайна не принимаются.

Дедлайн сдачи первого блока заданий: .

Моя электронная почта: `pechaeva.ab@phystech.edu`.

5. Задачи, обязательные для всех

1. Работа с числами и матрицами. Скрипт NUMBERS.M.

Задать любые три числа в переменные `a`, `b`, `c`. Вывести числа `d`, `e`, `f`, `g` – соответственно сумма всех чисел, произведение всех чисел, наибольшее из чисел и модуль разности `a` и `b`.

%% (Скрипт можно делить на части двойным знаком комментария, и выполнять части отдельно. Прошу так же, как тут, в ваших скриптах разделять таким символом подзадачи для удобства проверки.)

Задать матрицы (строки) X и $X0$ натуральных чисел от 10 до 100. "Перевернуть" одну матрицу от 100 к 10 вручную (цикл) и вторую с использованием команды сортировки. Вывести первый и последний элементы каждой матрицы после этого действия. Вывести размеры матриц.

%%

Задать случайную матрицу A размера 3×3 . Задать столбец B из трёх случайных элементов от 5 до 8. Решить уравнения $X1 \cdot A = B$ и $A \cdot X2 = B$ через операции $/$ и \backslash . Вывести результаты, в комментариях описать, в чём разница этих двух уравнений. Также решить уравнения $X3 \cdot A = B$ и $A \cdot X4 = B$ через обратные матрицы и умножение матриц. Сравнить полученные разными методами результаты.

%%

Задать через команды две матрицы: $Y1$ и $Y2$. Одна из них состоит из единиц, вторая – единичная (обе размерности 5). Записать в переменную $Y3$ произведение матриц (в любом порядке), а в переменную $Y4$ – матрицу, получившуюся поэлементным перемножением исходных матриц. Вывести детерминанты матриц $Y3$ и $Y4$. Затем записать вычисление матриц перемножения и поэлементного перемножения не через простейшие команды, а через циклы, сравнить результаты.

2. Работа с графикой 1. Скрипт GRAPHICS1.m.

Построить на одной фигуре графики функций модуль (красным), тангенс (чёрным), корня пятой степени (синим) на отрезке от -10 до 10 с шагом 0.5. Подписать оси, график, включить сетку и легенду. С выводом свойств каждого графика, постараться разобраться в этих свойствах (ваш главный помощник – **Help (F1)**). Сохранить фигуру в формате MATLAB FIGURE.

%%

Построить график кривой в пространстве, где по оси Ox – натуральные числа от 1 до 100, по оси Oy – синус, а по оси Oz – косинус. Подписать оси, график, включить сетку. Сохранить каждую из трёх проекций на плоскости в формате .PNG.

%%

Начертить красную пунктирную окружность в полярных координатах через `polar`.

%%

Задать две степенные функции: $y = x^2$ и $y = x^3$ на промежутке от 0 до 8, построить графики. Подписать оси, график, включить сетку и легенду. Толщина линий – 2.5 (толщина линий прописывается в функции `plot`). Не графически найти значение аргумента, при котором разность $x^3 - x^2$ принимает значение больше 100, провести в

этом месте вертикальную пунктирную линию толщины 1.5. Вывести ещё два таких же окна с графиками, но в пределах от 0 до пунктирной линии и от пунктирной линии до 10. Всего должно быть три окна с графиками, сохранить все три в формате .PDF.

3. Вычисление площадей фигур под графиком. Скрипт S-PLOTS.M.

Задать функцию

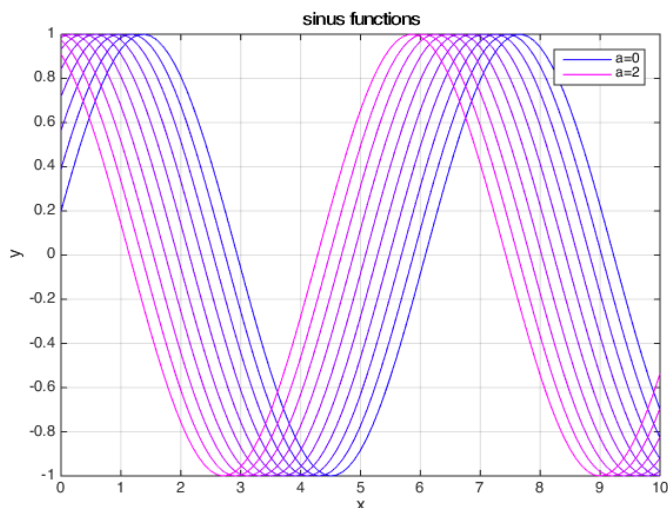
$$y = 5 \cdot \exp\left[-\frac{(x+3)^2}{4}\right]$$

от -5 до 5 с шагом 0.1. Посчитать площадь фигуры под этим графиком методом трапеций вручную и с помощью функции МАТЛАВ, результаты вывести. Также посчитать площадь под графиком методом Монте-Карло. Суть метода Монте-Карло можно почитать на википедии (статья «Метод Монте-Карло», раздел «Интегрирование методом Монте-Карло»). Сделать симуляцию для 100, 500 и 2000 точек, сравнить результаты с методом трапеций и объяснить (в комментариях).

6. Задачи повышенного уровня

1. Для предыдущей задачи в том же самом скрипте прописать анимацию выпадения 20 случайных точек на график. *Подсказка: использовать pause.*
2. Работа с графикой 1-adv. Скрипт GRAPHICS1-ADV.M.

Построить графики функций $y = \sin(x + a)$, где $a \in [0; 2]$. За шаг по a взять 0.25, т.е. всего должно быть 10 нарисовано 10 функций, пусть $a = 0$ соответствует синий цвет графика, $a = 2$ – розовый. Получить такой график (градиент):



Подписать оси, график, вывести сетку и легенду для конечных значений, сохранить график в формате .EPS. *Подсказка: использовать цикл.*

%%

(Задачка на фигуры Лиссажу и функцию сабплот)

Часть III

Ещё немного о работе с графикой и выводом данных

Часть IV

Второй блок заданий