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 MATLAB

Февраль 2019

Содержание

Вашим самым близким другом при работе с MATLAB должен стать раздел HELP, для поиска по которому выделите ваш запрос и нажмите кнопку F1. В хэлпе максимально ёмко описаны все команды и приведены примеры их использования. Также настоятельно рекомендую попутно с чтением методички держать открытой программу и вбивать туда всё, что вы видите в методичке, чтобы самостоятельно прочувствовать результат.

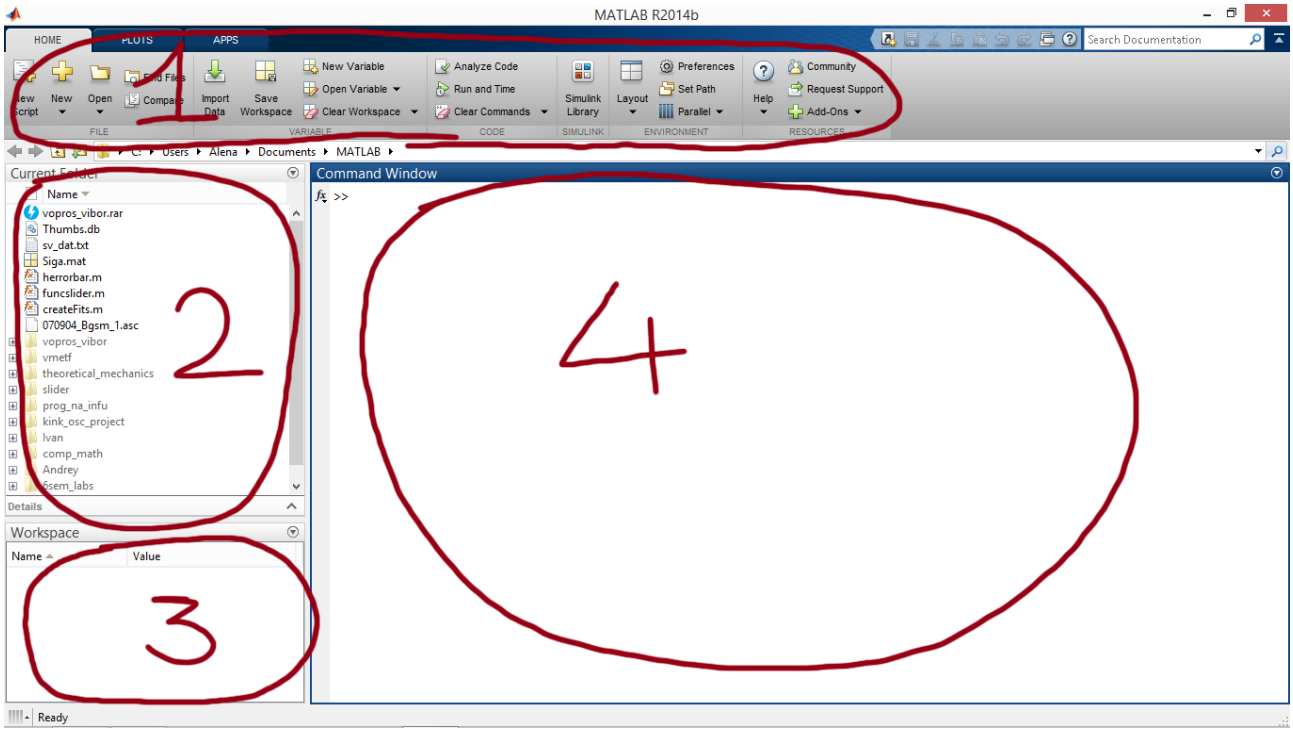
Часть I
Основные понятия

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, открывть 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 выводить.

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 опрерировать 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 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 высислений. 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 рассчетах, 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`. MATLAB 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 райтклиом на воркспейсе выберем **Clear Workspace**. Командное окно чистится командой `clc` 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]. Соотвественно `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: нарисуем график синусоиды, для этого зададим вектор иксов:

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

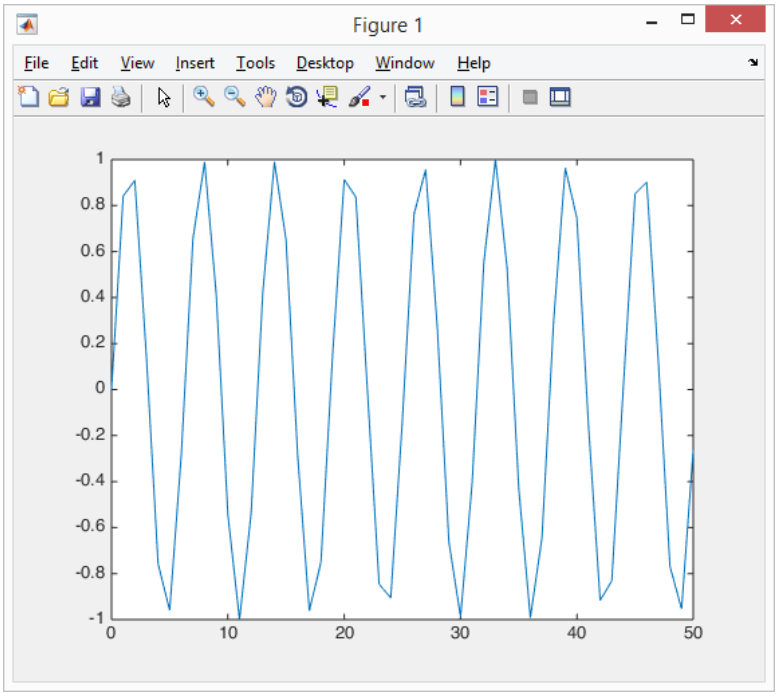
и зададим синусоиду:

```
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:



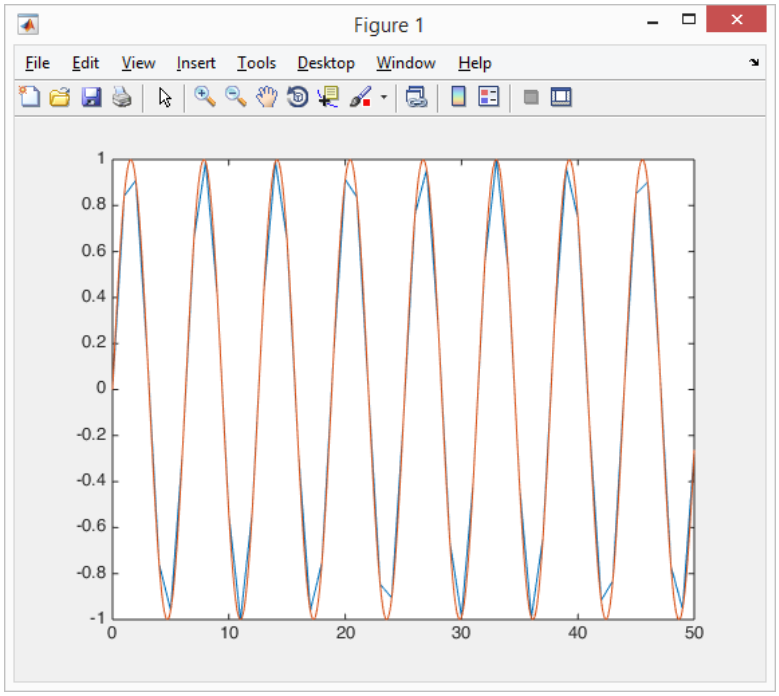
Как мы видим, это немного угловатый, но всё же синус. Закроем это окошко и зададим по оси x больше точек, чтобы наш синус был более плавным:

```
> > 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. Главное ничего не перепутать...

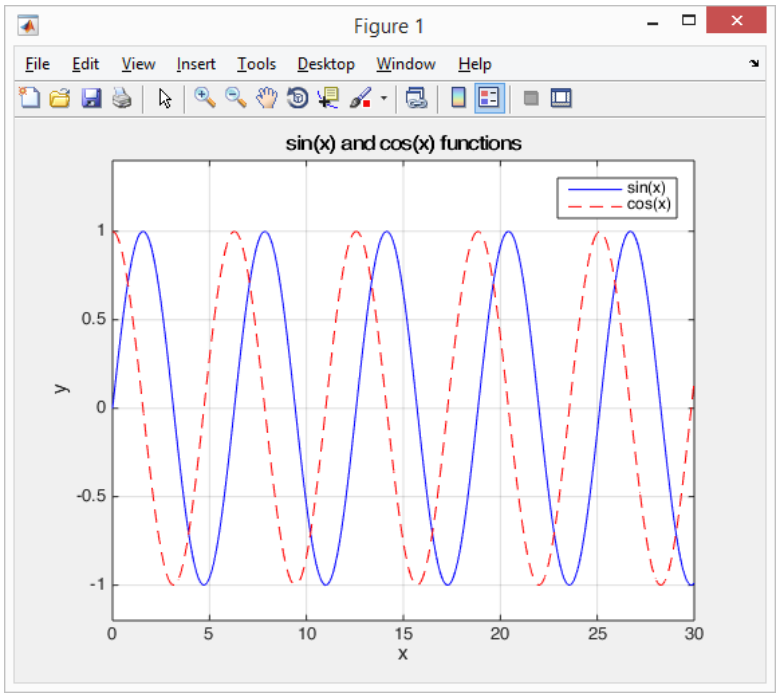
Но всё же наш график выглядит достаточно сыро, не так ли? Для этого создадим новый скрипт. Скрипты в MATLAB – это, по сути своей, маленькие автономные програмки. В командном окне работать неудобно, код нужно постоянно править, да и он не занимает две строчки, как если нужно использовать MATLAB исключительно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

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?

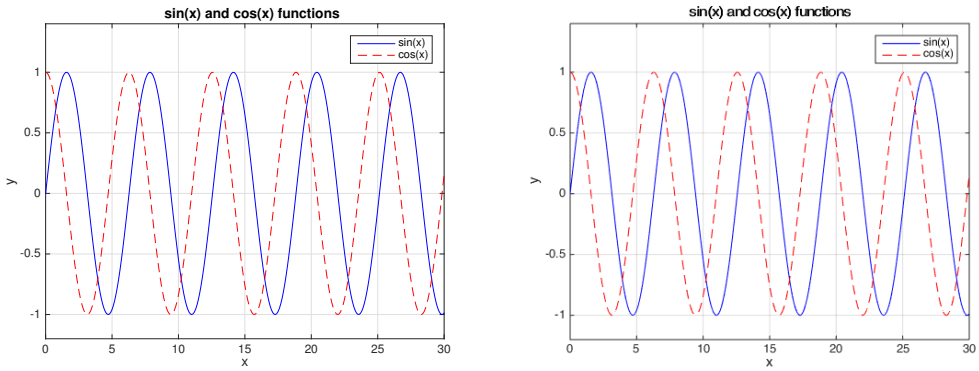
Во-первых, нужно определиться, какой тип графики вам нужен: растровый или векторный. Что это значит? Растровый формат графики – всем нам знакомые расширения типа .JPEG. Они хранят информацию в так называемых пикселах, соответственно при увеличении картинки в какой-то момент мы начнём видеть эти пикселы – 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 заинтрересованных).

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 избражением 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 представен 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 стандарный для него формат .FIG. Тогда у вас сохраняется график, при вызове которго снова открывается окно просмотра и все соответсвующие 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 присутсвуют 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, но желательны для выполнения, чтобы у нас с вами не возникло дальнейших разногласий при выставлении зачёта в конце семестра. Прошу всё же выполнять эти задания, так как они требуют не очень долгой, но всё же самостоятельной работы по поиску информации о некоторых фунциях и приёмах, это достаточно полезно для вас же самих. (То есть если мне вдруг будет казаться, что вашей работы в семестре не достаточно для выставления вам зачёта, в конце семестра вы, скорее всего, будете ~~долго и мутрно~~ досдавать мне задачи повышенного уровня, поэтому выполняя их в семестре, вы можете сэкономить себе время для подготовки к другим зачётам и экзаменам.)*

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-ный (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). Дальшнейнее 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: 23:59:59 17 марта 2019 года (время московское).

Моя электронная почта: nechaeva.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` – соответсвенно 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 рандомную матрицу `A` размера 3x3. Задать столбец `B` из трёх случайных элементов от 5 до 8. Решить уравнения `X1*A = B` и `A*X2 = B` через операции `/` и `\`. Вывести результаты, в комментариях описать, в чём разница этих двух уравнений. Также решить уравнения `X3*A = B` и `A*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 разница $x^3 - x^2$ 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

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

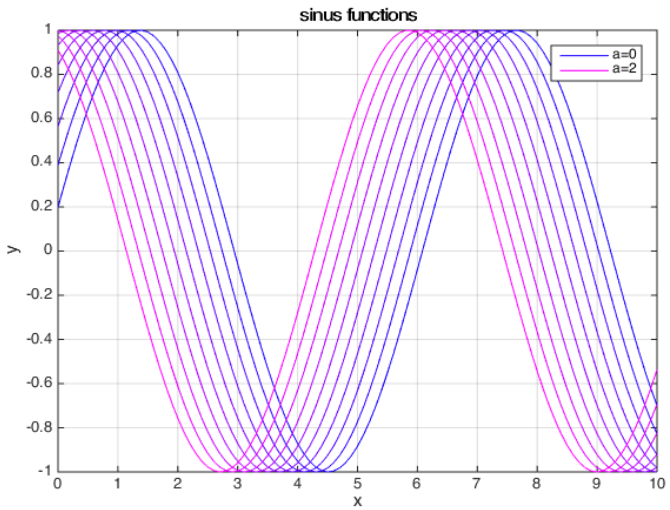
   от -5 до 5 с шагом 0.1. Посчитать площадь фигуры под этим графиком методом трапеций вручную и с помощью функции MATLAB, результаты вывести. Также посчитать площадь под графиком методом Монте-Карло. Суть метода Монте-Карло можно почитать на википедии (статя «Метод Монте-Карло», раздел «Интегрирование методом Монте-Карло»). Сделать симуляцию для 100, 500 и 2000 точек, сравнить результаты с методом трапеций и объяснить (в комментариях). Вывести в графическое окно рассматриваемую область, график функции красной линией толщины 2 и для случая 100 точек все эти точки должны быть также выведены маркером "точка" толщины 6.

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.**

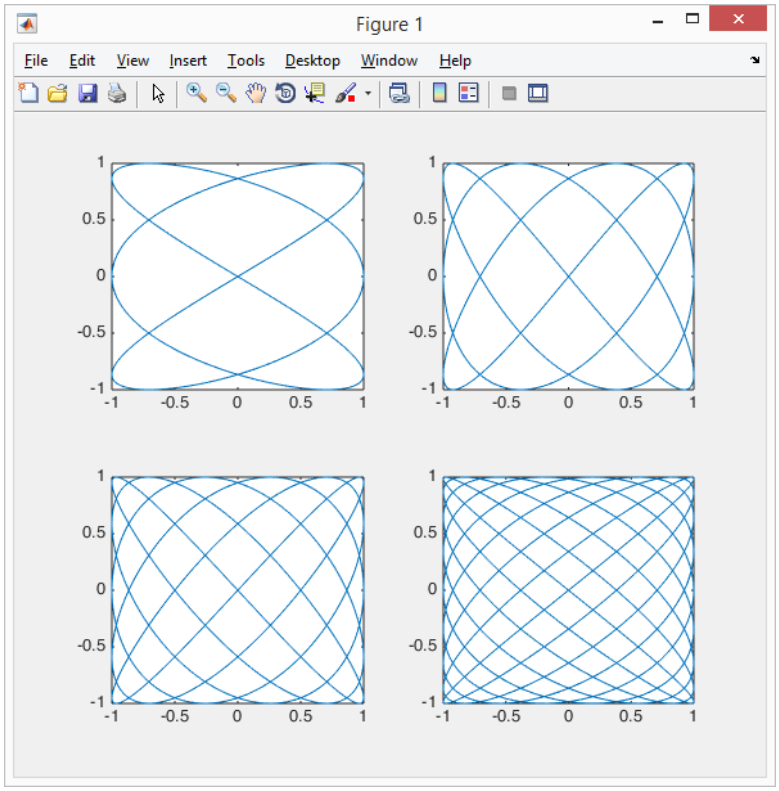
   Построить графики функций $y = sin(x + a)$, где $a \in [0; 2]$. За шаг по a взять 0.25, т.е. всего должно быть 10 нарисовано 10 функций, пусть $a = 0$ соотвествует синий цвет графика, $a = 2$ – розовый. Получить такой график (градиент):



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

%%

Нарисовать несколько фигур Лиссажу (как минимум 4) в одном графическом окне на подграфиках:



Здесь

$$x(t) = cos(at);$$
$$y(t) = sin(bt).$$

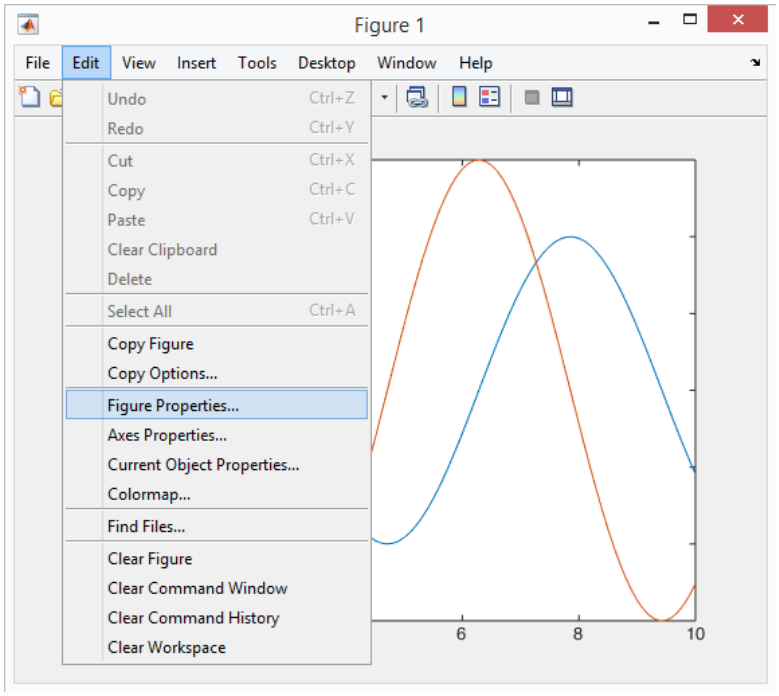
Подписать каждый из графиков, указать значения для a и b и сохранить в формате .PNG. Также сделать достаточно плавную анимацию изменения одной из частот (a или b) при неимзменной другой. Обязательно включить момент равных частот. *Подсказка: google it!*

Часть III
Ещё немного о работе с графикой и выводе данных

7. Изменение свойств фигуры

Задавать вид линий графика и подписывать оси в коде – это, конечно, дело хорошее, но что, если надо перепробовать кучу вариантов оформления и среди них найти тот, что выглядит нагляднее всего?

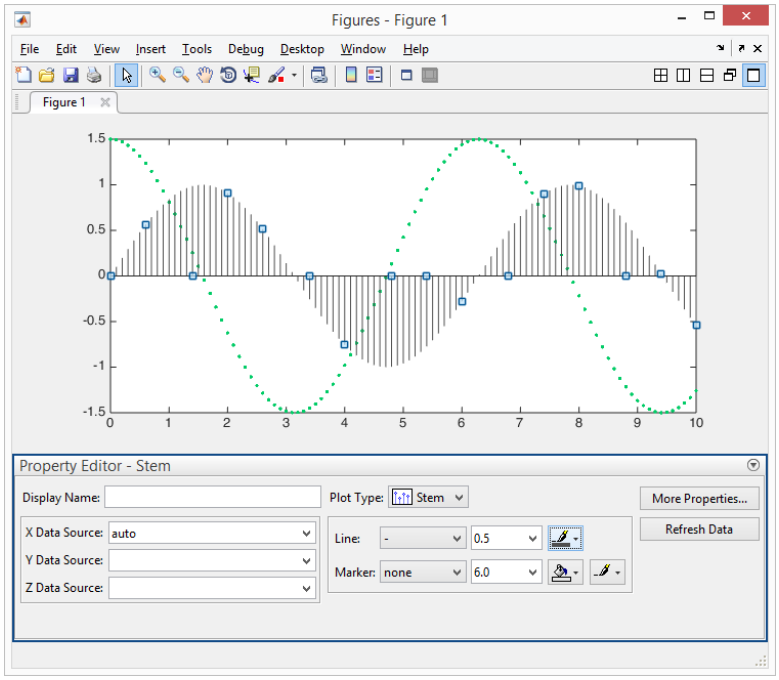
Для таких случаях легче всего использовать окошко изменения свойств фигуры. Рисуем простейший график синусоиды и косинусоиды без подписей осей, сетки и прочего и заходим в **Edit** > **Figure Properties...**:



И перед нами открывается множество дверей, точнее окошко для изменения нашей картинки. С ним полезнее всего иметь дело самостоятельно, исследуя всё методом тыка, но кое-что я всё же постараюсь показать. Нажимаем на одну из наших кривулин, и наблюдаем панель, где можно выбирать вид, толщину и цвет линии/маркеров, а так же вид самого графика:

Нажимая на оси, наблюдаем панель изменения графика, где можно добавлять сетку по осям, название, подписи осей, пределы осей и линейный/логарифмический вид. Также можно менять шрифт подписей и его размер. Также стоит помнить о том, что во всех текстовых полях используется теховский или латеховский интерпретатор. Чуть более расширенные функции (расположение, цвет фона или текста...) для изменения названия графика или подписей осей можно получить, кликнув на них непосредственно.

Также сверху можно залезть в раздел **Insert** и найти там много интересного: добавление



легенды (изменение подписей – даблклик на подписи), стрелок, линий, текста и прочее. Это всё очень полезные инструменты для того, чтобы сделать свой график наиболее наглядным для читателя. Но опять же, нужно соблюдать меру, если вы напихаете в вашу картинку много стрелок, сделаете сильно насыщенные цвета для графиков и подпишите это всё нечитаемым шрифтом, то это всё будет выглядеть как одно большое безумие. Пожалуйста, пощадите читателей!

8. Фитирование данных

Нет ничего понятнее, чем матлабовский хелп!
любой здравомыслящий человек

Очередной очень важный раздел, о котором можно говорить бесконечно, но я постараюсь наиболее ёмко и быстро описать самые базовые и нужные вещи.

При обработке данных, вы почти наверняка будете часто сталкиваться с задачей апроксимации полученного набора точек какой-то функцией. Вы, в принципе, уже сталкивались с этим не один раз на лабораторных работах, когда по каким-то экспериментальным данным вы определяли физические величины путём апроксимации вашего облака данных прямой или какой-либо степенной функцией (наиболее часто встречающиеся на кафедре общей физики зависимости). Однако всё может быть не так просто, как же тут быть?

Начнём с такого же наглядного инструмента, как описывался в прошлой секции. Но для начала зададим данные с шумами. Например, синусоиду:

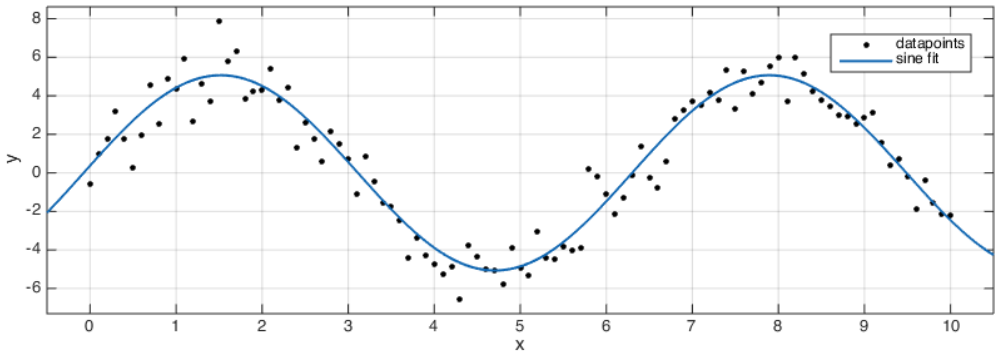
```
> > x = [0:0.1:10];
y = 5*sin(x) + randn(1,101);
```

Надеюсь, что после первой части этой методички всем понятно, что здесь написано. Вызываем один из моих самых любимых и наиболее часто используемых инструментов MATLAB – **Curve Fitting Tool**. Для этого просто пишем команду `cftool` в командное окно.

Слева сверху выбиваем наши матрицы `x` и `y` как X data и Y data. Получили на экране множество точек, как теперь апроксимировать его какой-либо функцией? Чуть правее от выбора набора данных для апроксимации есть блок с выбором функции апроксимации. Выбираем **Sum of Sine**. Чуть ниже можно выбрать, сколько слагаемых в нашей сумме, которой мы апроксимируем облако данных, будет. Выбираем 1 (что и стоит по умолчанию). Получаем на картинке апроксимацию наших точек синусоидой.

Как теперь узнать параметры нашей модели? Слева есть секция **Results**, где показана наша модель со всеми неизвестными параметрами, их значения и доверительные интервалы для них. Чуть ниже указываются коэффициенты, по которым можно судить о "хорошести" апроксимации данных нашей функцией. Если вы поменяете тип апроксимирующей функции, например, на полином 1 степени, то ваш **R-square** сильно упадёт. Оставляю задачу разобраться с этими коэффициентами заинтересованному читателю.

Теперь перед нами стоит вопрос, как же дальше работать с полученным графиком. Если такой вид графика вас устраивает, то можно просто нажать **File** > **Print to Figure**, сохранить полученный график и распечатать его, попутно вклеив в тетрадку с отчётом о выполненной лабораторной работе. Получится что-то такое:



Если же такой вид графика нас не устраивает, снова открываем **Edit** > **Figure Properties...** и развлекаемся.

Следующий уровень – это заполучить код нашей апроксимации. Но и это не так сложно, открываем **File** > **Generate Code** и видим перед собой описание функции, в котором также задачу разобраться с этим оставляю заинтересованному читателю (это полезно). Ещё раз напоминаю, что всю информацию по как-либо команде можно получить, выделив её, и нажав **F1**.

9. Вывод результатов в Command Window

Так как основные результаты выполнения работы выводятся для мониторинга в Command Window, было бы удобно научиться их понятно оформлять. Согласитесь, что вывод данных в виде:

```
old approach: period = 0.18664
new approach: period = 0.18719
difference is 0.29855%
```

Гораздо понятнее, чем вывод данных в виде трёх непонятно к чему относящихся чисел. Научимся так делать, поставим себе задачу вывести два числа, их НОД и НОК. Напишем для этого простой скрипт:

```
clear
a = 21;
b = 7;
X = ['greatest common divisor of a = ', num2str(a), ' and b = ', num2str(b), ' equals
', num2str(gcd(a, b)) ,', while least common multiple equals ', num2str(lcm(a, b))];
disp(X)
```

При выполнении которого получаем следующее:

```
greatest common divisor of a = 21 and b = 7 equals 7, while least common multiple
equals 21
```

Как можно догадаться, команда `disp(X)` выводит значение `X` без вывода имени переменной. В эту самую переменную можно пихать много чего, как, например, в нашем случае, текст и циферки. Чуть больше примеров можно посмотреть в разделе **Help** по `disp`.

10. Сохранение данных и вывод в файл

Как сохранить массивы данных, чтобы не вводить их вручную каждый раз?

Самый простой способ: сохранить их в матлабовском формате данных .MAT командой `save`. Этот способ очень удобен, чтобы сохранить результат какой-либо достаточно сложной программы по, например, предвратительной обработке полученного сигнала, а затем обращению к этому результату для дальнейшей с ним работы без запуска первой программы. Например, чтобы сохранить массив, напишем простейший код:

```
> > A = [0:10];
save('arrayA.mat','A')
```

В результате в списке наших файлов в папке появится файл ARRAYA.MAT, который содержит в себе переменную `A`. Также сохранить эту переменную можно, если кликнуть на её имени в списке переменных правой кнопкой и выбрать **Save As...**

Теперь поочистим все наши переменные в памяти командой `clear`. Чтобы снова загрузить из папки нашу переменную используем команду

```
> > load('arrayA.mat')
```

В списке переменных опять появился наш массив. Также загрузить переменную можно даблкликом по файлу с переменной.

Теперь поговорим о ситуации, когда нужно, например вывести массив данных в текстовый файл, чтобы потом преобразовать это в латеховкую табличку. Здесь также подойдёт знакомая нам команда `save`:

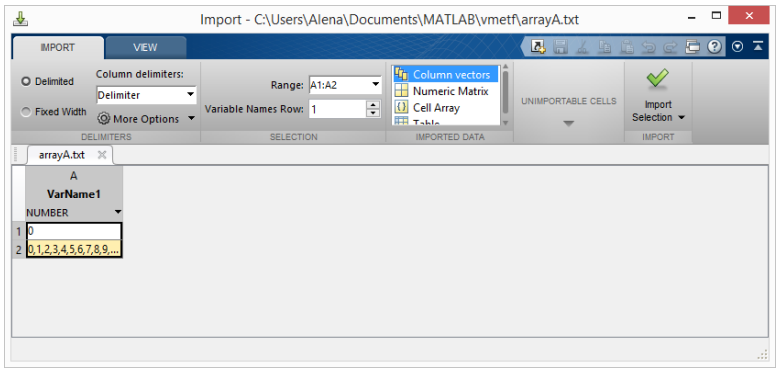
```
> > save('arrayA.txt','A','-ascii')
```

Результат работы данной команды можете проверить сами. Однако если мы захотим добавить в этот файл ещё одну переменную, нам придётся использовать какую-то другую команду (можете убедиться в том, что при записи другой переменной в тот же файл, содержимое файла переписывается полностью). С моей точки зрения, это удобно делать следующим образом:

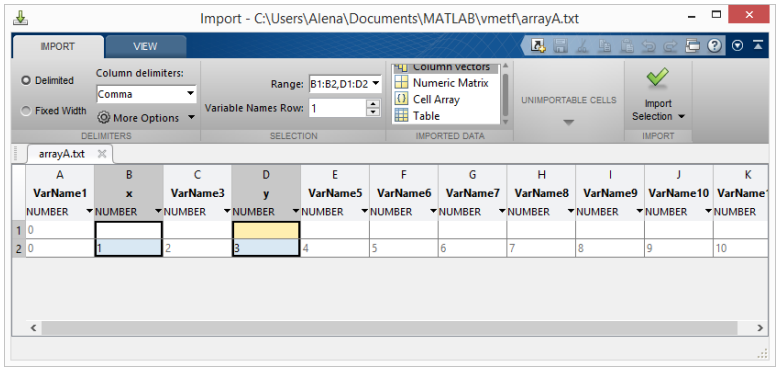
```
> > dlmwrite('arrayA.txt',data,'-append')
```

Здесь спецификация `'-append'` отвечает именно за добавление информации в файл к уже существующей, а не переписыванию. В этом можно убедиться, просто убрав эту спецификацию. Также эта команда очень удобна тем, что в ней можно задавать вид выводимых данных, вроде разделителя или количества знаков после запятой. Как это делать, можно найти во вкладке **Help**.

А теперь настало время упомянуть об очень удобном способе импорта сторонних данных. Нажимаем правой кнопкой на наш текстовый файл в разделе **Current Folder** и выбираем **Import Data...**, выплывает такое окошко:



На первый взгляд, в этом окошке прекрасно всё. Да и не на первый, потому что это действительно так. Здесь можно выбирать разделитель данных: табуляции, пробелы, запятые... Задавать пределы импорта данных (импортировать только одну колонку из многих), сразу в этом окошке называть выводимые переменные, выбирать их вид... В общем, простор для воображения. Например, если сменить разделители на запятые, для данных, которые имею я, и попросить MATLAB выводить мне только вторую и четвёртую колонку, обозвав их соотвественно как `x` и `y`, получаю:



Стоит помнить, что это далеко не все способы получения нужного результата, моя цель здесь – показать, что для работы в MATLAB существует множество различных инструментов, которые облегчают работу над какой-либо задачей. Ещё больше функций и примеров их использования вы можете найти, просто загуглив то, что вам нужно.

Часть IV
Второй блок заданий

Дедлайн сдачи второго блока заданий: 23:59:59 31 марта 2019 года (время московское).

11. Задачи, обязательные для всех

1. **Работа с графикой 2. Скрипт Graphics2.m.**

   Задать фунции

   $$y1 = sin(x) \text{ и } y2 = 0.1x$$

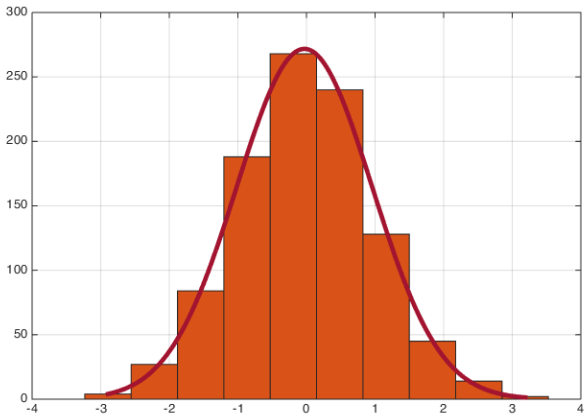
   от -5 до 15 по оси абсцисс. Вывести в командное окно (с текстом пояснения) координаты точек, в которых эти две функции пересекаются, начертить графики этих функций, подписать фигуру, подписать оси, включить сетку, по вкусу отрегулировать толщину графиков, на фигуре отметить стрелками все точки пересечения и подписать текстом их координаты. Сделать это максимально понятно и не налеплено друг на друга. Сохранить фигуру в формате .fig.

   %%

   Задать строку из 20 целых случайных чисел от 8 до 15. Построить гистограмму, как часто встречается каждое из чисел. В панели изменения фигуры задать переделы осей: $x \in [7; 16]$, $y \in [0; max + 1]$, где max – это значение по оси y для наиболее часто встречающегося числа. Включить сетку по оси ординат. Подписать оси, назвать график. Сменить цвет бинов на тёмно-зелёный. Сохранить в формате .png.

   %%

   Задать строку из 1000 случайных элементов, распределённых нормально. Построить гистограмму, апроксимировать её гауссианом и нарисовать эту апроксимацию поверх гистограммы. В результате должно получиться что-то такое:



Подписать оси, сделать сетку по оси ординат, вывести легенду, подписать график. Сохранить в формате .FIG.

%%

Загрузить из файла TABLE.TXT, находящегося в папке вашего проекта, 2 и 4 строчки как массивы `X` и `Y` соответственно (через `load`, любым способом, но никаких циклов). Сохранить их как X.MAT, Y.MAT. Найдите лучшую (на ваш взгляд) функцию, апроксимирующую данные, если известно, что она должна проходить через начало координат. Напишите вывод вашего решения с видом функции, которой вы апроксимируете данные, и значениями свободных параметров вашей модели. Сделать это всё с помощью функции `fittype` и специальных команд вывода свободных параметров модели, заданной через `fittype`. При желании также можно вывести доверительные интервалы для свободных параметров.

12. Задачи повышенного уровня

1. **Сборная солянка (из двух) задачек. Скрипт ADV2.M.**

Вывести в командное окно сообщение о том, что пользователю нужно ввести целое число, лежащий в пределах от 3 до 6 (в противном случае выводить сообщение об ошибке). Это число будет количество функций, которые нужно далее задать на промежутке от 0 до 10, вида:

$$y_i = a_i + b_i \sin(x) + c_i \sin^2(x) + d_i \sin^3(x),$$

где $i \in$ [3,6]. Коэффициенты должны генерироваться случайно в диапазоне от -1 до 1. Нарисовать на одной фигуре графики всех i функций и вывести общее число пересечений этих графиков на этом промежутке (также с поясняющим текстом). Число пересечений посчитать честно кодом, а не графически! *Подсказка: для ввода с клавиатуры использовать* `input`.

%%

Задать гиперболический параболоид (седло). Вывести две фигуры: поверхность как чёрную сетку без заливки и цветную поверхность без сетки. Задать для второй фигуры любую на выбор палитру и добавить цветовую шкалу. Подписать фигуры, оси. Все преобразования должны быть в коде, а не сделаны через **Figure Properties...** Объяснить, что делает `contour(X, Y, Z)`.