

## Лабораторная работа №6

### Графическое представление результатов вычислений при решении численных задач

*Цель:* Знакомство с графическими возможностями MATLAB, особенностями форматирования графиков. Визуализация результатов вычислений. Закрепление навыков по преобразованию типов данных, организации программ-сценариев, подпрограмм и организации диалогов.

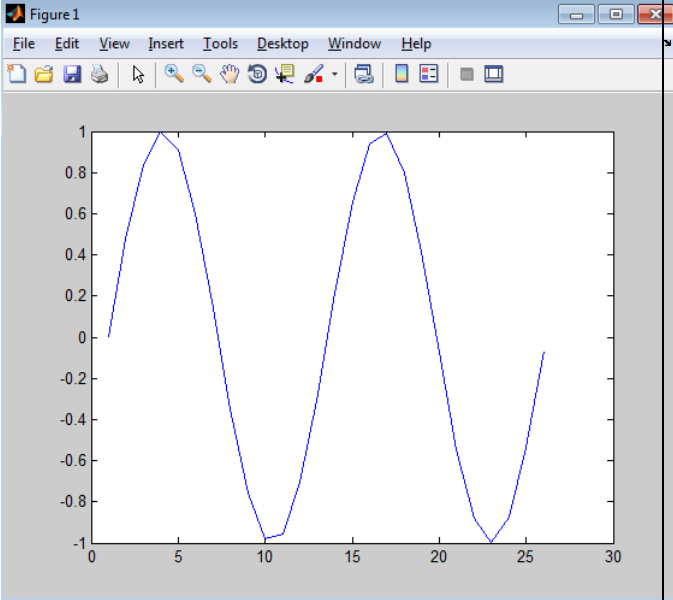
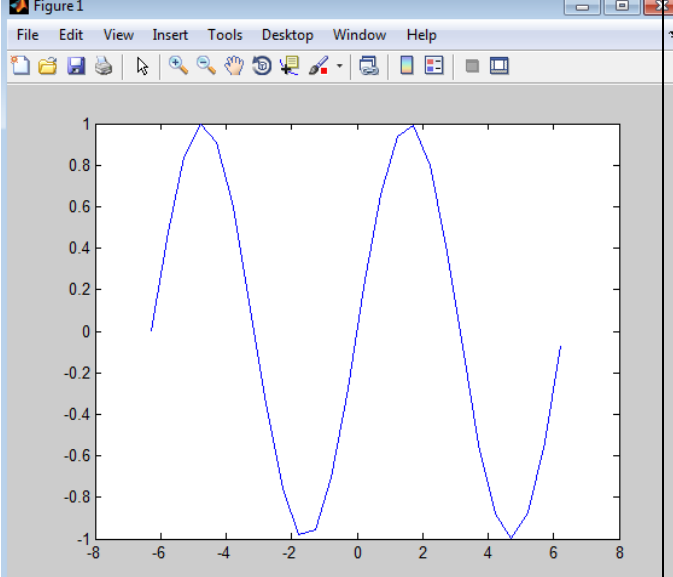
Одной из особенностей MATLAB являются простые и мощные возможности по построению графиков. MATLAB обладает набором встроенных функций для построения двумерных и трехмерных графиков, графиков в декартовых, полярных, логарифмических координатах, диаграмм, гистограмм, специальных графиков. Наиболее простой и в то же время чаще всего используемой является функция `plot`.

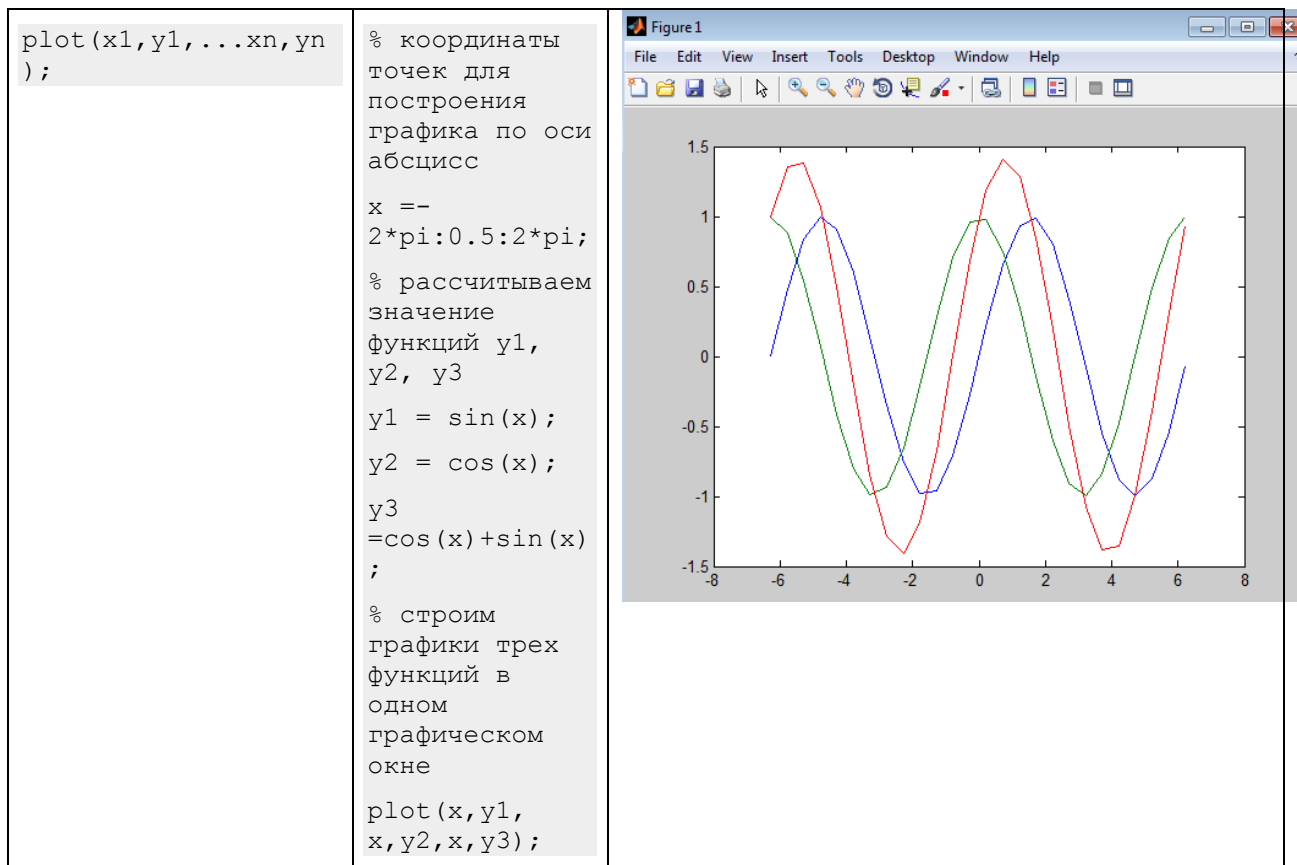
#### 1. Построение двумерного графика с помощью функции `plot`

Функция `plot` в наиболее простом случае может иметь один аргумент – вектор или матрицу. В таком случае график будет построен в зависимости от номера элемента. Функция `plot` может принимать 2 аргумента: два вектора одинаковой длины, задающие точки для построения графика. Первый аргумент – это координаты точек по оси абсцисс, а второй – соответствующие координаты по оси ординат. Задание аргументами функции `plot` нескольких пар векторов одинаковой длины позволяет построить несколько графиков в одном графическом окне. Примеры использования функции `plot` для разного количества аргументов представлены в таблице 1.

Таблица 1. Примеры использования функции `plot` для построения графиков без форматирования

Синтаксис функции	Пример программы	Результат работы программы

<pre>plot(y)</pre>	<pre>% задаем аргумент функции y  x =- 2*pi:0.5:2*pi;  % рассчитываем значения функции y y = sin(x);  % строим график y от номера элемента plot(y);</pre>	
<pre>plot(x, y);</pre>	<pre>% координаты точек для построения графика по оси абсцисс  x =- 2*pi:0.5:2*pi;  % координаты точек для построения графика по оси ординат y = sin(x);  % строим график по точкам, координаты которых содержатся в x, y plot(x, y);</pre>	 <p>График имеет такой же вид, как и в предыдущем случае, однако подписи оси абсцисс соответствуют заданным значениям аргумента, а не номеру элемента вектора.</p>



### 1.1. Установка параметров построения линий с помощью функции `plot`

#### Тип линии, тип маркеров, цвет линии и маркеров

Функция `plot` позволяет задать стиль линии на графике и стиль отображения точек, по которым график построен (тип маркера), а так же цвет линий и маркеров. Для этого после указания векторов аргумента и функции, по которым строится график, необходимо указать строку (или строковую переменную), содержащую специальные символы, описывающие особенности построения линии. Например, для прорисовки графика функции красной штрихпунктирной линией с маркерами в виде кружков необходимо задать:

```
plot(x,y,'-.or');
```

Основные символы, с помощью которых задаются тип линий и маркеров, а так же их цвет, представлены в таблице 2. Тип линии, маркера и цвет могут следовать в строке в любой последовательности.

По умолчанию, если не задано форматирование линий, график выводится сплошной линией без маркеров. Цвет линий выводится из первых шести, начиная с

синего.

Таблица 2. Символы, задающие типы линий, маркеров и их цвет в MATLAB

Тип линии		Тип маркера		Цвет линии и маркеров	
Символ	Значение	Символ	Значение	Символ	Значение
' - '	сплошная	' + '	знак плюс	' b '	синий
' -- '	штриховая линия	' o '	круг	' r '	красный
' : '	пунктирная линия	' * '	звездочка	' g '	зеленый
' - . '	штрихпунктирная линия	' . '	точка	' c '	голубой
' none '	нет линии	' x '	крестик	' y '	желтый
		' s '	квадрат	' m '	малиновый
		' d '	ромб	' w '	белый
		' ^ ', ' v ', ' > ', ' < '	треугольник	' k '	черный
		' p '	пентаграмма		
		' h '	гексаграмма		

### Толщина линий

Так же, при построении графика с помощью функции `plot` можно задать толщину линии. Для этого через запятую, после записи строки, определяющей тип и цвет линии и маркеров (если таковая присутствует), необходимо записать строку `'LineWidth'` и через запятую – скаляр, обозначающий необходимую толщину линии. По умолчанию толщина линии устанавливается равной 0,5 ед.

В качестве примера установки параметров линий при использовании функции `plot`, построим три графика в одном графическом окне. Первый график будет построен с настройками «по умолчанию», то есть сплошной линией синего цвета без маркеров, второй – пунктирной линией красного цвета с маркерами в виде кружков, третий – штрихпунктирной линией зеленого цвета с маркерами в виде звездочек. Линии будут прорисованы толщиной 2 ед.

```
>> x = -1 : 0.1 : 1;
>> plot(x, sin(x), x, sin(x+pi/2), '--or', x, sin(x-pi/2), 'gx-.',
'LineWidth', 2);
```

Созданный график будет выглядеть следующим образом:

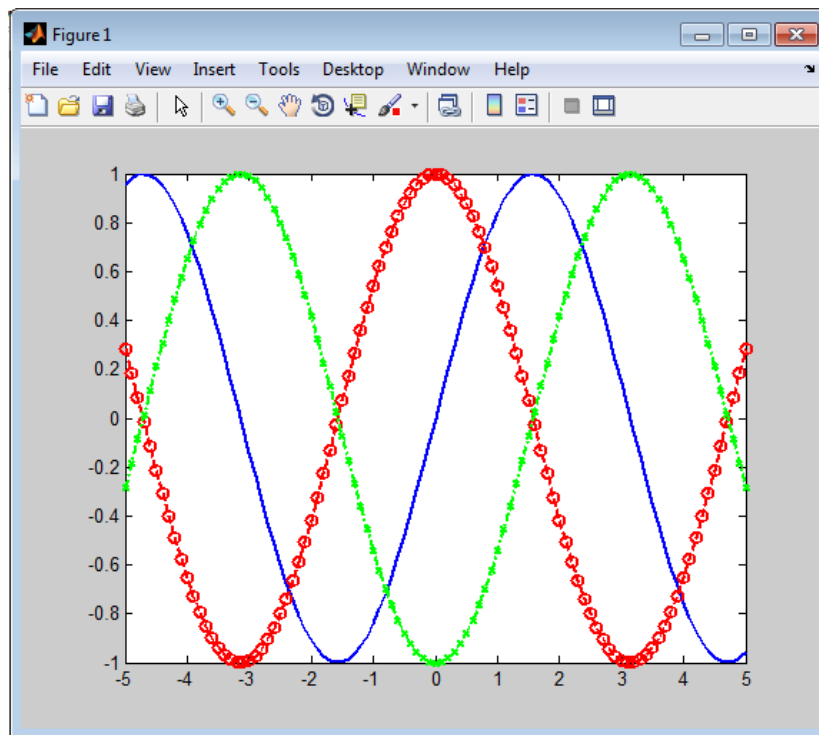


Рисунок 1

## 2. Оформление графиков

Кроме форматирования стиля линий в MATLAB можно дополнительно оформлять графики, подписывать их, подписывать оси и т.д.

Для создания заголовка текущего графика используется функция `title`, принимающая заголовок в виде строки и выводящая его над графиком, например:

```
>>title('Exaplegraphic');
```

Для того чтобы подписать ось абсцисс и ось ординат используются функции `xlabel` и `ylabel` соответственно, которые принимают название осей, например:

```
>>xlabel('x');
>>ylabel('y');
```

Функция `legend` выводит легенду графика. Данную функцию удобно использовать, когда в одной и той же системе координат строится несколько графиков. В качестве аргументов данной функции используются строки с текстом, который будет отображаться в легенде. Подписи линий необходимо задавать в порядке их построения. Например:

```
legend('y1=sin(x)', 'y2=cos(x)');
```

Для включения сетки на графике используют команду `grid on`, которая добавит

сетку на текущий график:

```
>>grid on
```

В MATLAB так же можно менять масштаб построения графика. По умолчанию MATLAB автоматически выбирает масштаб графика, однако в некоторых случаях необходимо задать масштаб вручную. Для этого используются функции `xlim` и `ylim`, аргументом которых является вектор из двух элементов, указывающий минимальное и максимальное значения для соответствующей оси, например:

```
xlim([-100 100]);
```

В данном случае график будет построен для значения аргумента от -100 до 100.

### 3. Вывод нескольких графиков в текущее окно

Для визуального сравнения нескольких графиков удобно строить несколько графиков в пределах одного окна и одной и той же системе координат. Как было показано ранее, функция `plot` позволяет это реализовать, используя запись `plot(x1,y1,...,xn,yn)`. Однако, такая запись не всегда является удобной для использования, особенно в тех случаях, когда в одной системе координат необходимо построить графики, используя различные функции MATLAB (например, функцию `ezplot`, позволяющую строить графики символьно заданной функции, или функцию `stem`, строящую график в виде линий, начинающихся на оси абсцисс и заканчивающихся кружком в соответствующем значении функции для данного аргумента). В таких случаях удобно пользоваться командами `hold`, `hold on`, `hold off`, `holdall`.

Команда `hold on` включает режим сохранения графиков функций в текущем графическом окне, а команда `hold off` отключает его.

Так, для того чтобы добавить еще один график необходимо использовать команду `hold on`, после чего снова воспользоваться функцией `plot`, либо же сначала использовать команду `hold on`, после чего перечислить все графики, которые необходимо построить в одной системе координат. Например:

```
>> x = -10:0.1:10;  
>> y1 = sin(x);  
>> y2 = (x / 10) .^3;  
>>plot(x, y1);  
>> hold on  
>>plot(x, y2);
```

В результате выполнения этих команд сформируется следующий график:

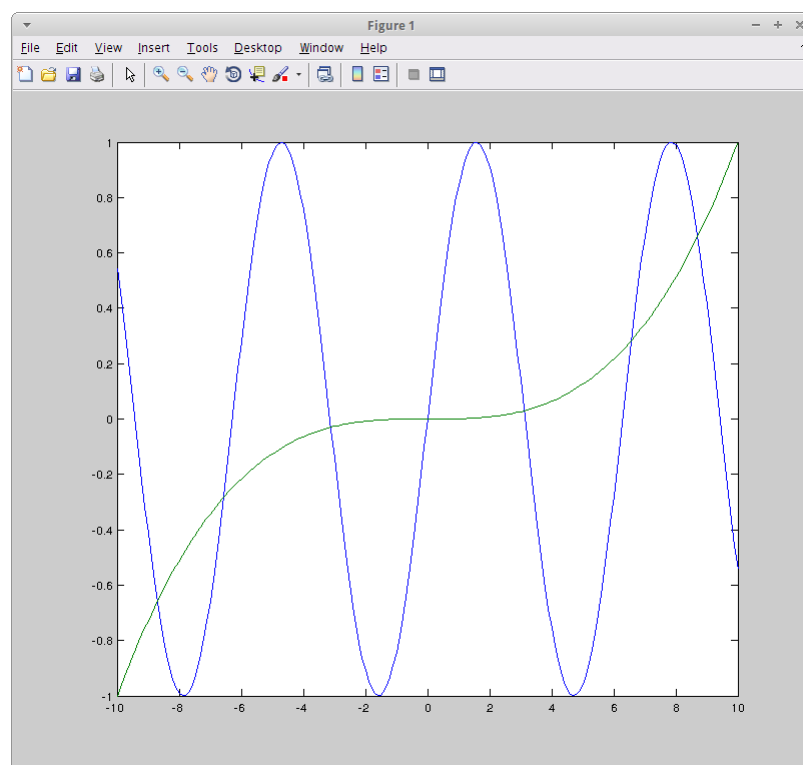


Рисунок 2

Отметим, что если не использовать команду `hold on`, то последний вызов функции `plot` затрет все предыдущие графики в текущем окне.

Команда `hold off` отключает режим построения графиков в одной системе координат. Так, если после команды `hold off` использовать функцию `plot`, будет создано новое графическое окно, а старое будет автоматически закрыто.

Для того, чтобы закрыть графики, в MATLAB предусмотрена функция `close`:

```
>>close
```

Вызов данной функции без параметров закроет все открытые окна графиков.

#### **4. Разбиение графического окна и вывод нескольких графиков с различными системами координат в одном графическом окне**

MATLAB позволяет выводить в одном графическом окне несколько систем координат. Например, для описания движения материальной точки могут быть использованы функции пути, скорости и ускорения, которые обычно строят один под другим.

Для разбиения графического окна в MATLAB используют функцию `subplot`. Функция `subplot` выполняется перед обращением к функциям построения графиков и в самом простом случае имеет синтаксис `subplot(m, n, p)`, где  $m$  – количество подокон, на которое разбивается графическое окно по горизонтали,  $n$  – количество подокон, на которое разбивается графическое окно по вертикали,  $p$  – номер подокна, в котором будет выводиться очередной график.

Для примера построим в одном графическом окне друг под другом функции пути, скорости и ускорения двух материальных точек от времени. Для первой материальной графики будем строить красной сплошной линией толщиной 2 ед., а для второй – малиновой пунктирной линией толщиной 1 ед. Графики и оси подпишем. Перемещение первой материальной точки описывается функцией  $S_1 = \sin(2t)$ , тогда скорость  $V_1 = S_1' = 2\cos(2t)$ , а ускорение  $a_1 = S_1'' = -4\sin(2t)$ . Для второй материальной точки  $S_2 = \cos(2t)$ ,  $V_2 = S_2' = -2\sin(2t)$ , а ускорение  $a_2 = S_2'' = -4\cos(2t)$ . Текст программы, реализующей построение таких графиков, приведен ниже.

```
% Задаем вектор времени
t=0:0.1:20;
% Рассчитываем значения пути для материальных точек
S1=sin(2*t);
S2=cos(2*t);
% Рассчитываем значения скорости материальных точек
V1=2*cos(2*t);
V2=-2*sin(2*t);
% Рассчитываем значения ускорения материальных точек
a1=-4*sin(2*t);
a2=-4*cos(2*t);
% Строим 3 графика в разных системах координат в одном
графическом окне
% для чего разбиваем графическое окно на 3 подокна,
% 3 по горизонтали (аналог числа строк) и 1 по вертикали (аналог
числу
% столбцов)

% В первом окне в верхней трети графического окна строим графики
пути от времени
subplot(3,1,1)
% Включаем функцию построения нескольких графиков в одной
системе координат
hold on
% Строим график для первой материальной точки
plot(t, S1, 'r', 'LineWidth',2);
% Строим график для второй материальной точки
plot(t, S2, 'b--', 'LineWidth',1);
% Подписываем график
title('Зависимость пути от времени');
```



```

% Подписываем оси
xlabel('Время, с');
ylabel(' Путь, м');
% Включаем сетку
grid on
% Добавляем легенду
legend('мат. точка 1', 'мат. точка 2');
% Выключаем функцию построения нескольких графиков в одной
системе координат
hold off

% Во втором окне посередине графического окна строим графики
скорости от времени
subplot(3,1,2)
% Включаем функцию построения нескольких графиков в одной
системе координат
hold on
% Строим график для первой материальной точки
plot(t, V1, 'r', 'LineWidth',2);
% Строим график для второй материальной точки
plot(t, V2, 'b--', 'LineWidth',1);
% Подписываем график
title('Зависимость скорости от времени');
% Подписываем оси
xlabel('Время, с');
ylabel(' Скорость, м/с');
% Включаем сетку
grid on
% Добавляем легенду
legend('мат. точка 1', 'мат. точка 2');
% Выключаем функцию построения нескольких графиков в одной
системе координат
hold off

% Во третьем окне внизу графического окна строим графики
ускорения от времени
subplot(3,1,3)
% Включаем функцию построения нескольких графиков в одной
системе координат
hold on
% Строим график для первой материальной точки
plot(t, a1, 'r', 'LineWidth',2);
% Строим график для второй материальной точки
plot(t, a2, 'b--', 'LineWidth',1);
% Подписываем график
title('Зависимость ускорения от времени');
% Подписываем оси
xlabel('Время, с');
ylabel(' Ускорение, м/с^2');
% Добавляем легенду
legend('мат. точка 1', 'мат. точка 2');
% Включаем сетку
grid on

```

```
% Выключаем функцию построения нескольких графиков в одной  
системе координат  
hold off
```

В результате работы программы будет выведено следующее графическое окно.

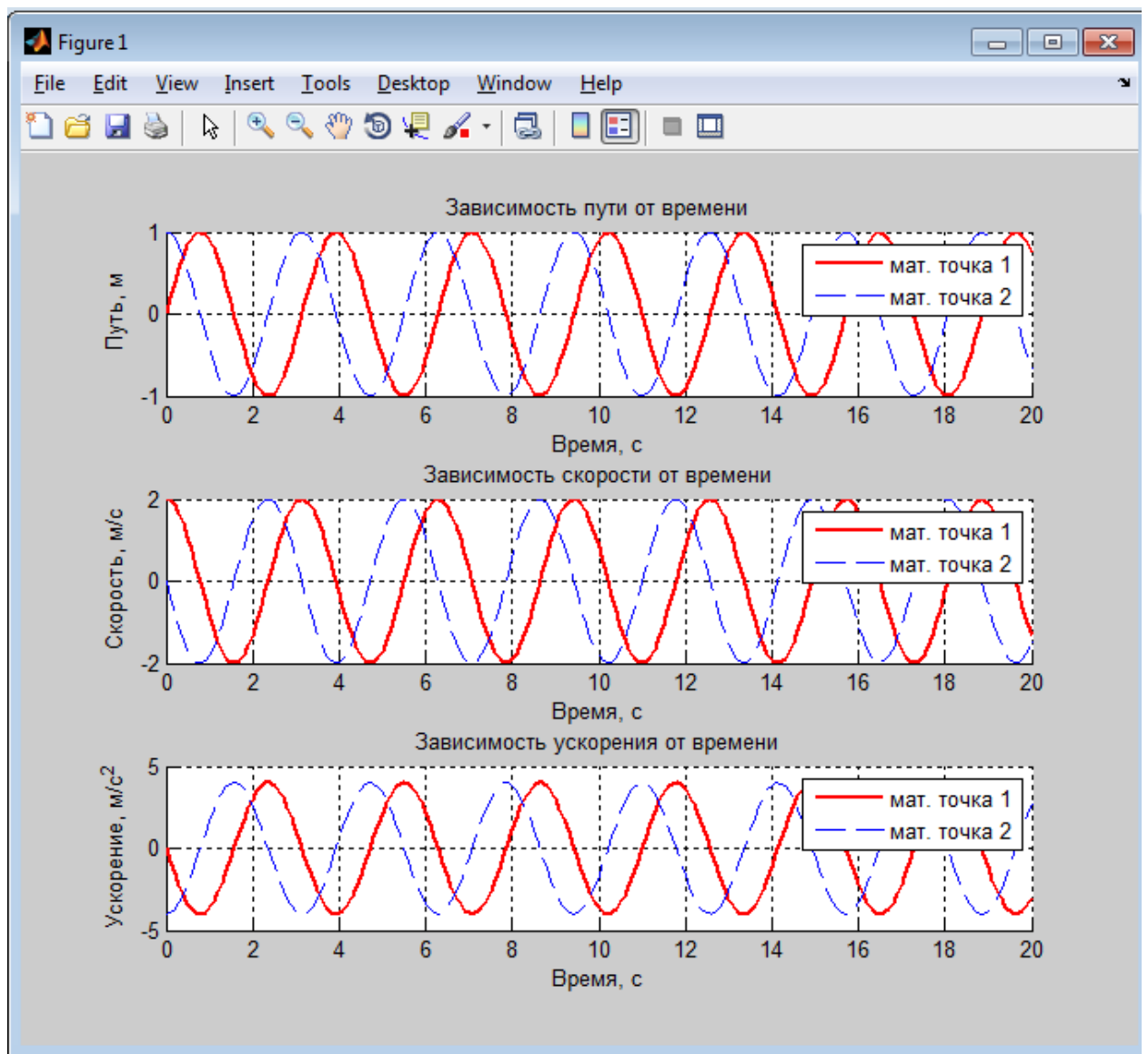


Рисунок 3

## 5. Функция eval

Для выполнения команд MATLAB их необходимо записать в m-файл или ввести в командное окно MATLAB, однако иногда бывает необходимо выполнить некоторые команды или выражения, которые заранее неизвестны, и например, вводятся пользователем во время работы программы. Для решения подобных проблем можно использовать функцию `eval`, которая имеет следующий синтаксис:

```
eval('выражение')
```

Функция `eval` принимает в качестве входного аргумента строку, которую интерпретирует как выражение MATLAB и вычисляет его.

Так например, данная функция будет полезна, если необходимо, чтобы пользователь программы ввел некоторую функцию, которая будет участвовать в работе программы. В качестве примера приведем небольшую функцию, которая предлагает пользователю ввести функцию и интервал для построения ее на графике, и на основе введенных данных строит график:

```
Function simplePlotDialog()
% функция для организации простого диалога для построения
графика

fprintf('Построение графика функции\n');

% попросим пользователя ввести функцию
% корректность введенных данных проверять не будем
fBody = input('Введите функцию: f(x) = ', 's');

% тело функции введено пользователем, теперь
% осталось составить саму функцию
% как помните, в лабораторной работе №2 упоминались анонимные
функции,
% для объявления которых использовалось выражения типа:
% переменная = @(список_аргументов) тело_функции;
% так как тело функции уже введено пользователем, то
% создадим анонимную функцию, которую используем
% для вычисления точек графика
f = eval(['@(x)' fBody]);
% помните что строки это векторы-строки, поэтому
% для их конкатенации (объединения строк)
% можно использовать выражение:
% [строка_1 строка_2]

fprintf('Введите диапазон в которых будем строить график\n');

% ввод нижней границы диапазона
xMin = NaN;
```

```

while isnan(xMin)
xMin = input('Нижняя граница диапазона: ');
    % проверка, что xMin это число
if ~isscalar(xMin) || isnan(xMin) || ~isreal(xMin)
xMin = NaN;
fprintf('Ошибка, некорректные данные\n');
end
end

% ввод верхней границы диапазона
xMax = NaN;
while isnan(xMax)
xMax = input('Верхняя граница диапазона: ');
    % проверка, что xMin это число
if ~isscalar(xMax) || isnan(xMax) || ~isreal(xMax)
xMax = NaN;
fprintf('Ошибка, некорректные данные\n');
end
end

% подготовка данных к построению графика
x = xMin : (xMax-xMin) / 200 : xMax;
y = zeros(size(x));
for i = 1:length(x);
y(i) = f(x(i));
end

% строим график
plot(x, y);
xlabel('x');
ylabel('y');
xlim([xMin xMax]);
title(['f(x) = ' fBody]);
grid on;

end

```

Приведем пример работы данной программы:

```
>>simplePlotDialog
```

Построение графика функции

Введите функцию:  $f(x) = \sin(x) / x$

Введите диапазон в котором будем строить график

Нижняя граница диапазона: [3 3]

Ошибка, некорректные данные

Нижняя граница диапазона: -30

Верхняя граница диапазона: 30

```
>>
```

В результате выполнения программы появится следующий график:

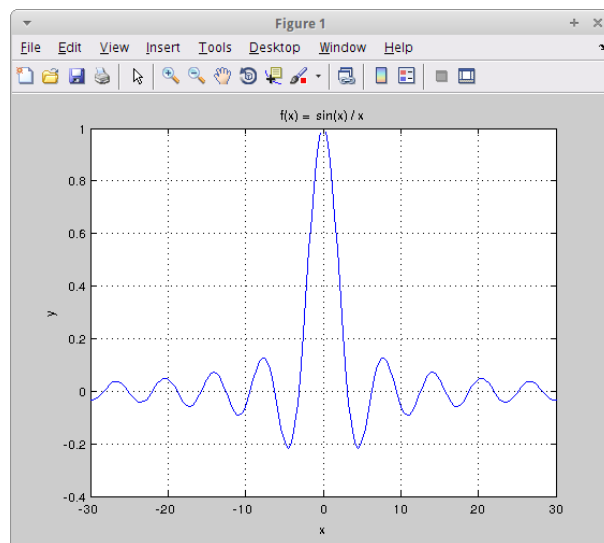


Рисунок 4

## 6. Аппроксимация данных с помощью функции `interpft`

Аппроксимацией называют научный метод, состоящий в замене одних объектов другими, в каком то смысле близкими к исходным, но более простыми. Например, косинус можно аппроксимировать параболой, а экспоненту – линейной функцией.

Частный случай аппроксимации – интерполяция. Интерполяцией называют такую разновидность аппроксимации, при которой кривая построенной функции проходит точно через имеющиеся точки данных. Так же интерполяция – способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений.

В MATLAB существует множество функций, производящих аппроксимацию и интерполяцию данных. Одной из них является функция `interpft`, производящая Фурье-интерполяцию периодических функций. Функция `interpft` имеет следующий синтаксис:

`y1=interpft(y,n)`, где `y` - вектор значений функции, `n` – количество точек для аппроксимирующей кривой (вектор `y1` будет иметь `n` элементов, интерполирующих вектор `y`). Пример использования функции и построения интерполирующей кривой и точек, для которых производилась интерполяция, приведен ниже.

```
% Задаем вектор y, для которого будем производить интерполяцию
y=[0.1767;0.1263;0.0151;0.0431;-0.2045;0.2684;0.5152;0.1412;-
0.0559;0.2845];
% Задаем количество точек интерполирующего вектора
n_app=100;
% Проводим интерполяцию и запоминаем ее в переменную y1
y1=interpft(y,n_app);
% Вычисляем шаг интерполирующей функции
dx=length(y)/n_app;
% Задаем аргумент для построения интерполирующей функции y1
x1=0:dx:length(y)-dx; %-dx, для того, чтоб
сопало количество точек
% Задаем аргумент для построения функции
x=0:length(X1)-1;
% Строим в рамках одного окна график интерполирующей функции и
% точки вектора y
hold on
plot(x1,y1, 'LineWidth',2);
plot(x,y, 'dr', 'LineWidth',2);
grid on
```

В результате работы программы будет выведен график

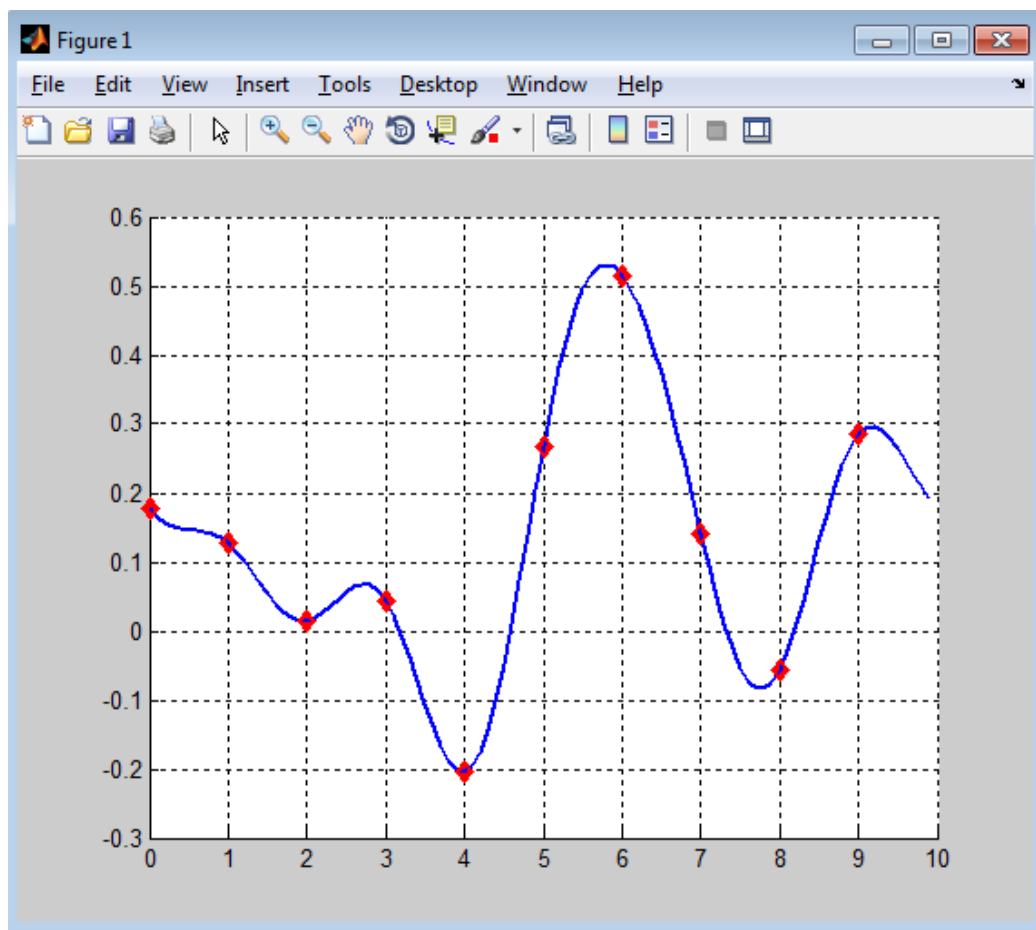


Рисунок 4

## Задание на лабораторную работу №6

Дополнить программу, реализованную в лабораторной работе №6, графическим представлением решения.

Программа должна запросить у пользователя математическую функцию, запросить интервал для построения графика заданной пользователем функции и проверить введенные значения согласно условию 1 и условию 2 (используя ранее написанные функции для лабораторной работы №5). Программа должна рассчитать значение интеграла  $\int_a^x f(x)dx$ , где  $f(x)$  – функция, введенная пользователем,  $a$  – нижняя граница интервала для построения графика,  $x$  – текущее значение аргумента,  $x \in [a, b]$ . Интеграл рассчитать любым удобным методом и вывести результаты расчетов в виде таблицы с дискретными данными с 3 столбцами (аргумент, функция, интеграл), а так же в виде графика. Для вывода таблицы использовать не более 15 строк, охватывающих всю ОДЗ с одинаковым шагом. Графическое окно должно быть разбито на два подокна, расположенных горизонтально или вертикально в зависимости от номера варианта (см. табл. 3). На графике функции указать маркерами точки, по которым строился график (для наглядности допускается прорисовывать точки с большим шагом). Стили линий и маркеров, их цвет, толщина выбирается в соответствии с номером варианта (см. табл. 3). На графике тонкими горизонтальными пунктирными линиями отметить максимальное и минимальное значение функции.

На графиках прорисовать сетку. Все графики и оси должны быть подписаны.

## Требования к содержанию отчета

Отчет по лабораторной работе оформляется в любом текстовом редакторе и предоставляется в печатном виде. Отчет должен состоять из следующих разделов:

1. Титульный лист. На титульном листе необходимо указать номер и название лабораторной работы, номер варианта, ФИО и группу исполнителя, ФИО преподавателя.
2. Цель работы.
3. Задание на лабораторную работу в соответствии с номером варианта.
4. Ход работы:
  - листинги основной программы и подпрограммы, реализующей вывод графика;
  - блок-схемы алгоритмов основной программы и подпрограммы, реализующей вывод графика \*;



- результаты работы программы с демонстрацией ошибки.
5. Выводы по работе.
- К отчету прилагаются:
- файл с текстом отчета;
  - m-файлы с текстом программы и подпрограмм
- Файлы отчета необходимо разместить в личном кабинете

Таблица 3. Задание на лабораторную работу №8

№ В-та	Расположение подокон	Первый график					Второй график		
		Тип линии	Тип маркера	Цвет линии	Цвет маркера	Толщина линии	Тип линии	Цвет линии	Толщина линии
1	Гориз.	сплошная	знак плюс	синий	красный	1	пунктирная линия	Красный	2,5
2	Вертик.	штриховая линия	круг	красный	синий	1,5	штрихпунктирная линия	Синий	1,25
3	Гориз.	пунктирная линия	звездочка	зеленый	красный	2	сплошная	Голубой	2
4	Вертик.	штрихпунктирная линия	точка	голубой	зеленый	2,5	штриховая линия	Зеленый	2,25
5	Гориз.	сплошная	крестик	желтый	голубой	1,25	пунктирная линия	Малиновый	1,5
6	Вертик.	штриховая линия	звездочка	синий	желтый	2	штрихпунктирная линия	Красный	1,75
7	Гориз.	пунктирная линия	квадрат	малиновый	зеленый	2,25	сплошная	Синий	1
8	Вертик.	штрихпунктирная линия	ромб	синий	малиновый	1,5	штриховая линия	Голубой	1,5
9	Гориз.	сплошная	треугольник	красный	зеленый	1,75	пунктирная линия	Зеленый	2

10	Вертик.	штриховая линия	знак плюс	зеленый	синий	1	штрихпунктир ная линия	Малиновый	2,5
11	Гориз.	пунктирная линия	круг	голубой	черный	1,5	сплошная	Красный	3
12	Вертик.	штрихпунктир ная линия	звездочка	желтый	красный	2	штриховая линия	Синий	2
13	Гориз.	сплошная	точка	голубой	желтый	2,5	пунктирная линия	Голубой	2,5
14	Вертик.	штриховая линия	крестик	малиновый	синий	3	штрихпунктир ная линия	Зеленый	1,25
15	Гориз.	пунктирная линия	кружок	синий	малиновы й	2	сплошная	Малиновый	2