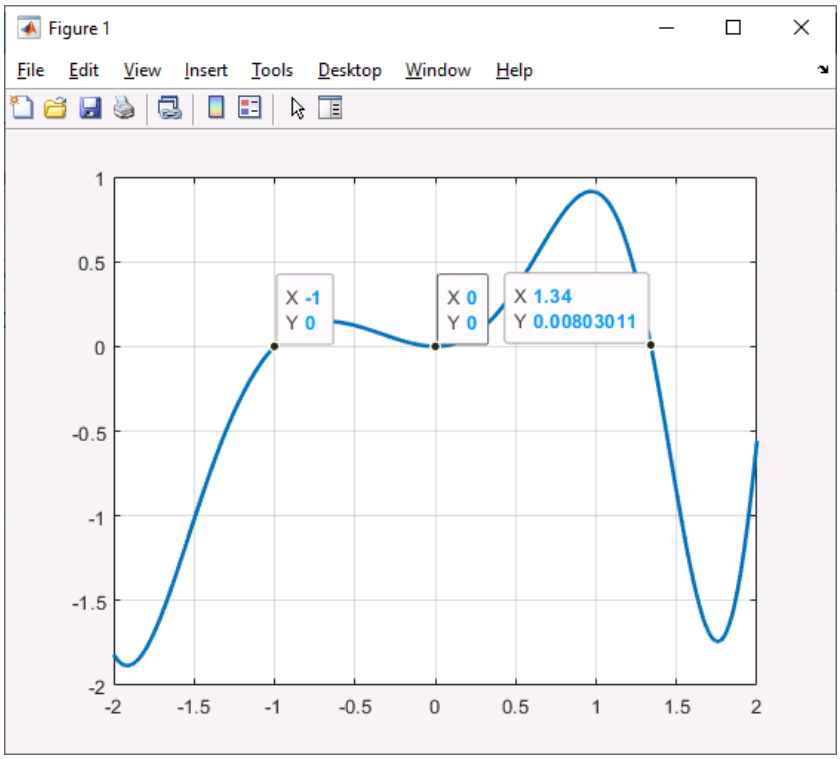
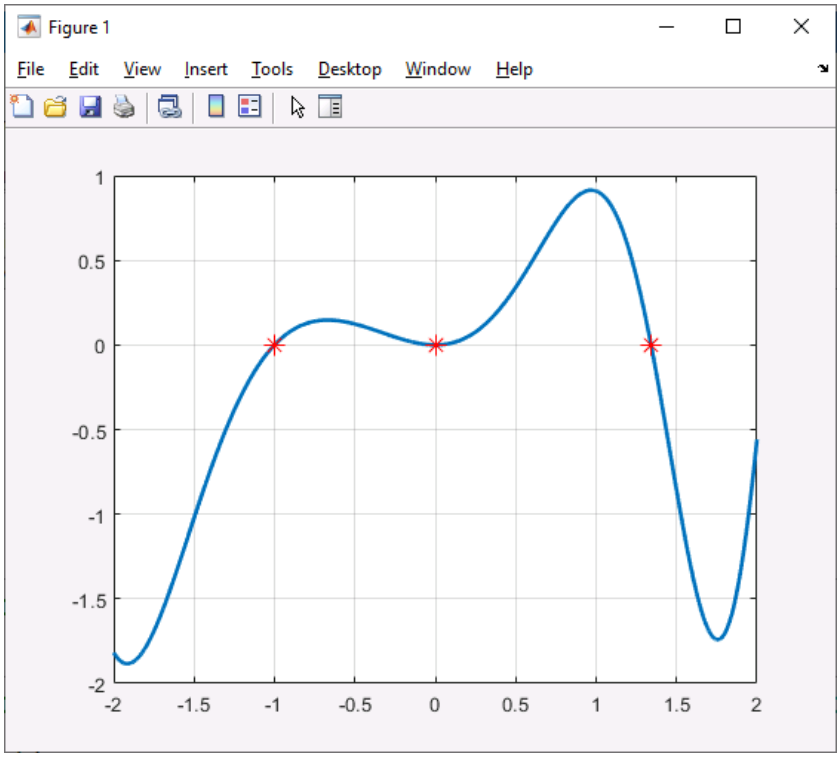
```
%% Лекция 8
%% Решение нелинейного уравнения в Matlab
clc,clear

x=[-3:0.01:3];
f=x.*sin(x+x.^2);

% 1. Визуально определяем корни, построив график
% График
figure (1)
p1=plot(x,f)
set(p1,linewidth=2)
grid
hold on
```

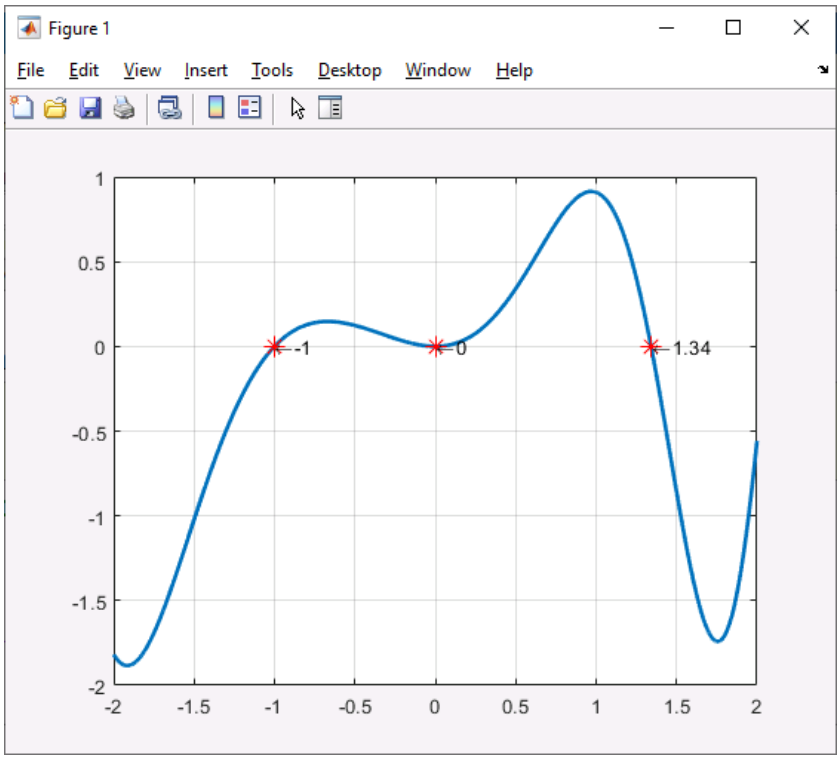



```
%% Записываем корни в вектор
r0=[-1, 0, 1.34 ]

%% Составляем анонимную функцию (в fsolve передается анонимная функция или файл-функция)
y =@(x)x.*sin(x+x.^2)

%% Ищем корни с помощью
%% fsolve(анонимная функция, вектор приближенных корней)
rx=fsolve(y,r0)

%% Формируем набор точек, соответствующих корням
ry=repmat([0],1,numel(rx));
```

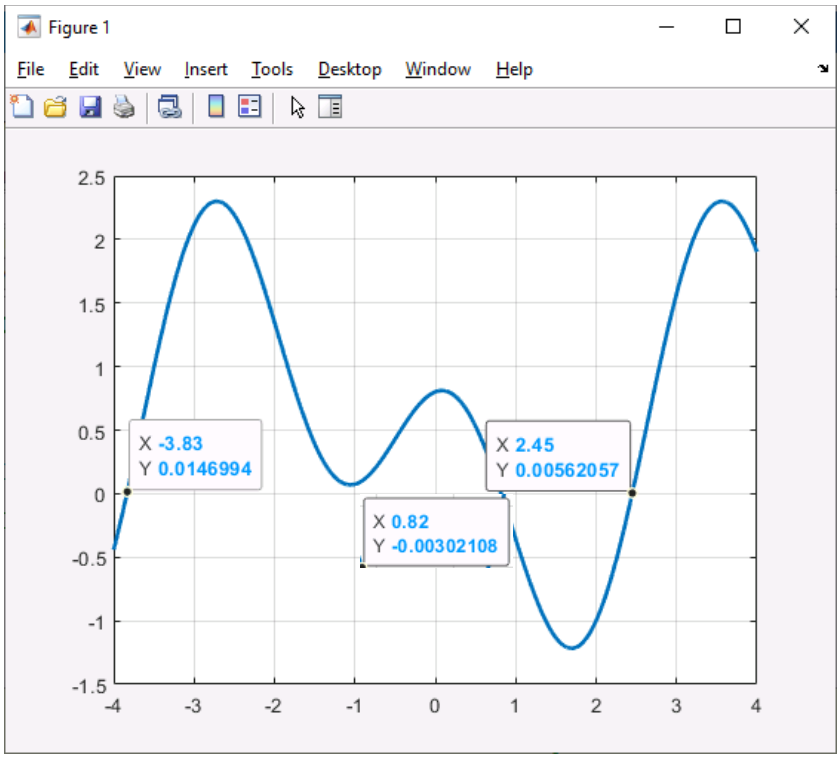
```
%% Обозначим корни на графике символами '*' красного цвета
p2=plot(rx,ry,'r*','MarkerSize',10)
```




```
%% Подпишем корни на графике
for i=rx
    rxt=num2str(i,3)
    text(i,0,['\leftarrow' rxt])
end
```

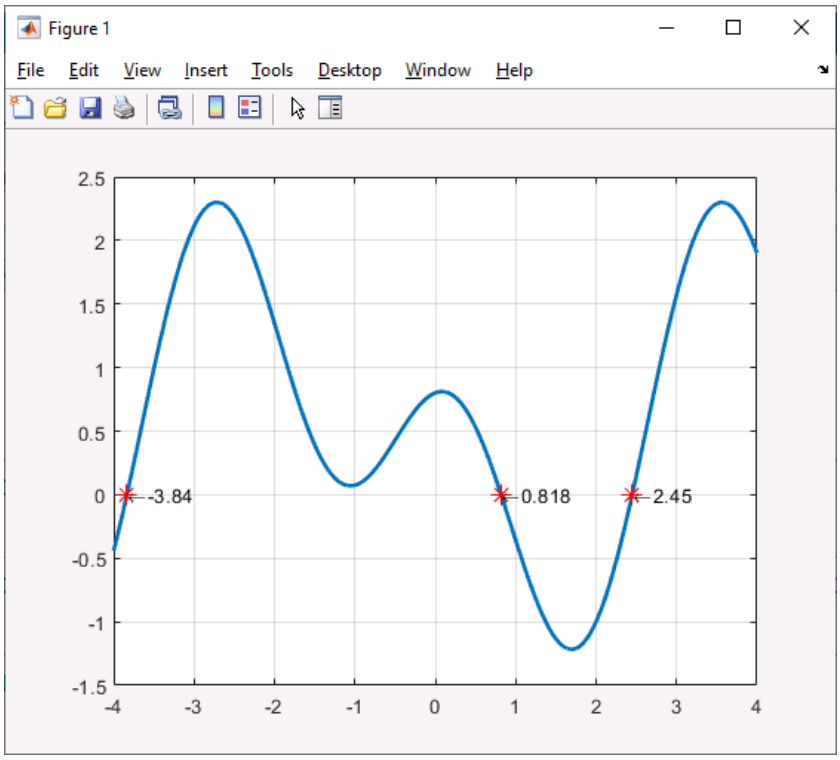


```
%% Решить систему нелинейных уравнений
% y=0.5-cos(x-1)
% y=-sin(2*x+1)

% Способ первый
% Вычтем одно уравнение из другого и воспользуемся
% схемой для решения одного нелинейного уравнения

clc, clear
x=-4:0.01:4
f=0.5-cos(x-1)+sin(2*x+1)

%%
% 1. Визуально определяем корни, построив график
% График
figure (1)
p1=plot(x,f)
set(p1,linewidth=2)
grid
hold on
```

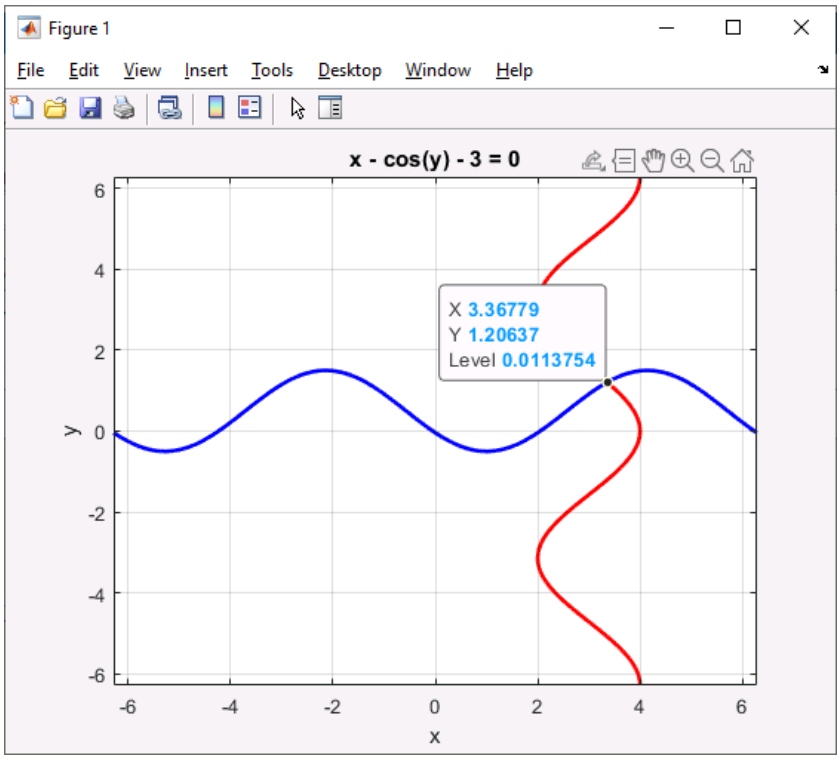



```
%% Записываем корни в вектор
r0=[-3.83, 0.82, 2.45 ]

%% Составляем анонимную функцию (в fsolve передается анонимная
функция или файл-функция)
y =@(x)0.5-cos(x-1)+sin(2*x+1)

%% Ищем корни с помощью
% fsolve(анонимная функция, вектор приближенных корней)
rx=fsolve(y,r0)
```

```
%% Формируем набор точек, соответствующих корням
ry=repmat([0],1,numel(rx));


%% Обозначим корни на графике символами '*' красного цвета
p2=plot(rx,ry,'r*','MarkerSize',10)


%% Подпишем корни на графике
for i=rx
    rxt=num2str(i,3)
    text(i,0,['\leftarrow' rxt])
end
```

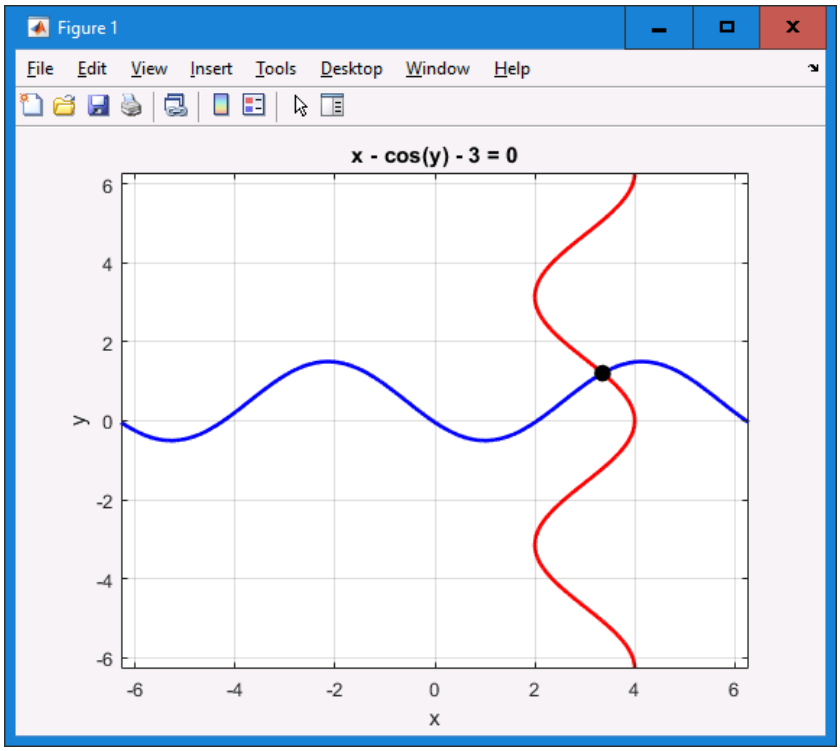


```
%% Решить систему нелинейных уравнений
% f1= cos(x-1) + y - 0.5
% f2= x - cos(y) - 3

%% Второй способ

%% Строим графики
clc, clear

y1 = ezplot('cos(x-1) + y - 0.5');
set(y1,'Color','b','LineWidth',2);

hold on;

y2 = ezplot('x - cos(y) - 3');
set(y2,'Color','b','LineWidth',2);

grid on;
%% Точки пересечения графиков - это решение системы
r0=[3.36 , 1.2]
```




```
% Создадим  файл-функцию (записать в  файл-функцию или в самый
% конец файла-скрипта)

% function f = ff(x) % в самый конец файла-скрипта!!!
%     f(1)= cos(x(1)-1) + x(2) - 0.5;
%     f(2)= x(1) - cos(x(2)) - 3;
% end
%
```

```
% Нахождение корней
[xr, fr, ex] = fsolve(@ff,[3.36 , 1.2])

% Точка пересечения линий на графике
plot(xr(1), xr(2), 'ko','MarkerSize',8,'MarkerFaceColor','k');
```




```
%% xr - это вектор решений
%% fr - это значения функций от xr, они должны быть близки к 0
%% ex - это параметр сходимости, если он равен 1, то все сошлось

%% Поиск корней нелинейных уравнений. Функции fzero, roots
%% Корень нелинейной функции fzero
clc, clear
fun = @sin; % function
x0 = 3; % initial point
x = fzero(fun,x0)


%% Поиск на заданном интервале
fun = @cos; % function
x0 = [1 2]; % initial interval
x = fzero(fun,x0)

%% Поиск корней функции, заданной в файле-функции
%% можно задать в этом же файле - в самом низу файла!!!
%%
fun = @f3; % function
x0 = 2; % initial point
z = fzero(fun,x0) % только вещественные корни (мнимые корни не
найдутся)

%%
```

```
% function y = f3(x) % - в самом низу файла!!!
% y = x.^3 - 2*x - 5;
% end

%% Поиск вещественных и мнимых корней полинома - roots
clc
roots([1, 0, -2, -5]) % аргументы - коэффициенты полинома

%% Поиск корней функции с параметром
myfun = @(x,c) cos(c*x);  % parameterized function
c = 2;                    % parameter
fun = @(x) myfun(x,c);    % function of x alone
x = fzero(fun,0.1)

%% Пример
clear,clc
x=[-2:0.01:2]

% Коэффициенты полинома
p=[1 0 0 1.4 -9 -8];

%% Корни полинома
R=roots(p)

%% Полином по коэффициентам
f=poly(p)

%% Значения полинома в точках x
fy=polyval(p,x)

%% Ищем вещественные корни, отмечаем их на графике:
% imag - мнимая часть комплексного числа
index_R=find(imag(R)==0)
R(index_R)
points=R(index_R)';
plot(x,fy,points,zeros(size(points)),'ro',
'Markersize',6,'LineWidth',2)

%% Интерактивное задание ginput нулевого приближения:
clc, clear
figure

% 'sin(x.*sin(10*x))' % функция с осцилляцией
myf='sin(x.*sin(10*x))';
fplot(myf,[-0.9,0.9],'LineWidth',2), hold on, grid

%% [x,y] = ginput(n) позволяет определить координаты n точек в
интерактивном режиме
%%  в пределах декартовой (полярной/географической) системы
координат

% X, Y - вектор координат
[X,Y]=ginput(5) % отмечаем 5 точек для нулевого приближения,
```

```
% График
plot(X,Y,'ro','MarkerSize',20) % отмечаем нулевые приближения

% Поиск корней fsolve
R=fsolve(myf,X)

% Корни на графике
plot(R,0,'m.','MarkerSize',20)% нашли корни по заданным
приближениям
legend(myf, 'initial points','roots')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
function f = ff(x) % в самый конец файла-скрипта!!!
    f(1)= cos(x(1)-1) + x(2) - 0.5;
    f(2)= x(1) - cos(x(2)) - 3;
end
%%
function y = f3(x) % - в самом низу файла!!!
y = x.^3 - 2*x - 5;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Интегрирование в Matlab

Математическая операция	MATLAB® Команда
$\int x^n dx = \begin{cases} \log(x) & \text{если } n = -1 \\ \frac{x^{n+1}}{n+1} & \text{в противном случае.} \end{cases}$	`int(x^n)` или `int(x^n,x)`
$\int_0^{\pi/2} \sin(2x)dx = 1$	`int(sin(2*x), 0, pi/2)` или `int(sin(2*x), x, 0, pi/2)`
$g = \cos(at + b)$ $\int g(t)dt = \sin(at + b)/a$	`g = cos(a*t + b) int(g)` или `int(g, t)`
$\int J_1(z)dz = -J_0(z)$	`int(besselj(1, z))` или `int(besselj(1, z), z)`

Определенный интеграл	Команда
$\int_a^b f(x)dx$	`int(f, a, b)`
$\int_a^b f(v)dv$	`int(f, v, a, b)`

```
%% Символьное интегрирование в Matlab
clear,clc;
syms x n
f = x^n
int(f)

%% Пример. Символьное интегрирование в Matlab
syms y
f = y^(-1);
int(f)

%% Пример. Символьное интегрирование в Matlab
syms x n
f = n^x;
int(f)

%% Пример. Символьное интегрирование в Matlab
syms a b theta
f = sin(a*theta+b);
int(f)

%% Пример. Символьное интегрирование в Matlab
syms u
f = 1/(1+u^2);
int(f)

%% Пример. Символьное интегрирование в Matlab
syms x
f = exp(-x^2)
int(f)

%% Пример. Символьное интегрирование в Matlab
clear,clc
syms x %Определение переменной
f=str2sym('a^x*e^(-x)'); %Определение функции
int(f,x) %Вычисление неопределенного интеграла
```



```
%% Определенный интеграл
clear,clc
syms x
expr = x*log(1+x);
F = int(expr,[0 1])

%% Определенный интеграл
syms u
f = 1/(1+u^2);
eval(int(f,u,1,2))

%% integral. Численное интегрирование (двойная точность)
clc, clear
%fun = @(x)log(x);
fun = @(x) exp(-x.^2).*log(x).^2;
q = integral(fun,0,1) % integral from x=0 to x=Inf.
```

```
%% integral. Численное интегрирование
fun = @(x,c) 1./(x.^3-2*x-c); %  Parameterized Function
q = integral(@(x) fun(x,5),0,2) % Evaluate the integral from x=0
to x=2 at c=5.

%% Singularity at Lower Limit
fun = @(x)log(x); %
format long
q1 = integral(fun,0,1) % Singularity at Lower Limit

%% Integral of Oscillatory Function
format long
fun = @(x)x.^5.*exp(-x).*sin(x); % Integral of Oscillatory
Function
q = integral(fun,0,Inf,'RelTol',1e-8,'AbsTol',1e-13)

%% Vector-Valued Function
fun = @(x)sin((1:5)*x);
q = integral(fun,0,1,'ArrayValued',true)

%% Численное интегрирование высокой точности vpaintegral
syms y(x)
y(x) = x^2;
vpaintegral(y, 1, 2)

%% Численное интегрирование высокой точности vpaintegral
% RelTol — Допуск относительной погрешности
% 1e-6 (значение по умолчанию) | положительное вещественное
число
% AbsTol — Допуск абсолютной погрешности
% 1e-10 (значение по умолчанию) | неотрицательное вещественное
число
syms x
vpaintegral(besselj(0,x), [0 pi], 'RelTol', 1e-32, 'AbsTol', 0)

%% Численное интегрирование в Matlab. Кратные интегралы
syms x y
vpaintegral(vpaintegral(x*y, x, [1 3]), y, [-1 2])

%% https://codetown.ru/matlab/integrirovanie/
%%
% Геометрический смысл интегрирования — это нахождение площади,
% которая находится под интегрируемой функцией.

% Методы прямоугольников
%   метод правых прямоугольников
%   метод левых прямоугольников
%   метод средних прямоугольников

%% Пример 1
% посчитать интеграл функции f(x) = x.*exp((sin(x)).^x)
% с шагом разбиения h = 0.02 на интервале от 0 до 1.
% f=inline('x.*exp((sin(x)).^x)');
```

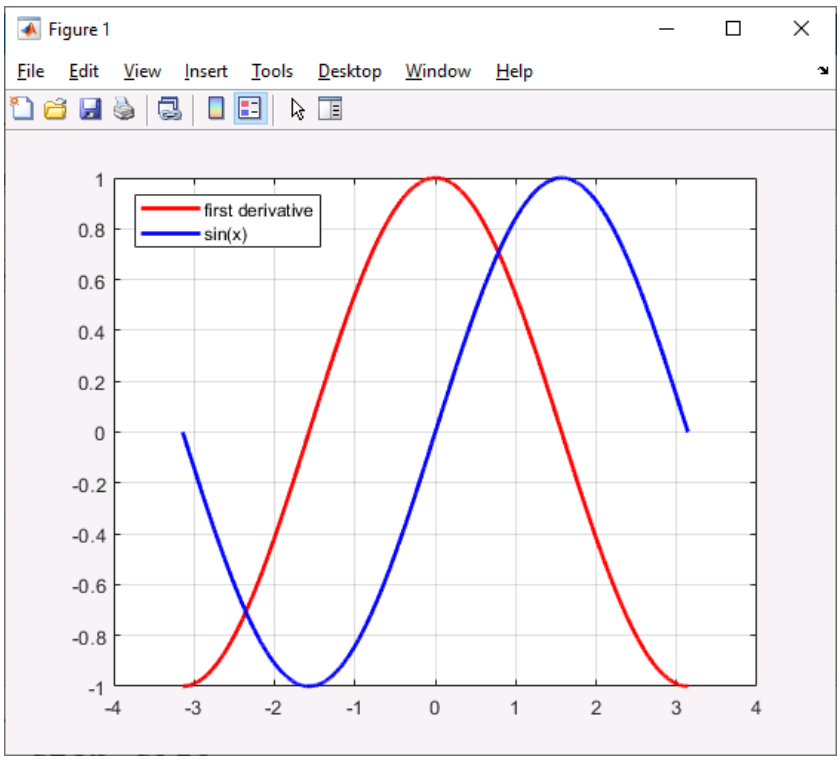
```
clear, clc
f=@(x)x.*exp((sin(x)).^x);
a=0;
b=1;
h=0.02;
N=((b-a)/h)+1;
i=1:N; %количество шагов
x=a:h:b; %вычисление координат узлов сетки
% feval вычисляет текстовую строку, которая может содержать
% либо арифметическое выражение, либо инструкцию, либо обращение
к функции
y=feval(f,x); %вычисление значений функции в узлах сетки
m=2:N;
y1(m-1)=y(m);
Fr=sum(h*y1)

%% Метод трапеций
% Ещё одни популярный и в тоже время простой метод — метод
трапеций.
% Аналогично методу прямоугольников строятся трапеции под кривой
% и находится их суммарная площадь.
% Данный метод имеет второй порядок точности
% (ошибка пропорциональна шагу в квадрате).
%
% В Matlab метод трапеций реализован двумя функциями:
%   cumtrapz()
%   trapz()
%
%% cumtrapz() применяется при работе с табличными данными или
векторами.
%% Откликом функции является n-интегралов, где n — число
элементов
%% вектора или элементов в каждом столбце матрицы.

%% Пример 2
% Пусть функция y(x) имеет значения, представленные в виде
следующего вектора:
% y = [1,2,3,4,5,6,7,8,9,10]. Необходимо вычислить интеграл от y
clear, clc
y=[1,2,3,4,5];
x=[1,2,3,4,5];
cumtrapz(x,y)
% Ответ - интеграл на каждом шаге вычисляется.
% Последнее значение - int(x,x=1..5)=12

%% Пример 3
% Функция y(x) задана в виде матрицы
% y(x) = [1 3 5; 3 5 7; 4 6 8; 4 7 9; 5 7 10].
% При этом аргумент представляет собой вектор: x = [1,3,7,9,10].
y = [1 3 5; 3 5 7; 4 6 8; 4 7 9; 5 7 10];
x = [1,3,7,9,10];
cumtrapz(x,y)
```

```
%% trapz() - позволяет работать не только с векторами и
матрицами,
%% но и с аналитической формой подынтегральной функции.

%% Пример 4
x = 1:5;
y = x;
trapz(y)

%% Пример 5
x = 1:0.8:10;
y = x.*exp(-x) + log(x) +1;
trapz(y)

%% Метод Симпсона
% Преимущество этого метода в том, что точки, взятые на каждом
шаге на кривой,
% интерполируются полиномом второй степени.
% Проще говоря, соединяются параболой.
% Это даёт методу четвёртый порядок точности.

%% Пример 6
quad('x.*exp(-x)+log(x)+1',1,10,1e-5) %зададим погрешность 10*-5
%% алгоритм Симпсона
clc,clear
F = @(x) x*exp(-x)+log(x)+1; %функция
a=1; %пределы интегрирования
b=10;
n=1000; %количество частей деления
h=(b-a)/n; %определяем шаг
integ = F(a);
for i=1:1:((n/2)-1) %сам алгоритм Симпсона
x=a+2*h*i;
integ=integ+2*F(x)+4*F(x+h);
end
integ=h*integ/3
```

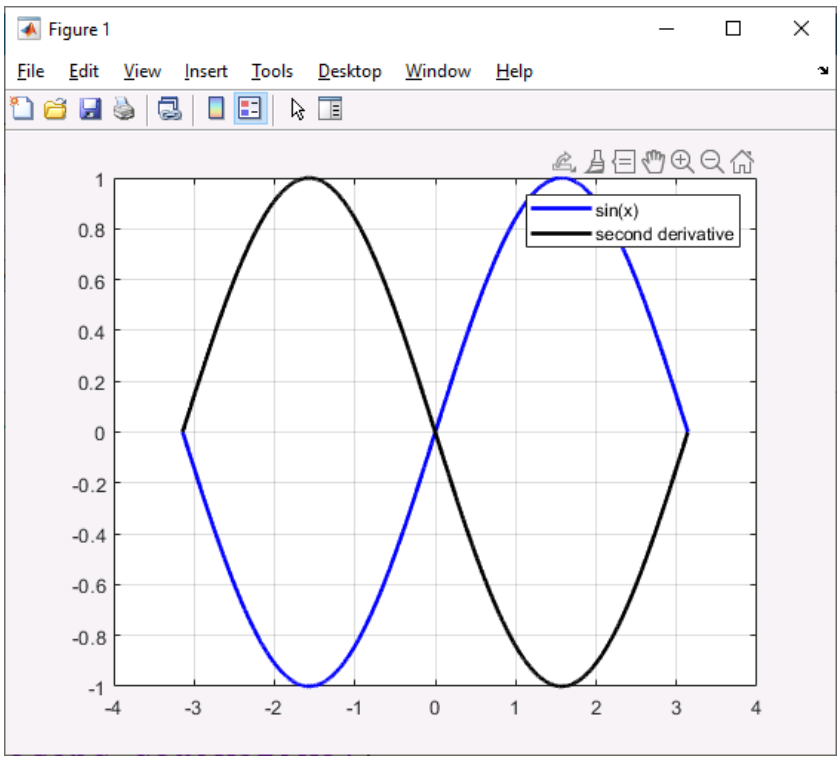

%% Символьное дифференцирование в Matlab

```
syms f(x)
f(x) = sin(x^2);
Df = diff(f,x)

%%
Df =eval(Df(2))

%% Дифференцирование относительно конкретной переменной
syms x t
Df = diff(sin(x*t^2)) % по умолчанию по x

%%
Df = diff(sin(x*t^2),t) % по t
```

```
%%
Df = diff(sin(x*t^2),t,t) % по t дважды

%% Производные высшего порядка
Df = diff(x^5,5)

%% Производные высшего порядка
Df = diff(sin(t*x^5),t,5) % по t 5 порядка
%%
% В первом вызове, diff дифференцирует x*y относительно x, и
возвращает y.
% Во втором вызове, diff дифференцирует y относительно y, и
возвращает 1.
Df = diff(diff(x*y))

%% Смешанные производные
syms x y
Df = diff(x*sin(x*y),x,y)

%% Численное дифференцирование в Matlab diff
%% Y = diff(X) вычисляет разности между смежными элементами X
%% вдоль первого измерения массива, размер которого не равняется
1
X = [1 1 2 3 5 8 13 21];
Y = diff(X) % Обратите внимание на то, что Y имеет меньше
элементов, чем X.
numel(X)-numel(Y)

%% Различия между матричными строками
X = [1 1 1; 5 5 5; 25 25 25];
Y = diff(X)

%% Различия между столбцами матрицы
X = [1 3 5;7 11 13;17 19 23]
Y = diff(X,1,2)

%% Аппроксимативные производные с diff
% Используйте diff, чтобы аппроксимировать частные производные
% синтаксисом Y = diff(f)/h, где f - вектор из значений функции,
% и h соответствующий размер шага.
clc,clear;
h = 0.001;        % step size
X = -pi:h:pi;     % domain
f = sin(X);       % range
Y = diff(f)/h;    % first derivative cos
plot(X(:,1:length(Y)),Y,'r',X,f,'b','LineWidth',2)
legend('first derivative','sin(x)')
grid on
```




```
%% second derivative
h = 0.001;          % step size

X = -pi:h:pi;       % domain
f = sin(X);         % range

Y = diff(f)/h;      % first derivative
Z = diff(Y)/h;      % second derivative = -sin(x)

plot(X,f,'b', X(:,1:length(Z)),Z,'k','LineWidth',2)

legend('sin(x)','second derivative')
grid on
```




```
%% https://codetown.ru/matlab/chislennoe-differencirovanie/
%% Численное дифференцирование строится на использовании
%% аппарата конечных разностей и соответствующего многообразия
аппроксимаций.
% Методы численного дифференцирования применяются,
% если исходную функцию y(x) трудно или невозможно
% продифференцировать аналитически.
%
% Методы имеют разную погрешность при расчётах.
%
% Рассмотрим основные из них и оценим погрешность помощью
Matlab:
% Сравним результаты, дифференцируя функцию sin(x).
%
%% Подготовительная работа
clc,clear;
h=0.2; % определим шаг сетки
x=0:h:pi; % интервал значений x
n=length(x); % число необходимых итераций
dy=cos(x); % производная y = sin(x) => y' = cos(x), так будем
судить об отклонении и погрешности.
dz=sin(x); % для сравнения

%% Метод нахождения производной правой конечной разностью
for i=1:(n-1)
 dy1(i)=(sin(x(i+1))-sin(x(i)))/h;
 er1(i)=abs(dy(i)-dy1(i));
end

%% Метод нахождения производной левой конечной разностью
for i=2:n
```

```
 dy2(i)=(sin(x(i))-sin(x(i-1)))/h;
 er2(i)=abs(dy(i)-dy2(i));
end

%% Метод нахождения производной центральной конечной разностью
for i=2:(n-1)
 dy3(i)=(sin(x(i+1))-sin(x(i-1)))/(2*h);
 er3(i)=abs(dy(i)-dy3(i));
end
%% Метод нахождения производной четвертого порядка точности
for i=3:(n-1)
 dy4(i)=(-sin(x(i+1))+27*sin(x(i))-...
 27*sin(x(i-1))+sin(x(i-2)))/(24*h);
 er4(i)=abs(cos(x(i)-0.5*h)-dy4(i)); % абсолютную погрешность,
которая вычисляется в точке x(i)-0.5*h
end
%%
%рисуем все на графиках
 plot(x([1:(n-1)]),er1([1:(n-1)]),'-o',...
             x([2:n]),er2([2:n]),'-p',...
     x([2:(n-1)]),er3([2:(n-1)]),'-h',...
     x([3:(n-1)]),er4([3:(n-1)]),'-*');
 title('Погрешность ("разность" анлитического и численного
решения)');
 legend('Правая', ' Левая', 'Центральная', '4-ый порядок')
 grid on;
figure; plot(x([1:(n-1)]),dy([1:(n-1)]),'-d',...
                x([1:(n-1)]),dz([1:(n-1)]),'-<',...
                x([1:(n-1)]),dy1([1:(n-1)]),'-o',...
     x([3:(n-1)]),dy4([3:(n-1)]) - 0.5*h,'-*');
 legend('cos(x)', ' sin(x)', 'производная по правой конечной
разности', 'производная по 4-ому порядоку')
 grid on
%%
syms t;
y=2*cos(t.^2);
z=diff(y,2);
str1=char(z)
t=1;
Result=eval(str1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
%%
function f = ff(x) % в самый конец файла-скрипта!!!
    f(1)= cos(x(1)-1) + x(2) - 0.5;
    f(2)= x(1) - cos(x(2)) - 3;
end
%%
function y = f3(x) % - в самом низу файла!!!
y = x.^3 - 2*x - 5;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```