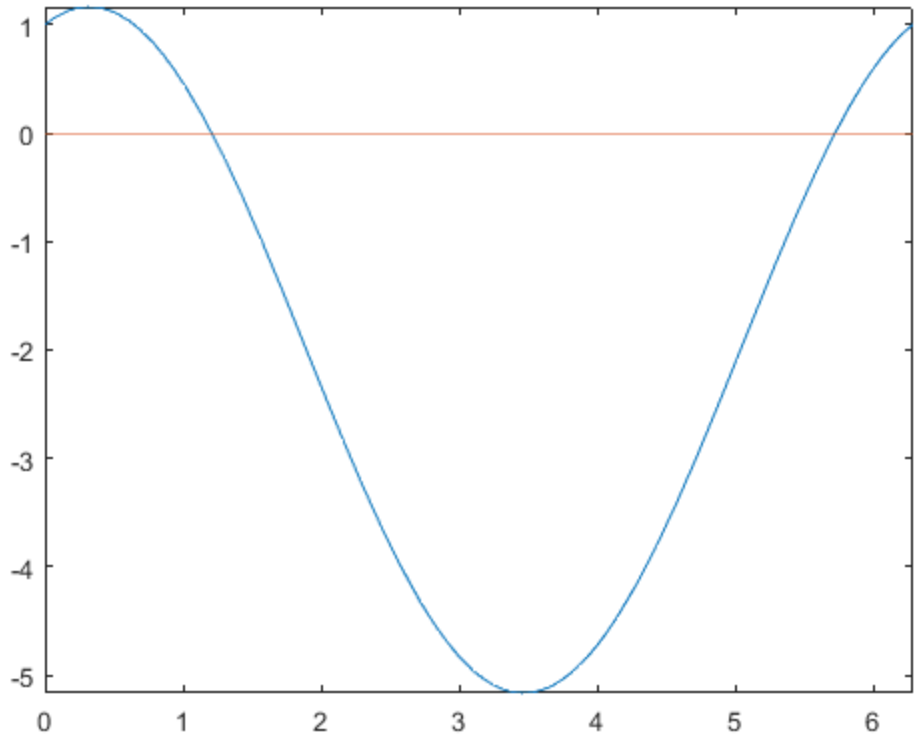
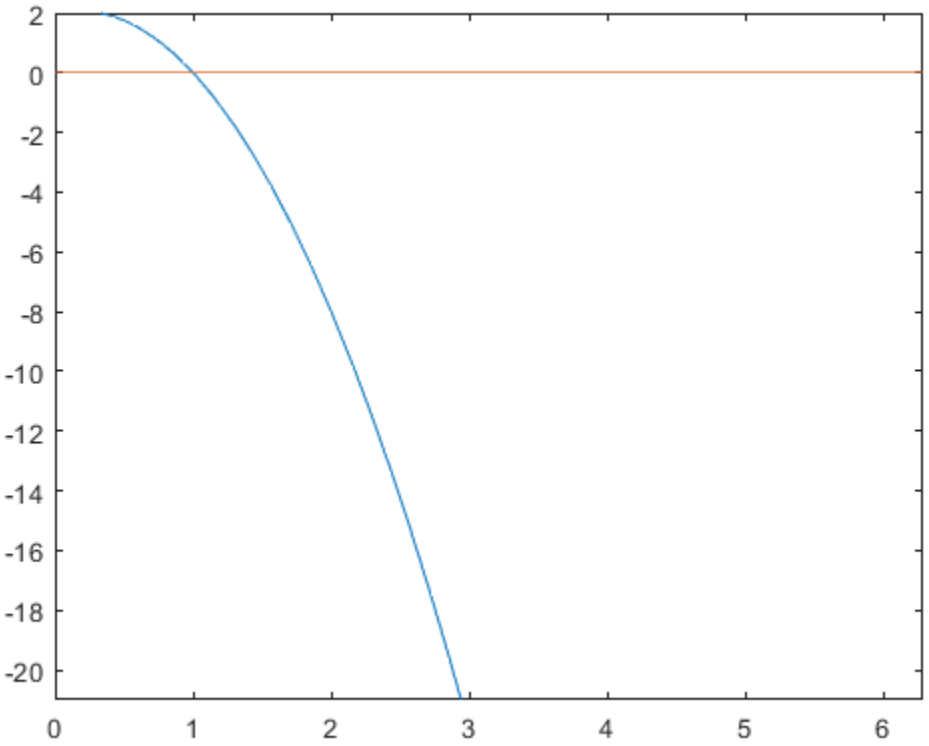
```
syms x
a=sin(x)+3*cos(x)-2;
f1sol=solve(a,x);
zero = 0;
b=x-3*x^2+2;
f2sol=solve(b,x);

root_a = double(f1sol)
root_b = double(f2sol)


figure(1)
fplot(a);
hold on;
fplot(zero);
hold on
xlim([0 2*pi]);

figure(2)
fplot(b);
hold on;
fplot(zero);
hold on
xlim([0 2*pi]);
ylim([-21,2]);
```

```
root_a =

    1.2078
   -0.5643


root_b =

   -0.6667
    1.0000
```


1
0
-1
-2
-3
-4
-5
0
1
2
3
4
5
6


2
0
-2
-4
-6
-8
-10
-12
-14
-16
-18
-20
0
1
2
3
4
5
6

```
a = @(x)(sin(x)+3*cos(x)-2);
b = @(x)(x-3*x^2+2);
root = bisect(a,0,pi)
root = bisect(b,0,pi)

a = @(x)(sin(x)+3*cos(x)-2);
da = @(x)(cos(x)-3*sin(x));
b = @(x)(x-3*x^2+2);
db = @(x)(1-6*x);
[root,numiter] = newton_simple(a,da,2)
[root,numiter] = newton_simple(b,db,2)

a = @(x)(sin(x)+3*cos(x)-2);
b = @(x)(x-3*x^2+2);
[root,numiter] = secant(a,2.01,2.00)
[root,numiter] = secant(b,2.01,2.00)

x0 = [1;1];
[root,iter] = newton(x0)

%function and jacobb defination for d part
function f = func(X)
    f1 = sin(X(1))+3*cos(X(2))-2;
    f2 = cos(X(1))-sin(X(2))+0.2;
    f = [f1;f2];return;
end

function J = Jacob(X)
    J(1,1) = cos(X(1));
    J(1,2) = -3*sin(X(2));
    J(2,1) = -sin(X(1));
    J(2,2) = -cos(X(2));
    return;
end

%function for bisection method
function root = bisect(func,x1,x2,tol)
    if nargin<4
        tol = 1e-12;
    end
    f1 = feval(func,x1);
    if f1 == 0.0
        root = x1;
        return;
    end
    f2 = feval(func,x2);
    if f2 == 0.0
        root = x2;
        return;
    end
    n = ceil(log(abs(x2-x1)/tol)/log(2));
    for i=1:n
        x3 = 0.5*(x1+x2);
        f3 = feval(func,x3);
        if f3 == 0.0
            root = x3;
            return;
        end
        if f3*f2 < 0.0
```

```
            x1 = x3; f1 = f3;
        else
            x2 = x3; f2 = f3;
        end
    end
    root = 0.5*(x1+x2); return;
end

%function for newton method
function [root,numiter] = newton_simple(func,dfunc,x,tol,N)
    if nargin < 4
        tol = 1e-6; N = 100;
    end
    if nargin < 5
        N = 100;
    end
    for i = 1:N
        dx = -feval(func,x)/feval(dfunc,x);
        x = x+dx;
        if abs(dx)<tol
            root = x; numiter = i; return;
        end
    end
    root = Nan; error('Select New value');
end

%function for secant method
function [root,numiter] = secant(func,x1,x2,tol,N)
    if nargin < 4
        tol = 1e-6; N = 100;
    end
    if nargin < 5
        N = 100;
    end
    for i = 1:N
        Q = (feval(func,x2)-feval(func,x1))/(x2-x1);
        dx = -feval(func,x2)/Q;
        x3 = x2+dx;
        if abs(dx)<tol
            root = x3; numiter = i; return;
        end
        x1 = x2; x2 = x3;
    end
    root = Nan; error('Select new value');
end

%function for newton general method
function [root,iter] = newton(x0)
    f = @func;
    J = @Jacob;
    N = 100; eps = 1e-10; maxVal = 10000.0;
    X = x0;
    while(N > 0)
        JJ = feval(J,X);
        if abs(det(JJ))<eps
            error('Jacobian is singular, try new initial condition');
        end
        xn = X-inv(JJ)*feval(f,X);
        if abs(feval(f,xn)) < eps
            root = xn; iter = 100-N; return;
        end
```

```
        if abs(feval(f,xn)) > maxVal
            iter = 100-N; error('Solution Diverges');
            disp(['Iteration = ',num2str(iter)]);
        end
        N = N-1; X = xn;
    end
    error('No convergence');
end
```

```
root =

    1.2078


root =

    1.0000


root =

    1.2078


numiter =

     4


root =

    1.0000


numiter =

     5


root =

    1.2078


numiter =

     5


root =

    1.0000


numiter =

     7
```

```
% Script file

[xsol,ysol]=taylor(@f,0,1,3,0.2);
Sol = [xsol,ysol];
display(Sol);

%Function file

function [xsol,ysol]=taylor(derivat,x,y,xstop,h)
if size(y,1)>1
    y=y';
end
n=floor(xstop/h);
xsol=zeros(n,1);
ysol=zeros(n,length(y));
xsol(1,1)=x;
ysol(1,length(y))=y;
k=1;
while x<xstop
    h=min(h,xstop-x);
    d=feval(derivat,x,y);
    hh=1;
    for j=1:4
        hh=hh*h/j;
        y=y+d(j,:)*hh;
    end
    x=x+h;
    k=k+1;
    xsol(k,1)=x;
    ysol(k,length(y))=y;
end
end

%derivation file
function derivation = f(x,y)
derivation=zeros(4,1);
derivation(1,1)=x^2-4*y;
derivation(2,1)=2*x-4*x^2+16*y;
derivation(3,1)=2-8*x+16*x^2-64*y;
derivation(4,1)=-8+32*x-64*x^2+256*y;
end
```

```
Sol =

         0    1.0000
    0.2000    0.4539
    0.4000    0.2189
    0.6000    0.1356
    0.8000    0.1316
    1.0000    0.1745
    1.2000    0.2495
    1.4000    0.3500
    1.6000    0.4729
    1.8000    0.6170
    2.0000    0.7816
    2.2000    0.9664
    2.4000    1.1713
```

```
    2.6000    1.3963
    2.8000    1.6413
    3.0000    1.9063
```

Published with MATLAB® R2019b

```
% Finite difference method

x_start=0;
x_stop=pi/2;
n=20
h=(x_stop-x_start)/(n-1);
x=linspace(x_start,x_stop,n); % x values
[A,b]=finitedifference(x,h,n); % calling function
Y=inv(A)*b;   % A*Y = b
Sol=[transpose(x),Y];
display(Sol);

%Function file
function [A,b]=finitedifference(x,h,n)

A=zeros(n);
A(1,1)=1;
A(n,n-1)=2;

for i=2:n
    A(i,i)=4*h^2-2;
end

for i=2:n-1
    A(i,i-1)=1;
    A(i,i+1)=1;
end

b=4*h^2*transpose(x);

end
```

```
n =

    20


Sol =

         0         0
    0.0827    0.1653
    0.1653    0.3284
    0.2480    0.4871
    0.3307    0.6392
    0.4134    0.7828
    0.4960    0.9164
    0.5787    1.0384
    0.6614    1.1479
    0.7441    1.2441
    0.8267    1.3267
    0.9094    1.3955
    0.9921    1.4511
    1.0748    1.4941
    1.1574    1.5257
    1.2401    1.5471
    1.3228    1.5602
    1.4054    1.5668
```

```
function shoot

xstart = 0;
xstop = 1;
h = 0.05;
u1 = -5;
u2 = 5;
x = xstart;

u = bisect(@residual,u1,u2);
display(u);
[xsol,ysol] = rungeKut4(@dE,x,inCond(u),xstop,h);
Sol = [xsol,ysol];
display(Sol);


function F = dE(x,y)
    F = [y(2),9*y(1)+18*x-9];
end

function y = inCond(u)
    y = [1 u];
end

function [xsol,ysol] = rungeKut4(dE,x,y,xStop,h)
    if size(y,1) > 1
        y = y.' ;
    end

    xsol = zeros(2,1);
    ysol = zeros(2,length(y));
    xsol(1) = x;
    ysol(1,:) = y;
    i = 1;
    while x < xStop
        i = i + 1;
        h = min(h,xStop - x);
        K1 = h*feval(dE,x,y);
        K2 = h*feval(dE,x + h/2,y + K1/2);
        K3 = h*feval(dE,x + h/2,y + K2/2);
        K4 = h*feval(dE,x+h,y + K3);
        y = y + (K1 + 2*K2 + 2*K3 + K4)/6;
        x = x + h;
        xsol(i) = x; ysol(i,:) = y;
    end

end

function r = residual(u)
    x = xstart;
    [xsol,ysol] = rungeKut4(@dE,x,inCond(u),xstop,h);
    r = ysol(size(ysol,1),1) + 1;
end

function root = bisect(func,x1,x2,tol)
    if nargin<4
        tol = 1e-12;
    end
    f1 = feval(func,x1);
```

```
    if f1 == 0.0
        root = x1;
        return;
    end
    f2 = feval(func,x2);
    if f2 == 0.0
        root = x2;
        return;
    end
    n = ceil(log(abs(x2-x1)/tol)/log(2));
    for i=1:n
        x3 = 0.5*(x1+x2);
        f3 = feval(func,x3);
        if f3 == 0.0
            root = x3;
            return;
        end
        if f3*f2 < 0.0
            x1 = x3; f1 = f3;
        else
            x2 = x3; f2 = f3;
        end
    end
    root = 0.5*(x1+x2); return;
end

end
```

```
u =

   -2.0000


Sol =

         0    1.0000   -2.0000
    0.0500    0.9000   -2.0000
    0.1000    0.8000   -2.0000
    0.1500    0.7000   -2.0000
    0.2000    0.6000   -2.0000
    0.2500    0.5000   -2.0000
    0.3000    0.4000   -2.0000
    0.3500    0.3000   -2.0000
    0.4000    0.2000   -2.0000
    0.4500    0.1000   -2.0000
    0.5000   -0.0000   -2.0000
    0.5500   -0.1000   -2.0000
    0.6000   -0.2000   -2.0000
    0.6500   -0.3000   -2.0000
    0.7000   -0.4000   -2.0000
    0.7500   -0.5000   -2.0000
    0.8000   -0.6000   -2.0000
    0.8500   -0.7000   -2.0000
    0.9000   -0.8000   -2.0000
    0.9500   -0.9000   -2.0000
    1.0000   -1.0000   -2.0000
```