Table of Contents

Demonstrates the dot product and cross product of vectors.

```
clc % ##command window
clear all % ##WorkSpace##
close all %  ######

syms a1 a2 a3 b1 b2 b3 real
a = [a1, a2, a3];
b = [b1, b2, b3];
d = dot(a,b)
c = cross(a,b)
```

```
d =

a1*b1 + a2*b2 + a3*b3


c =

[ a2*b3 - a3*b2, a3*b1 - a1*b3, a1*b2 - a2*b1]
```

Calculates an angle and the unit normal of a triangle defined by three points in the space.

```
clc % ##command window
clear all % ##WorkSpace##
close all %  ######
p1 = [3, 5, 2];
p2 = [1, 0 ,0];
p3 = [3, -1, 0];
a = p2-p1;
b = p3-p1;
d = dot(a,b)
theta = acosd(d/(norm(a)*norm(b)))
c = cross(a,b)
theta = asind(norm(c)/(norm(a)*norm(b)))
n = c/norm(c)
```

```
d =

    34


theta =

       20.639


c =

    -2    -4    12


theta =

       20.639


n =

     -0.15617     -0.31235      0.93704
```

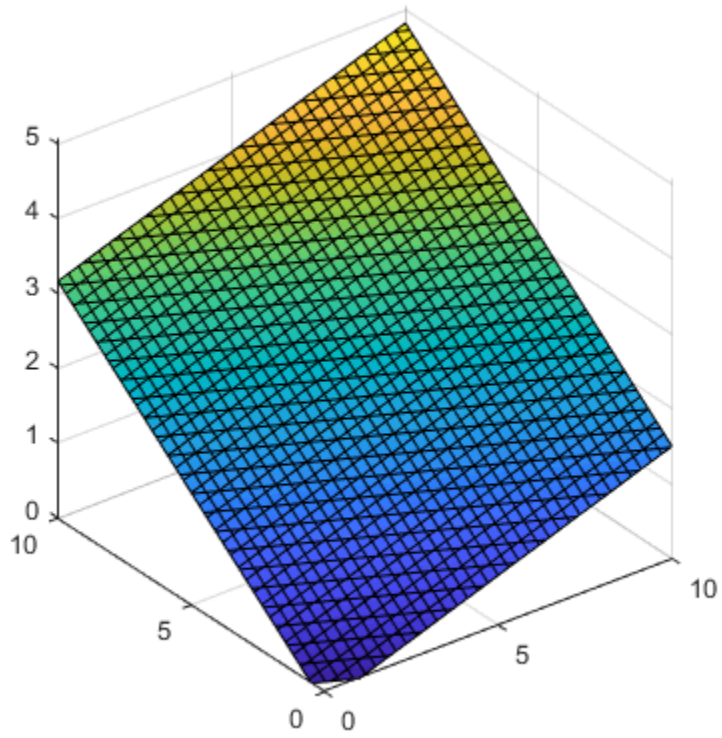
Derives the plane equation defined by three points in the space and plots the plane.

```
clc % ##command window
clear all % ##WorkSpace##
close all %  ######
p1 = sym([3,5,2]);
```

```
p2 = sym([1,0,0]);
p3 = sym([3,-1,0]);
a = p2-p1;
b = p3-p1;
syms x y z
r = [x, y, z];
eq = dot(cross(a,b),r-p1)==0
fimplicit3(eq, [0 10])
axis vis3d
```

```
eq =

12*z - 4*y - 2*x + 2 == 0
```



Demonstrates the conversion of a symbolic expression to a numerical function.

```
clc % ##command window
clear all % ##WorkSpace##
close all %  ######
z = solve(eq, z)
z = matlabFunction(z)
[X,Y] = meshgrid(0:10,0:10);
```

```
Z = z(X,Y);
mesh(X,Y,Z)

Error using ==
Not enough input arguments.

Error in Example1 (line 45)
z = solve(eq, z)
```

Demonstrates the solving of a system of n linear equations of n unknowns.

```
clc % ##command window
clear all % ##WorkSpace##
close all %  ######
A = [1 2; 3 4]; b = [5; 6];
det(A)
x = linsolve(A, b)
x = A\b
A = A'; b = b';
det(A)
x = b/A
A = [1 2; 3 6]; b = [5; 6];
det(A)
x = A\b
b = [5; 15];
x = A\b
```

Solves the system of linear equations governing the 3-bar truss problem in Section 3.14.

```
clc % ##command window
clear all % ##WorkSpace##
close all %  ######
a = sqrt(2)/2;
A = [-a  a  0  0  0  0;
     -a -a  0  0  0  0;
      a  0  1  1  0  0;
      a  0  0  0  1  0;
      0 -a -1  0  0  0;
      0  a  0  0  0  1];
b = [0, 1000, 0, 0, 0, 0]';
x = A\b
x = linsolve(A,b)
det(A)
A = A';
b = b';
x = b/A
```

Solves a system of linear equations by means of the LU factorization.

```
clc % ##command window
clear all % ##WorkSpace##
close all %  ######
a = sqrt(2)/2;
A = [-a  a  0  0  0  0;
     -a -a  0  0  0  0;
      a  0  1  1  0  0;
      a  0  0  0  1  0;
      0 -a -1  0  0  0;
      0  a  0  0  0  1];
b = [0, 1000, 0, 0, 0, 0]';
[L,U] = lu(A)
y = L\b
x = U\y
```

Demonstrates the solving of an eigenvalue problem, Kx = #Mx.

```
clc % ##command window
clear all % ##WorkSpace##
close all %  ######
K = [40, -25,   0;
    -25,  50, -30;
      0, -30,  60]
M = diag([3, 4, 5])
[X, Lamda] = eig(K, M)
Omega = sqrt(Lamda)
```

Demonstrates the manipulations of polynomials

```
clc % ##command window
clear all % ##WorkSpace##
close all %  ######
p = [1, -2, -1, 2];
polyval(p, 3)
x = linspace(-2,3);
plot(x, polyval(p,x)), grid on
r = roots(p)
poly(r)
p1 = polyint(p)
polyder(p1)
a = [1, 3, 5];
b = [2, 4, 6];
c = polyder(a,b)
```

```
[n,d] = polyder(a,b)
```

Does the same tasks as Example10_05a, using symbolic mathematics.

```
clc % ##command window

syms x
p(x) = poly2sym([1, -2, -1, 2])
p(3)
fplot(p), axis([-2,3,-12,8]), grid on
r = solve(p)
p1 = int(p)
diff(p1)
a = poly2sym([1, 3, 5])
b = poly2sym([2, 4, 6])
c = diff(a*b)
[n,d] = numden(diff(a/b))
```

Finds the polynomials of degrees 1, 2, and 3 that best fit a set of 2-D data points.

```
clc % ##command window

T = [ 33, 144, 255, 366, 477, 589, 700, 811, 922];
E = [220, 213, 206, 199, 192, 185, 167, 141, 105];
temp = 0:50:1000;
format shortG
for k = 1:3
    P = polyfit(T, E, k)
    young = polyval(P, temp);
    subplot(2,2,k)
    plot(T, E, 'o', temp, young)
    axis([0,1000,100,250])
    xlabel('Temperature (K)')
    ylabel('Young''s Modulus (GPa)')
    title(['Polynomial of Degree ', num2str(k)])
    residual = E - polyval(P, T);
    error = norm(residual)
    R = sqrt(1-error^2/norm(E)^2)
end
```

Uses an interactive curve-fitting tool to perform the tasks in Example10_06

```
clc % ##command window

T = [ 33, 144, 255, 366, 477, 589, 700, 811, 922];
```

```
E = [220, 213, 206, 199, 192, 185, 167, 141, 105];
plot(T, E, 'o')
axis([0, 1000, 100, 250])
```

Finds a line through origin that best-fits the data points.

```
clc % ##command window

x = [0.008, 0.008, 0.008, 0.008, ...
     0.016, 0.016, 0.016, 0.016, ...
     0.032, 0.032, 0.032, 0.032, ...
     0.064, 0.064, 0.064, 0.064];
y = [4.8,  1.2,  5.7,  4.4, ...
    11.1,  8.6, 13.0, 11.8, ...
    23.1, 18.1, 25.1, 21.4, ...
    42.0, 36.0, 43.2, 37.6];
slope = sum(x.*y)/sum(x.^2);
residual = y-slope*x;
error = norm(residual);
plot(x,y,'o'), hold on
hAxes = gca;
hAxes.XTick = [0.008,0.016,0.032,0.064];
axis([0,0.08,0,70]);
plot([0,0.08],[0,slope*0.08])
text(0.032, 45, ['Slope = ', num2str(slope)])
text(0.032, 40, ['Error = ', num2str(error)])
legend('Measured Data', 'Linear Fit')
title('Original Design')
```

Finds a line through origin that best-fits the data points representing an improved design.

```
clc % ##command window

x = [0.008, 0.008, 0.008, 0.008, ...
     0.016, 0.016, 0.016, 0.016, ...
     0.032, 0.032, 0.032, 0.032, ...
     0.064, 0.064, 0.064, 0.064];
y = [5.3,  4.6,  5.8,  5.4, ...
    12.2, 10.1, 13.2, 11.9, ...
    24.6, 23.1, 25.0, 24.3, ...
    49.3, 47.1, 50.1, 48.2];
slope = sum(x.*y)/sum(x.^2);
residual = y-slope*x;
error = norm(residual);
plot(x,y,'o'), hold on
hAxes = gca;
hAxes.XTick = [0.008,0.016,0.032,0.064];
axis([0,0.08,0,70]);
```

```
plot([0,0.08],[0,slope*0.08])
text(0.032, 45, ['Slope = ', num2str(slope)])
text(0.032, 40, ['Error = ', num2str(error)])
legend('Measured Data', 'Linear Fit')
title('Improved Design')
```

Finds a line through origin that best-fits the data points representing an improved design.

```
clc % ##command window

x = [0.008, 0.008, 0.008, 0.008, ...
     0.016, 0.016, 0.016, 0.016, ...
     0.032, 0.032, 0.032, 0.032, ...
     0.064, 0.064, 0.064, 0.064];
y = [5.3,  4.6,  5.8,  5.4, ...
    12.2, 10.1, 13.2, 11.9, ...
    24.6, 23.1, 25.0, 24.3, ...
    49.3, 47.1, 50.1, 48.2];
slope = sum(x.*y)/sum(x.^2);
residual = y-slope*x;
error = norm(residual);
plot(x,y,'o'), hold on
hAxes = gca;
hAxes.XTick = [0.008,0.016,0.032,0.064];
axis([0,0.08,0,70]);
plot([0,0.08],[0,slope*0.08])
text(0.032, 45, ['Slope = ', num2str(slope)])
text(0.032, 40, ['Error = ', num2str(error)])
legend('Measured Data', 'Linear Fit')
title('Improved Design')
```

Finds a line through origin that best-fits the data points representing an improved design.

```
clc % ##command window

x = [0.008, 0.008, 0.008, 0.008, ...
     0.016, 0.016, 0.016, 0.016, ...
     0.032, 0.032, 0.032, 0.032, ...
     0.064, 0.064, 0.064, 0.064];
y = [5.3,  4.6,  5.8,  5.4, ...
    12.2, 10.1, 13.2, 11.9, ...
    24.6, 23.1, 25.0, 24.3, ...
    49.3, 47.1, 50.1, 48.2];
slope = sum(x.*y)/sum(x.^2);
residual = y-slope*x;
error = norm(residual);
plot(x,y,'o'), hold on
hAxes = gca;
```

```
hAxes.XTick = [0.008,0.016,0.032,0.064];
axis([0,0.08,0,70]);
plot([0,0.08],[0,slope*0.08])
text(0.032, 45, ['Slope = ', num2str(slope)])
text(0.032, 40, ['Error = ', num2str(error)])
legend('Measured Data', 'Linear Fit')
title('Improved Design')
```

Performs linear/spline interpolations.

```
clc % ##command window

T = [ 33, 144, 255, 366, 477, 589, 700, 811, 922];
E = [220, 213, 206, 199, 192, 185, 167, 141, 105];
plot(T, E, 'o'), hold on
temp = 0:10:1000;
young = interp1(T, E, temp);
plot(temp, young, 'r:')
young = interp1(T, E, temp, 'spline');
plot(temp, young, 'b-')
axis([0,1000,100,250])
xlabel('Temperature (K)')
ylabel('Young''s Modulus (GPa)')
legend('Data Points', 'Linear Interpolation', 'Spline Interpolation')
```

Demonstrates how to find a root of an algebraic equation.

```
clc % ##command window

x = linspace(0,pi);
y = sin(x)+2*cos(x)-0.5;
plot(x,y,[0,pi], [0,0]), hold on
axis([0, pi, -2.5, 2])
x1 = interp1(y,x,0)
plot([x1, x1], [-2.5, 2])
text(x1,0.1,num2str(x1))
```

Demonstrates 2-D interpolation.

```
clc % ##command window

[X,Y] = meshgrid(-3:3);
Z = peaks(X,Y);
surf(X,Y,Z)
title('Measured Data')
[X1,Y1] = meshgrid(-3:0.1:3);
Z1 = interp2(X,Y,Z,X1,Y1,'spline');
figure
surf(X1,Y1,Z1)
title('Spline Interpolation')
```

Published with MATLAB® R2018a