# draw2d

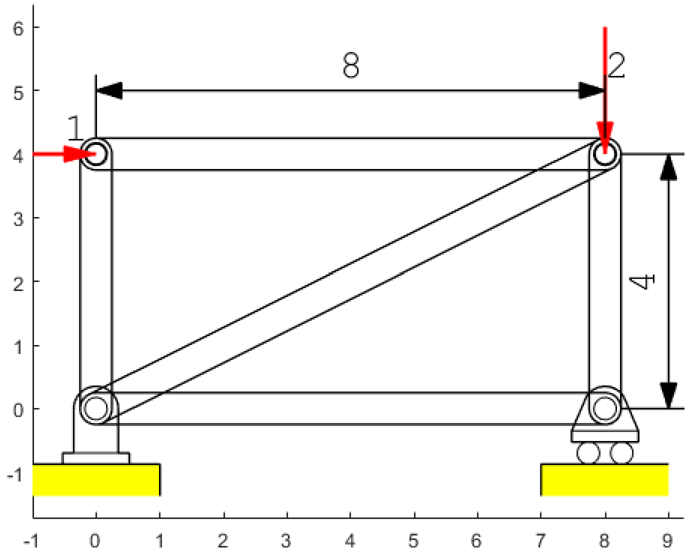
Ver 1.0 , May-June 2019

# Reference manual

# Examples

## Example 1

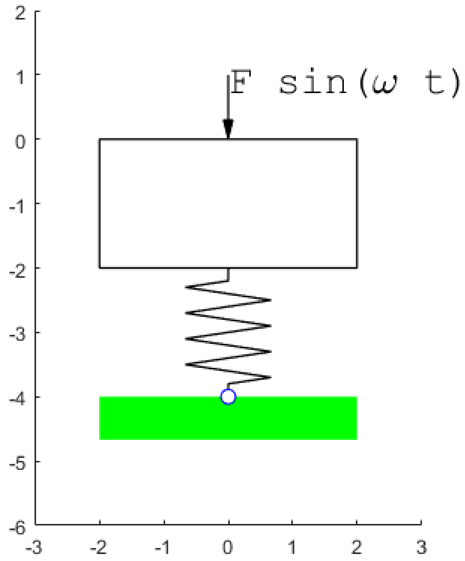
```
drawInit
ht = 0.5;hs = 0.7*ht;F=1;d1=8;d2=4;
% draw truses
drawCanoe1( ht, 0,0,d1,0 );
drawCanoe1( ht, d1,0,d1,d2 );
drawCanoe1( ht, 0,0,d1,d2 );
drawCanoe1( ht, 0,d2,d1,d2 );
drawCanoe1( ht, 0,0,0,d2 );
% draw pins
drawDonut(hs,0.9*hs,0,d2);
drawDonut(hs,0.9*hs,d1,d2);
% draw supports
s1 = drawSupport(1,hs,0,0);
s2 = drawSupport(2,hs,d1,0);
% draw floor
fillRect('y',d2*ht,ht,s1.xk(3),s1.yk(3),'-pos',d1)
fillRect('y',d2*ht,ht,s2.xk(3),s2.yk(3),'-pos',d1)
drawRect(d2*ht,ht,s1.xk(3),s1.yk(3),'-pos',d1,'-v',2,4)
drawRect(d2*ht,ht,s2.xk(3),s2.yk(3),'-pos',d1,'-v',3,1)
% draw forces
f1=drawForce(F,180,0,0,d2,'-ad',0.5,'r','LineWidth',2);
f2=drawForce(2*F,90,0,d1,d2,'-ad',0.5,'r','LineWidth',2);
drawText(f1.xk(3),f1.yk(3),num2str(F));
drawText(f2.xk(3),f2.yk(3),num2str(2*F));
% dimension
drawVDim(3,0.8*ht,ht/2,d1,0,d1,d2,d1+1,d2/2)
drawHDim(3,0.8*ht,ht/2,0,d2,d1,d2,d1/2,d2+1)
```



```
drawShow
```

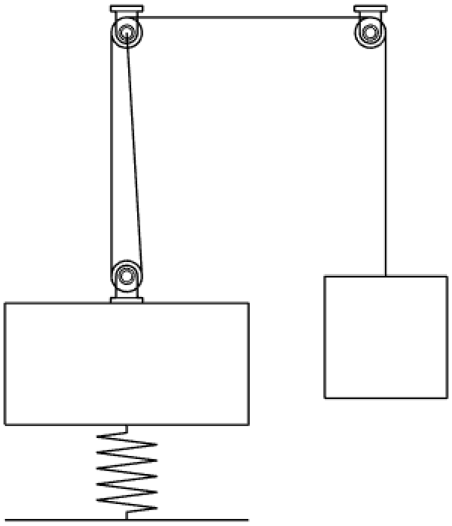
## Example 2

```
% animatio outside Live script
h = 2;w=4;
fg=drawInit;
drawInit(fg)
drawLimits(-3,3,-6,2)
r = drawRect(w,h,0,0,'-pos',8);
s=drawSpring1(2,h/3,4,r.xk(2),r.yk(2),r.xk(2),-2*h);
f = drawForce(1,90,0,0,0,'-ad',0.3);
fillRect('g',w,h/3,s.xk(2),s.yk(2),'-pos',8)
fillDonut('b','w',h/8,h/10,s.xk(2),s.yk(2))
drawText(f.xk(3),f.yk(3),'F sin(\omega t)')
```



## Example 3

```
d1 = 0.5;wd=4;ht=2;h=2+ht;y=0;
drawInit
axis off
c1=drawDonut(d1,d1/2,0,y);
s1 = drawSupport(1,0.7*d1/2,0,y);
r1 = drawRect(wd,ht,s1.xk(3),s1.yk(3),'-pos',8);
sp = drawSpring1(2,d1,5,r1.xk(2),r1.yk(2),r1.xk(2),-h);
drawLine( sp.xk(2),sp.yk(2),'-delta',1,0,-wd/2,2)
s2 = drawSupport(1,0.7*d1/2,0,h,-180);
c2=drawDonut(d1,d1/2,0,h);
s3 = drawSupport(1,0.7*d1/2,wd,h,-180);
c3=drawDonut(d1,d1/2,wd,h);
drawLine(c1.xk(1),c1.yk(1),c2.xk(1),c2.yk(2))
drawLine(c1.xk(2),c1.yk(2),c2.xk(5),c2.yk(5))
drawLine(c2.xk(4),c2.yk(4),c3.xk(4),c3.yk(4))
L1=drawLine(c3.xk(2),c3.yk(2),'-delta',0,-h-y);
drawRect(ht,ht,L1.xk(2),L1.yk(2),'-pos',8)
```



```
drawShow
```

# drawAngDim

# drawAngDim3p

Draw angle dimension

## Description

Angle dimension measures the angle between two directions.

## Syntax

drawAngDim( form, ad1, ad2, xc, yc, sang, ang, td, tang)

drawAngDim3p( form, ad1, ad2, xc, yc, x1, y1, x2, y2, rt, at )

drawAngDim(__,'-str',str)

drawAngDim(_,LineSpec)

### Description

### Method

drawAngDim uses drawArcArrow to draw an angle dimension . For drawing text, the function drawAngDim use current text attributes. They can be changed by function drawSet.

## Arguments

### Input Arguments

**form** - arrowhead form: use 1,2,3,11 for arrow

**d1,d2** - arrowhead width and height

**xc,yc** - the center point

for drawAngDim

**sang** - start angle in degrees

**ang** - central angle in degrees

or for drawAngDim3p

**x1, y1**,  -- the first point

**x2, y2** -- second point

**rt,at** - polar coordinates of text location.

### Optional Name-Value Pair Input Arguments

**'-str'|'-txt',str** - dimension text

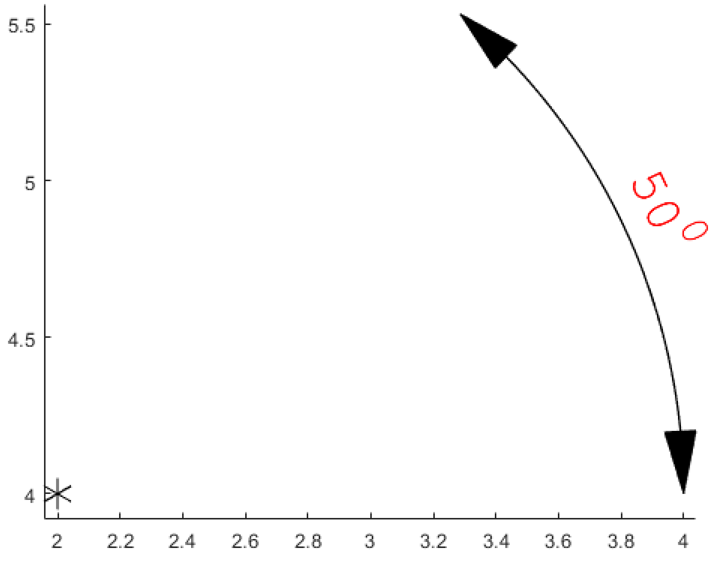
***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

## Examples

### Example 1

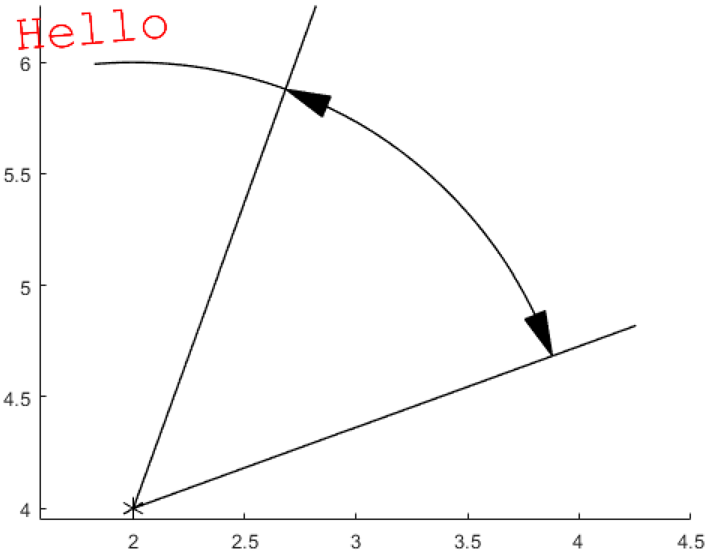
```
drawInit
ad1 = 0.2; ad2 = ad1/2;
x1 = 2; y1 = 4;
r = 2; sang = 0; ang=50; at = ang/2;
gkSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawAngDim(3,ad1,ad2,x1,y1,sang,ang,r,at)
```



```
%grid on
```

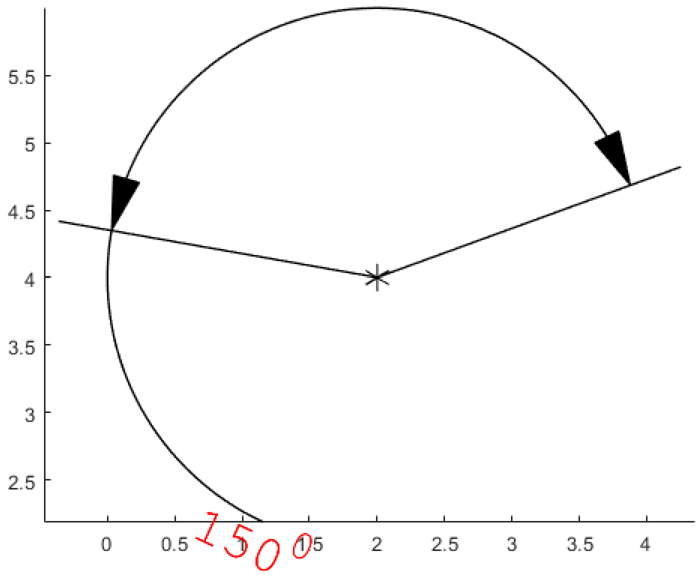
### Example 2

```
drawInit
ad1 = 0.2; ad2 = ad1/2;
x1 = 2; y1 = 4;
r = 2; sang = 20; ang=50; at = 1.5*ang;
gkSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawAngDim(3,ad1,ad2,x1,y1,sang,ang,r,at,'-str','Hello')
drawLine(x1,y1,'-rtheta',1.2*r,sang)
drawLine(x1,y1,'-rtheta',1.2*r,sang+ang)
```



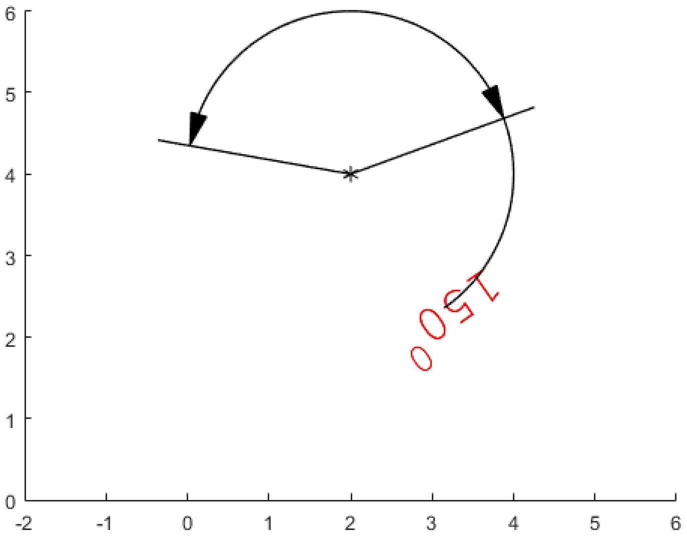
## Example 3

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 4;
r = 2; sang = 20; ang=150; at = 1.5*ang;
gkSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawAngDim(3,ad1,ad2,x1,y1,sang,ang,r,at)
drawLine(x1,y1,'-rtheta',1.2*r,sang)
drawLine(x1,y1,'-rtheta',1.2*r,sang+ang)
```



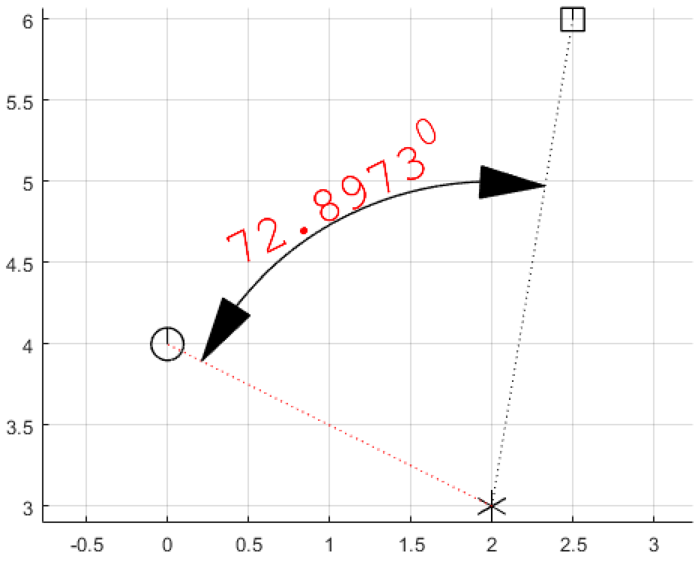
## Example 4

```
drawInit
drawLimits(-2,6,0,6)
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 4;
r = 2; sang = 20; ang=150; at = -0.5*ang;
gkSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawAngDim(3,ad1,ad2,x1,y1,sang,ang,r,at)
drawLine(x1,y1,'-rtheta',1.2*r,sang)
drawLine(x1,y1,'-rtheta',1.2*r,sang+ang)
```



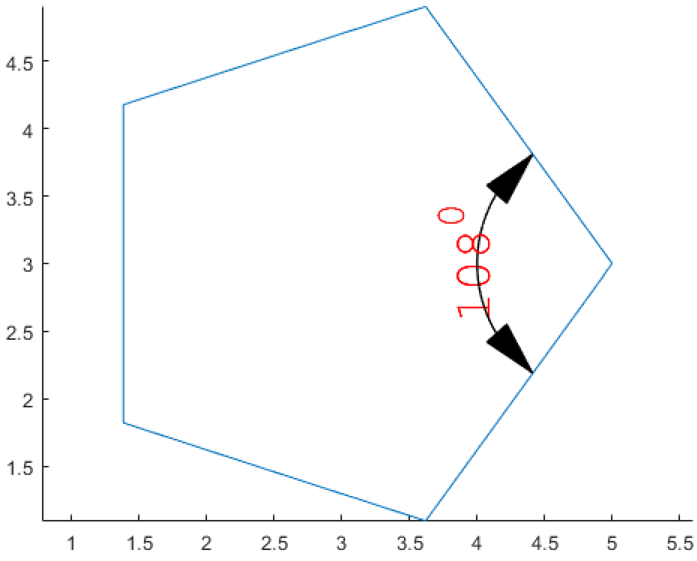
## Example 5

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 2.5; y2 = 6;
xt = 0; yt = 4;
drawSet('FontSize',26,'textColor','r')
drawPoint(1,ad1/2,x1,y1)
drawPoint(2,ad1/2,x2,y2)
drawPoint(3,ad1/2,xt,yt)
drawLine(x1,y1,x2,y2,'k:')
drawLine(x1,y1,xt,yt,'r:')
drawAngDim3p(3,ad1,ad2,x1,y1,x2,y2,xt,yt,2,36); %,'-str','Hello')
grid on
```



## Example 6

```
drawInit
p = drawNgon(5,2,3,3);
%drawSet('FontSize',26,'textColor','r')
%drawPoint(1,ad1/2,x1,y1)
%drawPoint(2,ad1/2,x2,y2)
%drawPoint(3,ad1/2,xt,yt)
drawAngDim3p(3,ad1,ad2,p.xk(1),p.yk(1),p.xk(5),p.yk(5),p.xk(2),p.yk(2),1,-54);
%,'-str','Hello')
```



```
%grid on
```

## See also

## References

# drawArc

Draw circular arc defined by center point and end points.

## Description

## Syntax

drawArc( xc, yc, x1, y1, x2, y2)

drawArc(__,'-large')

drawArc( __,'-np',np)

drawArc( __, LineSpec)

## Description

## Method

## Arguments

### Input Arguments

xc,yc -- center point

x1, y1, -- first point

x2, y2 -- second point

### Optional Input Argumnts

**'-large'** -- draw arc with central angle > 180

**'-pie'|'-sec'** - draw pie (section)

**'-seg'** - segment

### Optional Name-Value Pair Input Arguments

**'-np', np** - number of points along the curve ( scalar integer value > 2)

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

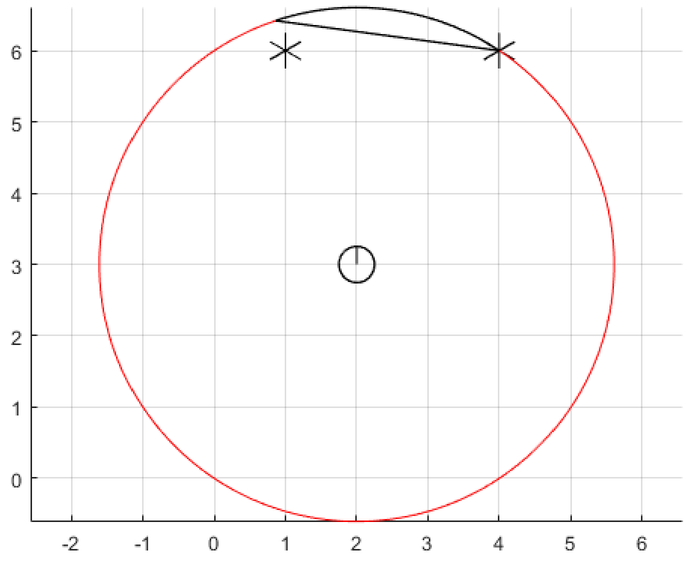
**p** - structure with fields

- p.xk, p.yk - key points: 1=start,2=end,3=center
- p.th - tangent angle: 1=start,2=end
- p.color - line color

# Examples

## Example 1

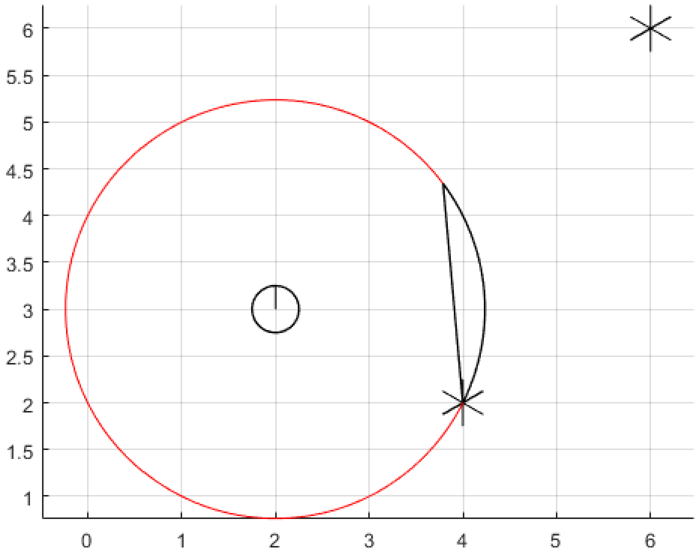
```
drawInit
xc = 2; yc = 3; x1 = 4; y1 = 5; x2 =1; y1 = 6;
drawPoint(3,0.5,xc,yc)
drawPoint(1,0.5,x1,y1)
drawPoint(1,0.5,x2,y2)
drawArc(xc,yc,x1,y1,x2,y2,'-seg')
drawArc(xc,yc,x1,y1,x2,y2,'-large','r')
grid on
```



## Example 2

```
drawInit
xc = 2; yc = 3; x1 = 4; y1 = 5; x2 =6; y1 = 2;
drawPoint(3,0.5,xc,yc)
drawPoint(1,0.5,x1,y1)
drawPoint(1,0.5,x2,y2)
drawArc(xc,yc,x1,y1,x2,y2,'-seg')
drawArc(xc,yc,x1,y1,x2,y2,'-large','r')
```

```
grid on
```



## See Also

## References

# drawArcArrow

Draw the arc arrow between two points.

## Description

Draw arc arrow between two points. Arrowhead is drawn at end point if form > 0 and at start point if form<0. Arrowhead types are taken from IGES 5.3,  4.62 Leader (arrow) entity, pp 259-251 ([1]).

## Syntax

drawArcArrow( form, ad1, ad2, xc, yc, r, sang, ang)

drawArcArrow( __, LineSpec)

p = drawArcArrow( __)

## Description

drawArcArrow( form, ad1, ad2, xc, yc, r, sang, ang)  draw arrow between given points.

drawArcArrow( form, ad1, ad2, xc, yc, r, sang, ang, LineSpec)  set line specification.

## Method

drawArcArrow use function **drawArrowhead** to draw arrowhead

## Arguments

### Input Arguments

**form** - arrowhead type number: 1,...,12 ([1]). If form > 0 then arrow head point is the end point i.e. the arrow is directed from (x1,y1), given by sang  to (x2,y2) given by ang,(see Example 1) if form < 0 then the start point is the arrow tail i.e. the arrow is directed from (x2,y2) to (x1,y1) (see Example 2).

Form Meaning

1. Wedge
2. Triangle
3. Filled Triangle
4. No Arrowhead
5. Circle
6. Filled Circle
7. Rectangle
8. Filled Rectangle

9. Slash
10. Integral Sign
11. Open Triangle
12. Dimension Origin

**ad1** - arrowhead height (>0)

**ad2** - arrowhead width (>0)

**xc, yc** - center

**r** - radius

**sang** - start angle in degrees. The start point (x1,y1): x1 = xc + r*cos(sang), y1 = yc + r*sin(sang)..

**ang** - central angle (arc's angular distance) in degrees: >0 is CCLW, <0 is CCW. The end point is (x2,y2): x2 = xc + r*cos(sang+ang),.y2=yc + r*sin(sang+ang)..

## Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

## Optional output:

**p** - structure with fields

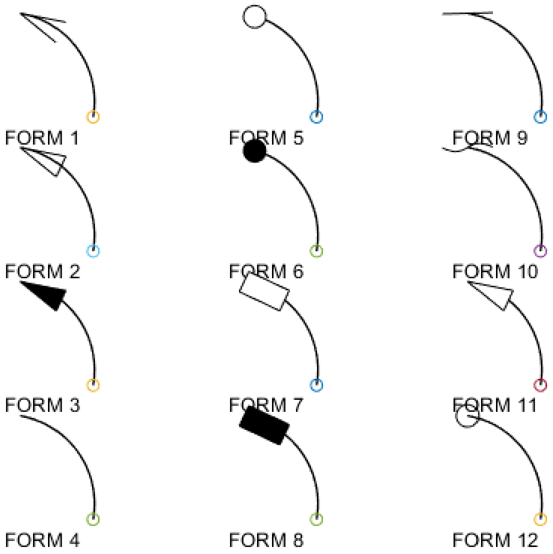
p.xk, p.yk - key points: 1=start, 2 = end

p.color - line color

# Examples

## Example 1

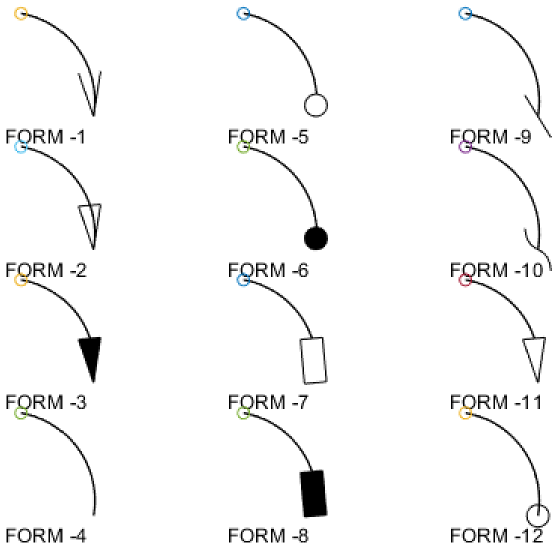
```
figure
hold on
axis equal
dx = 2.5;
dy = 1.5;
k = 0;
x = 0;
y = 3*dy;
ad2 = 0.5/2;
for n = 1:12
    y = y - dy;
    if n == 5 || n == 6 || n == 12
        ad1 = ad2;
    else
        ad1 = 2*ad2;
    end
    % is arrowhead is at end point becouse form > 0
    sang = -10;
    ang = 90;
    r = 1;
    p = drawArcArrow(n,ad1,ad2,x,y,r,sang,ang);
    scatter(p.xk(1),p.yk(1))
    text(x,y-dy/4,sprintf('FORM %d',n))
    k = k + 1;
    if k > 3
        k = 0;
        x = x + dx;
        y = 3*dy;
    end
end
axis off
```



## Example 2

```
figure
hold on
axis equal
dx = 2.5;
dy = 1.5;
k = 0;
x = 0;
y = 3*dy;
ad2 = 0.5/2;
```

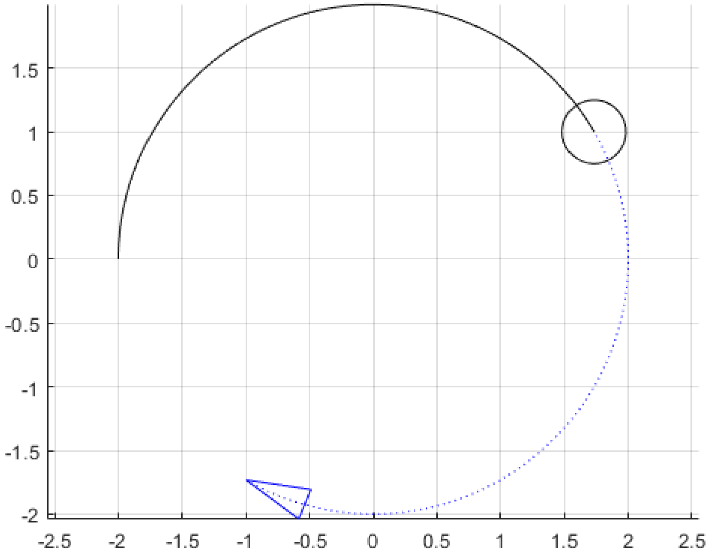
```
for n = 1:12
    y = y - dy;
    if n == 5 || n == 6 || n == 12
        ad1 = ad2;
    else
        ad1 = 2*ad2;
    end
    % is arrowhead is at start point becouse form < 0
    sang = -10;
    ang = 90;
    r = 1;
    p = drawArcArrow(-n,ad1,ad2,x,y,r,sang,ang);
    scatter(p.xk(1),p.yk(1))
    text(x,y-dy/4,sprintf('FORM %d',-n))
    k = k + 1;
    if k > 3
        k = 0;
        x = x + dx;
        y = 3*dy;
    end
end
axis off
```



## Example 3

```
figure
hold on
axis equal
ad1 = 0.5;
```

```
 ad2 = ad1/2;
 drawArcArrow(  2,ad1,ad2,0,0,2,30,-150,'b:') % linestyle does not affect
arrowhead
 drawArcArrow(-12,ad1,ad1,0,0,2,30,150,'k')
 grid on
```



## See Also

## References

[1] IGES Initial Graphics Exchange Specification IGES 5.3

# drawArrow

Draw the arrow between two points.

## Description

Draw arrow between two points. Arrowhead is drawn at end point if form > 0, and at start point if form < 0. Arrowhead types are taken from IGES 5.3, 4.62 Leader (arrow) entity, pp 259-251 ([1]). Note that the leader has always arrowhead at start point.

## Syntax

drawArrow( form, ad1, ad2, x1, y1, x2, y2)

drawArrow( form, ad1, ad2, x1, y1, '-delta', dx, dy)

drawArrow( form, ad1, ad2, x1, y1, '-polar', r, th)

drawArrow( form, ad1, ad2, x1, y1, x2, y2, LineSpec)

p = drawArrow(__)

## Description

drawArrow( form, ad1, ad2, x1, y1, x2, y2) draw arrow between given points. Arrow head is at (x2,y2) if the form > 0, and at (x1,y1) if the form < 0.

drawArrow( form, ad1, ad2, x1, y1, '-delta', dx, dy) end point is given in delta coordinates with respect to start point.

drawArrow( form, ad1, ad2, x1, y1, '-polar', r, th) end point is given in polar coordinates with respect to start point.

drawArrow( form, ad1, ad2, x1, y1, x2, y2, LineSpec) set line specification.

p = drawArrow(__) returns a structure with additional data

## Method

For drawing of arrowhead **drawArrow** use function **drawArrowhead** . For usage of **drawArrowhead** see Example 1. For forms = 5,6,12 **drawArrowhead** use function **evalCircle**. For form = 10 **drawArrowhead** use function **evalBezier.**

The curve is plotted by MATLAB function plot.

## Arguments

### Input Arguments

**form** - arrowhead type number: 1,...,12 ([1]). If form > 0 then arrow head point is the second point point i.e. the arrow is from (x1,y1) to (x2,y2), if form < 0 then the start point is the arrow tail i.e. the arrow is from (x2,y2) to (x1,y1).

Form Meaning

1. Wedge
2. Triangle
3. Filled Triangle
4. No Arrowhead
5. Circle
6. Filled Circle
7. Rectangle
8. Filled Rectangle
9. Slash
10. Integral Sign
11. Open Triangle
12. Dimension Origin

**ad1** - arrowhead height (>0)

**ad2** - arrowhead width (>0)

**x1, y1** - start point

**x2, y2 -** end point (real scalar)

or

**'-polar'|'-rtheta',r,th** - polar coordinates of endpoint with respect to start point

or

**'-delta',dx,dy** - relative coordinates of end point with respect to start point

### Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

### Optional output:

**p** - structure with fields

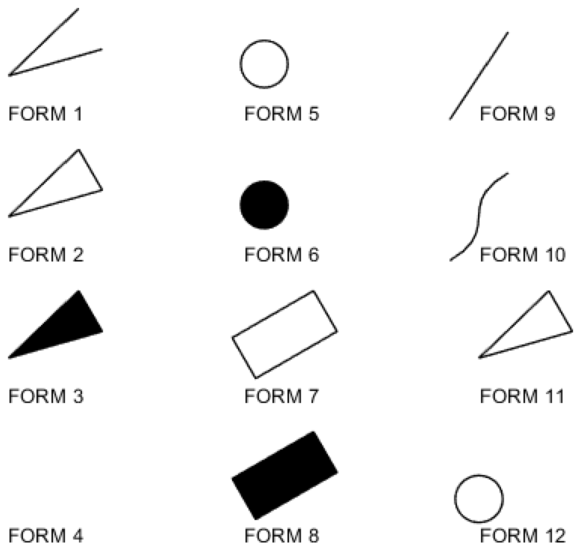
- p.xk, p.yk - key points: 1-start,2-end,3-mid
- p.color - line color

## Examples

### Example 1

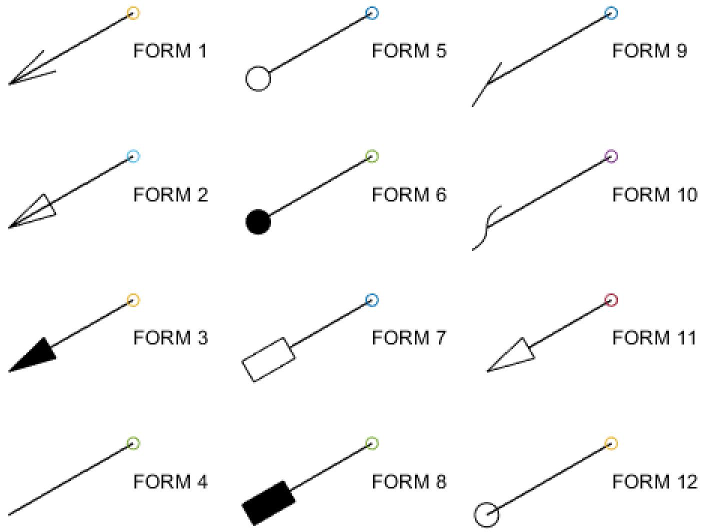
```
%Arrowhead types (IGES 5.3)
```

```
figure
hold on
axis equal
dx = 2.5;
dy = 1.5;
k = 0;
x = 0;
y = 3*dy;
ad2 = 0.5;
for n = 1:12
    y = y - dy;
    if n == 5 || n == 6 || n == 12
        ad1 = ad2;
    else
        ad1 = 2*ad2;
    end
    drawArrowhead(n,ad1,ad2,x,y,210)
    text(x,y-dy/4,sprintf('FORM %d',n))
    k = k + 1;
    if k > 3
        k = 0;
        x = x + dx;
        y = 3*dy;
    end
end
axis off
grid on
```



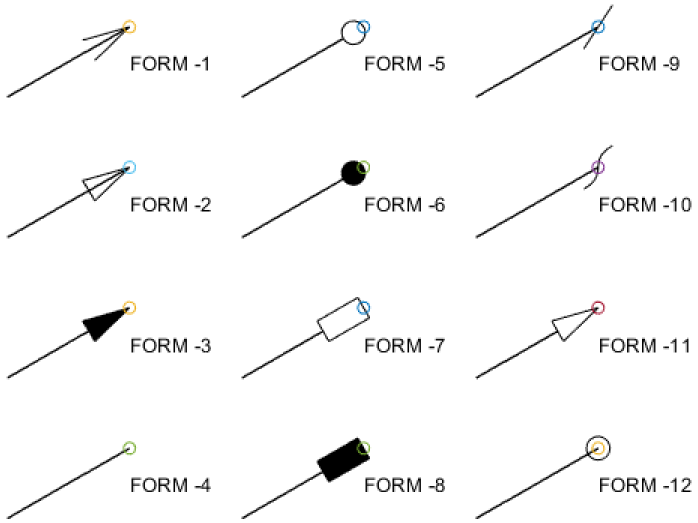
## Example 2

```
%Arrow from (x1,y1) to (x2,y2)
figure
hold on
axis equal
dx = 2.5;
dy = 1.5;
k = 0;
x = 0;
y = 3*dy;
ad2 = 0.5/2;
for n = 1:12
    y = y - dy;
    if n == 5 || n == 6 || n == 12
        ad1 = ad2;
    else
        ad1 = 2*ad2;
    end
    % arrowhead is drawn at end point becouse form > 0
    drawArrow(n,ad1,ad2,x,y,'-polar',1.5,210)
    scatter(x,y)
    text(x,y-dy/4,sprintf('FORM %d',n))
    k = k + 1;
    if k > 3
        k = 0;
        x = x + dx;
        y = 3*dy;
    end
end
axis off
```



## Example 3

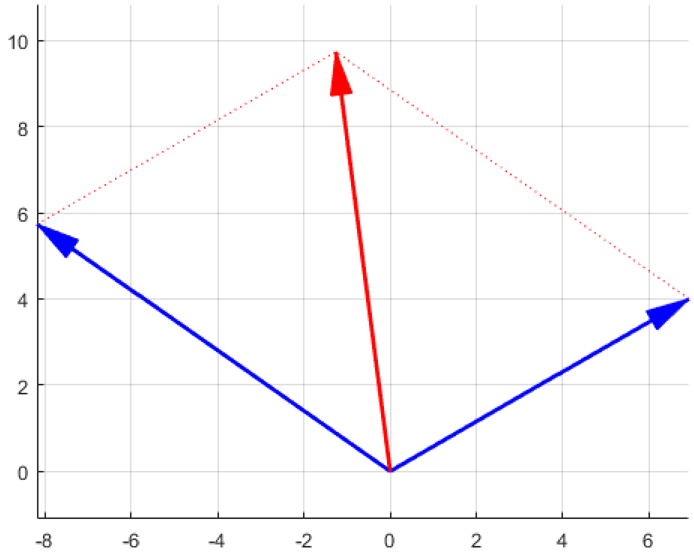
```
%Arrowhead from (x2,y2) to (x1,y1)
figure
hold on
axis equal
dx = 2.5;
dy = 1.5;
k = 0;
x = 0;
y = 3*dy;
ad2 = 0.5/2;
for n = 1:12
    y = y - dy;
    if n == 5 || n == 6 || n == 12
        ad1 = ad2;
    else
        ad1 = 2*ad2;
    end
    % x,y is arrow head point becouse form<0
    drawArrow(-n,ad1,ad2,x,y,'-polar',1.5,210)
    scatter(x,y)
    text(x,y-dy/4,sprintf('FORM %d',-n))
    k = k + 1;
    if k > 3
        k = 0;
        x = x + dx;
        y = 3*dy;
    end
end
axis off
```



## Example 4

```
%Arrowhead types (IGES 5.3)
figure
hold on
axis equal
dx = 2.5;
dy = 1.5;
k = 0;
x = 0;
y = 3*dy;
ad1 = 1;
ad2 = ad1/2;
an1 = 30;
an2 = 145;
F1 = 8;
F2 = 10;
a1 = drawArrow(3,ad1,ad2,0,0,'-polar',F1,an1,'LineWidth',2,'Color','b');
a2 = drawArrow(3,ad1,ad2,0,0,'-polar',F2,an2,'LineWidth',2,'Color','b');
s1 = drawLine(a2.xk(2),a2.yk(2),'-polar',F1, an1,'r:');
s2 = drawLine(a1.xk(2),a1.yk(2),'-polar',F2,an2,'r:');
drawArrow(3,ad1,ad2,0,0,s1.xk(2),s1.yk(2),'LineWidth',2,'Color','r');
```

```
grid on
```



## See Also

## References

[1] IGES Initial Graphics Exchange Specification IGES 5.3

# drawAxes

Draw coordinate axes.

## Description

## Syntax

drawAxes(form,d1,d2,xc,yc)

drawAxes(form,d1,d2,xc,yc,LineSpec)

p = drawAxes(__)

## Description

## Method

## Arguments

### Input Arguments

**form** - arrowhead type: use 1,2,3,11 for arrow

**d1** - width of arrowhead

**d2** - length of axes

**xc** - the center point

**yc** -

### Optional input

**rot** - axes rotation angle

### Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

**p** - structure with fields

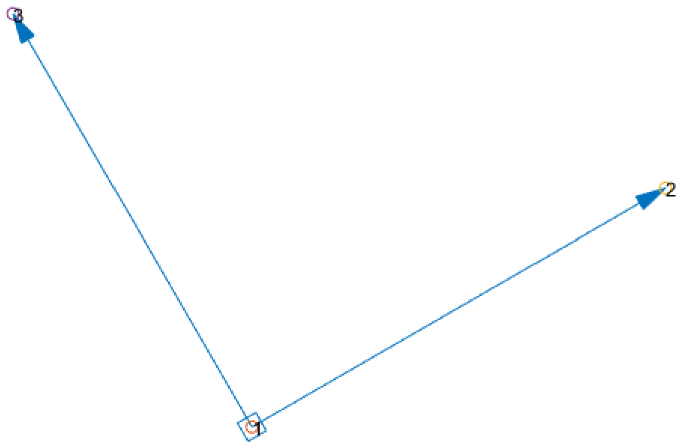
- p.xk, p.yk -- key points

- p.color - line color

# Examples

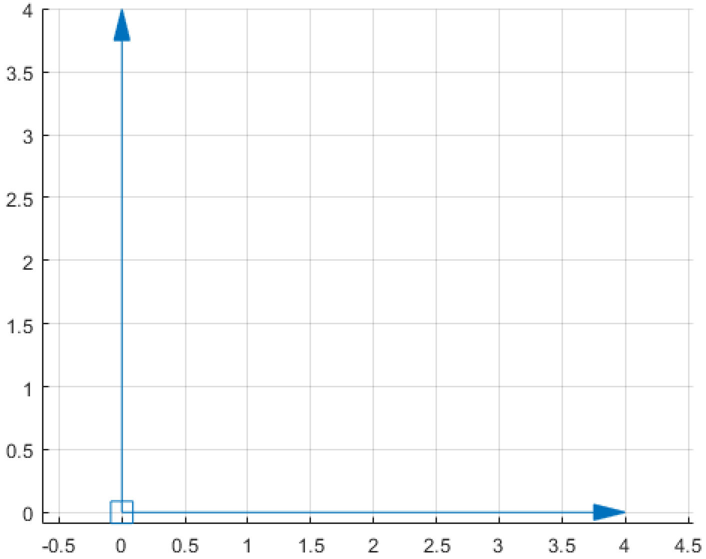
## Example 1

```
figure
hold on
axis equal
axis off
a = drawAxes( 2, 0.25, 4, 0, 0, 30);
for k = 1:length(a.xk)
    scatter(a.xk(k),a.yk(k))
    text(a.xk(k),a.yk(k),num2str(k))
end
grid on
```



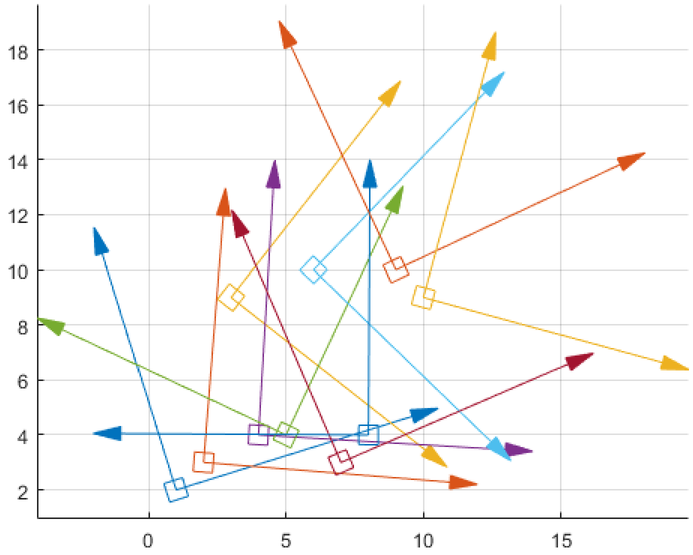
## Example 2

```
figure
axis equal
hold on
drawAxes( 2, 0.25, 4, 0, 0)

grid on
```



## Example 3

```
figure
axis equal
hold on
for k = 1:10
    drawAxes( 2, 1, 10,k, randi(10,1,1), 45*randn)
end
grid on
```



## Example 4

## Principal components analysis in two-dimensions

```
drawInit
np = 1000;
xx = 10*randn(np,1);
yy = randn(np,1);
th = 180*rand(np,1);
x = xx.*cosd(th) - yy.*sind(th);
y = xx.*sind(th) + yy.*cosd(th);
scatter(x,y,20,'.')
p=PCA2d(x,y)
```

```
p = struct with fields:
     xm: -0.0758
     ym: -0.3990
     s1: 7.3495
     s2: 6.9394
    th1: 4.9269
    th2: 94.9269
      S: [2×2 double]
      C: [2×2 double]
```

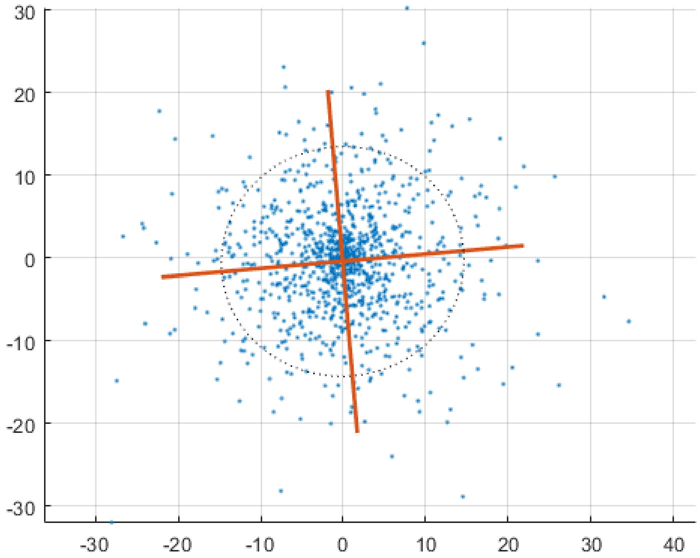
```
p.C
```

```
ans = 2×2
    0.9963    0.0859
   -0.0859    0.9963
```

```
[coeff]=pca([x,y])
```

```
coeff = 2×2
    0.9963   -0.0859
    0.0859    0.9963
```

```
r1=drawCross(3*p.s1,3*p.s2,p.xm,p.ym,p.th1,'LineWidth',2);
drawEllipse(2*p.s1,2*p.s2,p.xm,p.ym,p.th1,'k:')
grid on
```



## See also

## References

# drawBezier

Draw 2-D Bezier curve.

## Description

The Bezier curve is a polynomal blending function which interpolates between the first and the last vertices ([1],[2]). The curve points are given by

$$\langle x(t), y(t) \rangle = \sum_{k=1}^{n} \langle x_k, y_k \rangle B_k(t), \quad 0 \le t \le 1$$

where $\langle x_k, y_k \rangle$ are coordinates of *k*-th vertix, and $B_k(t)$ are basis functions given by

$$B_k(t) \equiv C_k^{n-1} t^k (1-t)^{n-1-k}, \quad k = 1, \ldots, n-1$$

## Syntax

drawBezier(xp,yp)

drawBezier(xp,yp,LineSpec)

drawBezier(xp,yp,'-np',np)

p = drawBezier(__)

## Description

drawBezier(xp,yp) draw Bezier curve with default number of points.

drawBezier(xp,yp,LineSpec) sets the line style, marker symbol, and color.

drawBezier(xp,yp,'-np',np) draw curve with np points.

p = drawBezier(__) returns structure with fields contain x-value and y-value for the curve.

## Method

For calculation of the coordinates of the curve **drawBezier** call function **evalBezier( xp, yp, t)** which returns coordinates *x* and *y* of the curve at given values of parameter *t*. Function **evalBezier** is based on subroutine BEZIER from [1] (pp 225-226). For usage of **evalBezier**, see Example 1.

## Arguments

### Input Arguments

**xp -** x-coordinate of polygon vertices (real vector)

**yp** - y-coordinate of polygon vertices (real vector)

xp, yp must be of the same size.

## Optional Name-Value Pair Input Arguments

**'-np', np** - number of points along the curve, np is scalar integer value > 2. Default:100

***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

**p** - structure with the fields

- p.x, p.y - points on curve
- p.xk, p.yk - key points: 1=start,2=end
- p.th - tangent angle in degrees: 1=start,2=end
- p.color - line color

*Note*: p.x, p.y are row vectors if xp is row vector, and coulmn vectors if xp is coulmn vector.

# Examples

## Example 1

Example 5-3 from [1] (pp 143-144).

```
% coordinates of polygon vertices
xp=[1 2 4 3];
yp=[1 3 3 1];
% values from the book
tt = [0 0.15 0.35 0.5 0.65 0.85 1]; % parameter values
xt = [1, 1.5, 2.248, 2.75, 3.122, 3.248, 3];
yt = [1, 1.675, 2.367, 2.5, 2.36, 1.75, 1];
% calculate values for tt
[xb,yb]=evalBezier(xp,yp,tt);
% print coordinates and differences
fprintf('%4s%10s%10s%10s%10s\n','n','xb','aerr','yb','aerr');
```

```
   n        xb      aerr        yb      aerr
```

```
for n = 1:length(xb)
    fprintf('%4d%10f%10f%10f%10f\n',n,xb(n),xb(n)- xt(n),yb(n),yb(n)-yt(n));
end
```

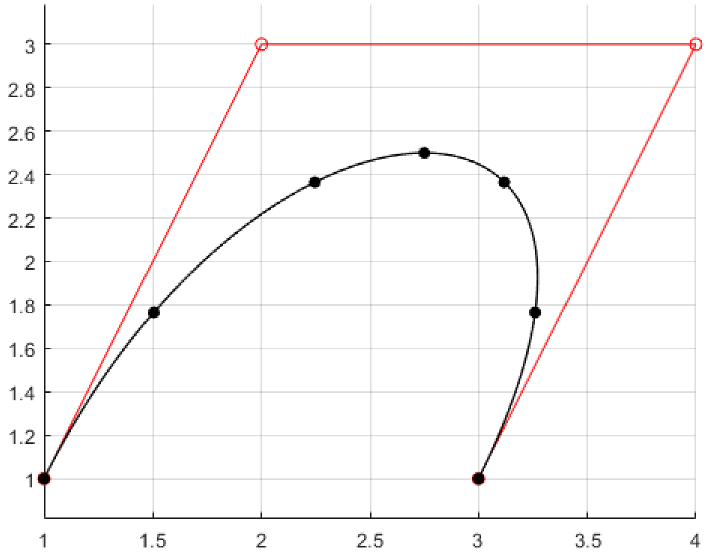
```
   1  1.000000  0.000000  1.000000  0.000000
   2  1.504000  0.004000  1.765000  0.090000
   3  2.246000 -0.002000  2.365000 -0.002000
   4  2.750000  0.000000  2.500000  0.000000
   5  3.119000 -0.003000  2.365000  0.005000
   6  3.261000  0.013000  1.765000  0.015000
   7  3.000000  0.000000  1.000000  0.000000
```

## Example 2

```
figure
hold on
axis equal
% plot definiting polygon vertices
scatter(xp,yp,'r')
% plot definiting polygon
plot(xp,yp,'r')
% plot calculated points
scatter(xb,yb,'k','filled');
% draw curve
p = drawBezier(xp,yp)
```

```
p = struct with fields:
        x: [100×1 double]
        y: [100×1 double]
       xk: [1 3]
       yk: [1 1]
       th: [63.4349 -116.5651]
    color: 'k'
```

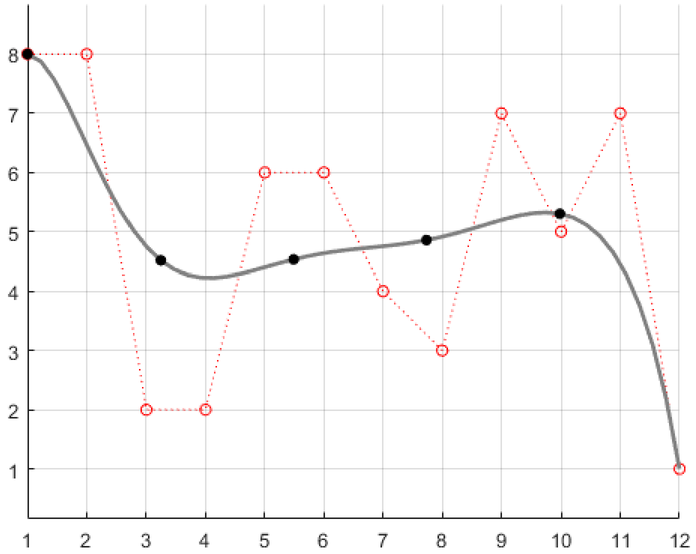
```
grid on
```



### Example 3

```
figure
axis equal
hold on
% coordinates of polygon vertices
nv = 12; % number of vertices
```

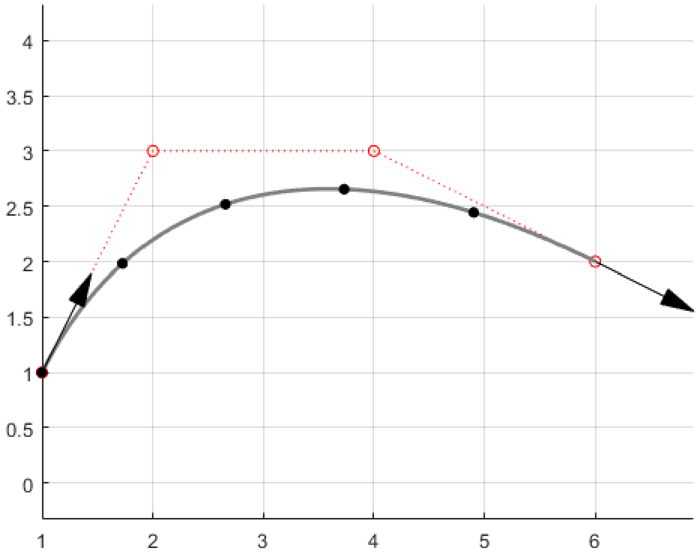
```
xv=1:nv; %randi(10,nv,1);
yv=randi(10,nv,1);
% plot vertices
scatter(xv,yv,30,'r')
% plot definiting polygon
plot(xv,yv,'r:')
% plot calculated points
% draw curve using 50 points in gray color
b = drawBezier(xv,yv,'-np',50,'LineWidth',2,'Color',[1 1 1]*0.5);
% label every 10th point
scatter(b.x(1:10:end),b.y(1:10:end),30,'k','filled')
grid on
```



### Example 4

```
% intercept invaid data
figure
axis equal
hold on
% coordinates of polygon vertices
xv = [ 1 2 4 6];
yv = [ 1 3 3 2];
% plot vertices
scatter(xv,yv,30,'r')
% plot definiting polygon
plot(xv,yv,'r:')
% plot calculated points
% draw curve using 50 points in gray color
b = drawBezier(xv,yv,'-np',50,'LineWidth',2,'Color',[1 1 1]*0.5);
```

```
% label every 10th point
scatter(b.x(1:10:end),b.y(1:10:end),30,'k','filled')
drawArrow(3,0.3,0.3/2,xv(1),yv(1),'-rtheta',1,b.th(1),'k')
drawArrow(3,0.3,0.3/2,xv(4),yv(4),'-rtheta',1,b.th(2),'k')
grid on
```



## References

[1] Rogers, Adams, *Mathematical elements for Computer Graphics*, McGraw-Hill,1976

[2] WikipediA, Bezier curve

# drawBspline

Draw 2-D B-spline curve.

## Description

## Syntax

drawBspline(c,xp,yp)

drawBspline(c,xp,yp,LineSpec)

drawBspline(c,xp,yp,'-np',np)

p = drawBspline(__)

### Description

drawBspline(c,xp,yp) draw B-spline curve with default number of points.

drawBspline(__,LineSpec) sets the line style, marker symbol, and color.

drawBspline(__,'-np',np) draw curve with np points.

p = drawBspline(__) returns structure with fields contain x-value and y-value for the curve.

### Method

## Arguments

### Input Arguments

**c** - order of B-spline basis

**xp -** x-coordinate of polygon vertices (real vector)

**yp** - y-coordinate of polygon vertices (real vector)

xp, yp must be of the same size.

### Optional Name-Value Pair Input Arguments

**'-np', np** - number of points along the curve, np is scalar integer value > 2.

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

**p** - structure with the fields

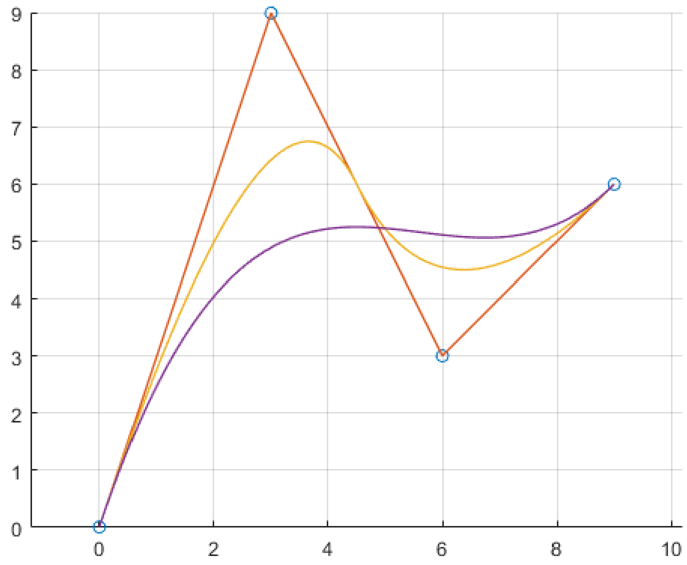
- p.x, p.y - points on curve
- p.xk, p.yk - key points: 1=start,2=end
- p.th - tangent angle in degrees: 1=start,2=end
- p.color - line color

# Examples

## Example 1

Example 5-3 from [1] (pp 147-150).

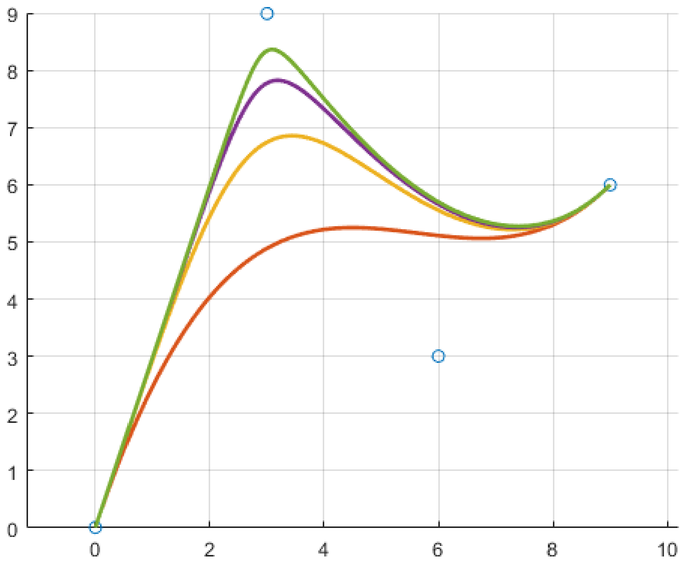
```
xv = [ 0 3 6 9];
yv = [ 0 9 3 6];

figure
clf
axis equal
hold on
scatter(xv,yv)
plot(xv,yv,'r:')
grid on
for k = 2:4
    drawBspline(k,xv,yv,'LineWidth',1);
end
```



## Example 2

Example 5-3 from [1] (pp 147-150).

```
figure
clf
axis equal
hold on
xv = [ 0 3 6 9];
yv = [ 0 9 3 6];
scatter(xv,yv)
xv = [ 0 3 6 9];
yv = [ 0 9 3 6];
drawBspline(4,xv,yv,'LineWidth',2);
xv = [ 0 3 3 6 9];
yv = [ 0 9 9 3 6];
drawBspline(5,xv,yv,'LineWidth',2);
xv = [ 0 3 3 3 6 9];
yv = [ 0 9 9 9 3 6];
drawBspline(6,xv,yv,'LineWidth',2);
xv = [ 0 3 3 3 3 6 9];
yv = [ 0 9 9 9 9 3 6];
drawBspline(7,xv,yv,'LineWidth',2);
grid on
```



# References

[1] Rogers, Adams, *Mathematical elements for Computer Graphics*, McGraw-Hill,1976

# drawCanoe

# drawCanoe1

# fillCanoe

# fillCanoe1

Draw or fill canoe i.e., rectangle with rounded ends.

## Description

Canoe is IGES flash entity,  form number 4. (see [1], pp 120,123)

## Syntax

drawCanoe(wd,ht,xr,yr)

drawCanoe(wd,ht,xr,yr,rot)

drawCanoe1( ht,x1,y1,x2,y2)

drawcanoe1(ht,x1,y1,'-delta',dx,dy)

drawCanoe1(ht,x1,y1,'-polar',r,th)

drawCanoe(__,'-pos',ip)

drawCanoe(__,'-np',np)

drawCanoe(__,LineSpec)

p = drawCanoe(__)

fillCanoe(c,__)

## Description

drawCanoe(wd,ht,xr,yr,rot) draw canoe of length wd and height ht rotated by given angle *rot* about reference point *xr*,*yr*.

drawCanoe(wd,ht,xr,yr,rot,'-pos',ip) draw canoe with reference point at position ip:1,...,11, (def. is 5, i.e., canoe center)

drawCanoe(__,LineSpec)  sets the line style.

p = drawCanoe(__) returns some output data.

## Method

For calculation of the coordinates of the drawCanoe call the function **evalRect** and **evalCircle**.

# Arguments

## Input Arguments

**c** - fill color (for fillCanoe)

**wd** - width

**ht** - height

**xr, yr** - reference point

**rot** - rotation angle about the reference point in degrees

or for drawCanoe1

**x1, y1 -** start point (key point 4)

**x2, y2 -** end point (key point 6)

or instaed of x2,y2

**'-rtheta',r,th** - polar coordinates of end point from start point

or

**'-delta',dx,dy** - shift vector

## Optional Name-Value Pair Input Arguments

**'-pos',ip** - position of reference point:1,...,11 (default is 5, i.e. center) (see Example 2)

***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

**p** - structure .

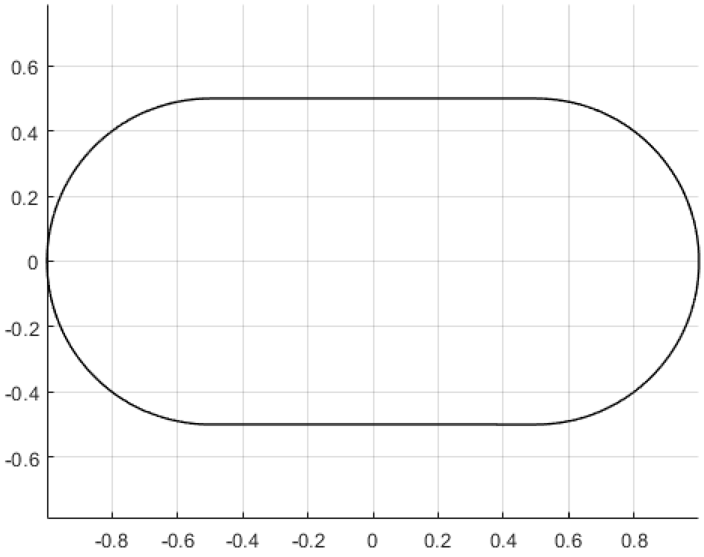
- p.wd - width;
- p.ht - height;
- p.x - x-coordinates of the reference points (real column vector)
- p.y - y coordinates of the reference points (real column vector)
- p.style - line style
- p.width - line width
- p.color - line color

*Note. Rectangle* is closed so x(end) = x(1) and y(end) = y(1).

# Examples

## Example 1

```
figure
hold on
axis equal
% close polygon
drawCanoe( 2, 1,0,0,0);
grid on
```

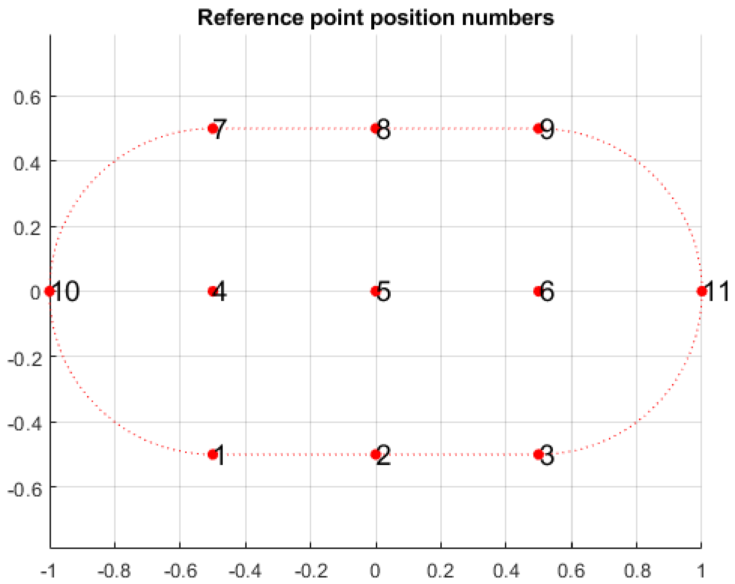


## Example 2

```
figure
axis equal
hold on
% plot definiting polygon
p=drawCanoe(2,1,0,0,0,'r:')
```

```
 p = struct with fields:
        xk: [11×1 double]
        yk: [11×1 double]
     color: [1 0 0]
```

```
% plot referebnce points
scatter(p.xk,p.yk,30,'r','filled')
for k = 1:length(p.xk)
    text(p.xk(k),p.yk(k),num2str(k),'FontSize',14)
end
title('Reference point position numbers')
grid on
```

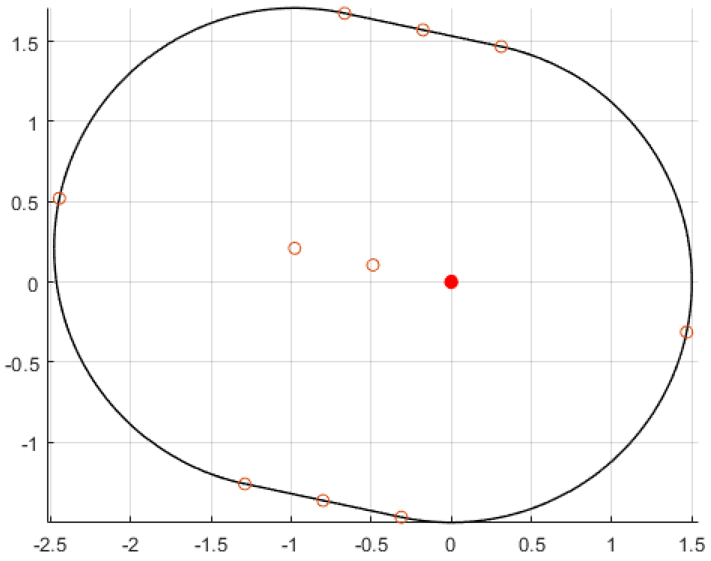


## Example 3

```
figure
axis equal
hold on
ip = 6; % rotation about position point
p=drawCanoe(4,3,0,0,-12,'-pos',ip,'-np',100)
```

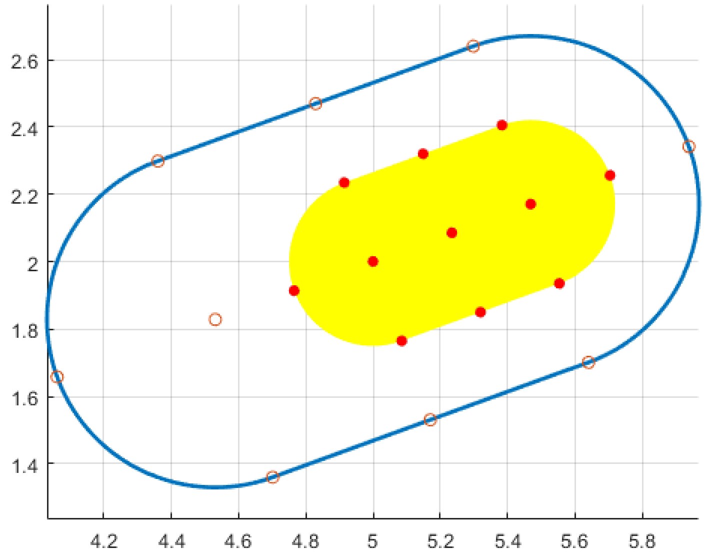
```
p = struct with fields:
       xk: [11×1 double]
       yk: [11×1 double]
    color: 'k'
```

```
scatter(p.xk,p.yk)
scatter(p.xk(ip),p.yk(ip),50,'r','filled')
grid on
```



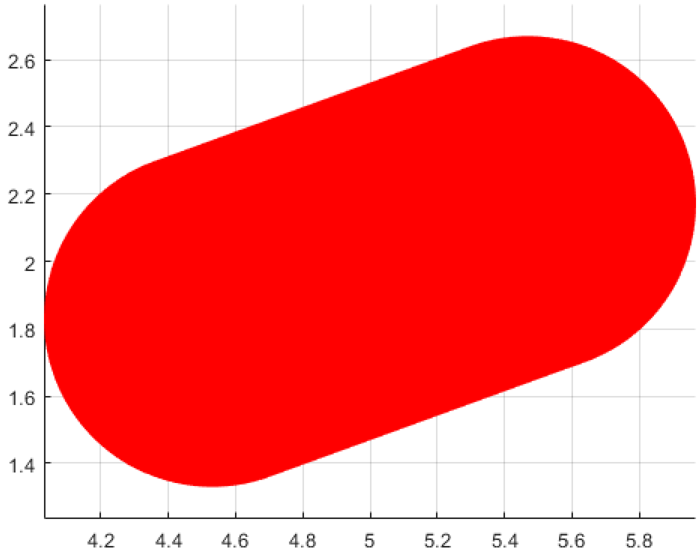
## Example 4

```
figure
axis equal
hold on
r1=drawCanoe(2,1,5,2,20,'LineWidth',2);
scatter(r1.xk,r1.yk)
r2=fillCanoe('y',1,0.5,r1.xk(5),r1.yk(5),20,'-pos',4);
scatter(r2.xk,r2.yk,30,'r','filled')
grid on
```



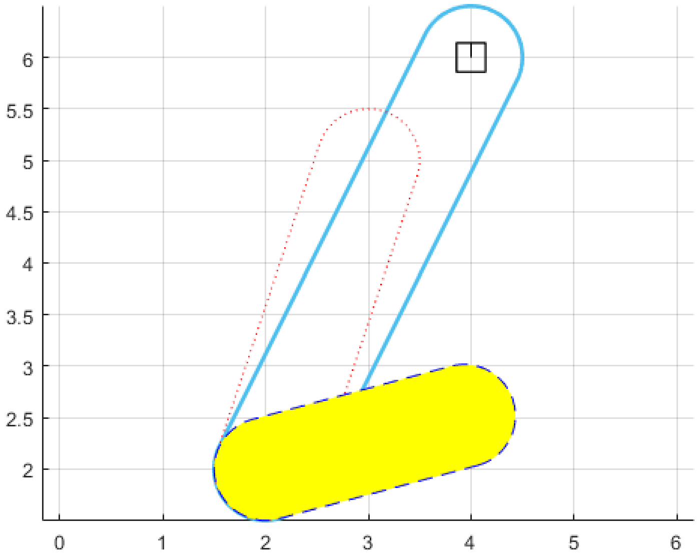
## Example 5

```
figure
axis equal
hold on
r1=fillCanoe('r',2,1,5,2,20);
grid on
```



## Example 6

```
figure
axis equal
hold on
x1 = 2; y1 = 2;
x2 = 4; y2 = 6;
drawPoint(1,0.4,x1,y1)
drawPoint(2,0.4,x2,y2)
drawCanoe1(1,x1,y1,x2,y2,'LineWidth',2);
drawCanoe1(1,x1,y1,'-delta',1,3,'r:');
fillCanoe1('y',1,x1,y1,'-polar',2,15);
drawCanoe1(1,x1,y1,'-polar',2,15,'b--');
grid on
```



## See also

## References

[1] IGES Initial Graphics Exchange Specification IGES 5.3

# drawCatenary

Draw 2-D catenary

## Description

The catenary is the curve that an idealized hanging homogeneous chain or cable assumes under its own weight when supported only at its ends [1]. The coordinates of the curve points are given by [2]

$$\begin{aligned} x(s) &= x_1 + \lambda\left[\sinh^{-1}\left(\frac{s}{\lambda}\right) - \sinh^{-1}\left(\frac{s_1}{\lambda}\right)\right] \\ y(s) &= y_1 + \lambda\left[\sqrt{1+\left(\frac{s}{\lambda}\right)^2} - \sqrt{1+\left(\frac{s_1}{\lambda}\right)^2}\right] \end{aligned}, \quad s_1 \le s \le s_2$$

where *s is arc-length parameter,* $(x_1, y_1)$ are the coordinates of the catenary starting point, *and* $\lambda$ is characteristic length. The length of catenary $L$ , the horizontal distance (span) $\Delta x > 0$ and vertical distance $\Delta y$ between the two supports *are* given by

$$L = s_2 - s_1, \quad \Delta x = x_2 - x_1, \quad \Delta y = y_2 - y_1.$$

The catenary appex $(x_a, y_a)$ is at $s = 0$. The sags  of  of the supports are

$$h_1 = y_1 - y_a, \quad h_2 = y_2 - y_a$$

## Syntax

drawCatenary(,x1,y1,dx,dy,L)

drawCatenary(,x1,y1,dx,dy,'-L',L)

drawCatenary(,x1,y1,dx,dy,'-h',h1)

drawCatenary(x1,y1,dx,dy,L,LineSpec)

drawCatenary(x1,y1,dx,dy,L,'-np',np)

p = drawCatenary(__)

## Description

drawCatenary(x1,y1,dx,dy,L) draw catenary  with default number of points.

drawCatenary(x1,y1,dx,dy,L,LineSpec)  sets the line style, marker symbol, and color.

drawCatenary(x1,y1,dx,dy,L,'-np',np) draw catenary with np points.

p = drawCatenary(__) returns structure with fields contain x-value and y-value for the curve.

## Method

For calculation of the coordinates of the curve **drawCatenary** call function **evalCatenary( x1,y1,lambda, s1, s)** which returns coordinates *x* and *y* of the curve at given values of parameter s. For usage of **evalCatenary** see Example 1.

When span $\Delta x$, verical distance between supports $\Delta y$, and catenary lengt $L$ are given then characteristic length $\lambda$ is obtained by solving the equation

$$2\lambda^2 \cosh\frac{\Delta x}{\lambda} = L^2 - \Delta y^2$$

Once $\lambda$ is known one can calculate parameter value at end points by

$$s_1 = \frac{1}{2}\left(\sqrt{\frac{L^2 - \Delta y^2 + 4\lambda^2}{L^2 - \Delta y^2}} - L\right), \qquad s_2 = L - s_1 .$$

When span $\Delta x$, verical distance between supports $\Delta y$, and sag $h_1$ of start support are given then characteristic length $\lambda$ is obtained by solving the equation

$$\lambda\left[\cosh^{-1}\left(1 + \frac{\Delta y + h_1}{\lambda}\right) + \cosh^{-1}\left(1 + \frac{h_1}{\lambda}\right)\right] = \Delta x$$

Once $\lambda$ is known one can calculate parameter value at end points by

$$s_1 = -\sqrt{h_1(h_1 + 2\lambda)}, \qquad s_2 = \sqrt{(h_1 + \Delta y)(h_1 + \Delta y + 2\lambda)},$$

where we assume that the appex is between suports.

The solution of above equations solution is obtained numerically by using MATLAB function fzero**.** Default initial quess is in both cases is $\lambda = 1$.

Catenary is plotted by MATLAB function plot.

# Arguments

## Input Arguments

**x1 -** x-coordinate of starting point (real scalar)

**y1** - y-coordinate of starting point (real scalar)

**dx** - horizontal distance between end points (real scalar, >0)

**dy** - vertical distance between end points (real scalar)

## Optional arguments

**L** - catenary length (real scalar ,>0)

## Optional Name-Value Pair Input Arguments

**'-L', L** - catenary length (real scalar >0)

**'-h', h1** - the vertical  distance from appex to starting point (real scalar)

**'-np', np** - number of points along the curve ( scalar integer value, > 1)

**'-guess',z0** - initial quess for **fzero**, (real scalar, > 0)

***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

**p** - structure with field.

- p.type - 'Catenary'
- p.x - x-coordinates of the curve (real column vector)
- p.y - y coordinates of the curve (real column vector)
- p.xa - x coordinate of the curve appex
- p.ya - y coordinate of the curve appex
- p.lambda - characteristic length
- p.s1 - parameter starting value (real scalar)
- p.s2 - parameter end value (real scalar)
- p.L - catenary length
- p.dx - horizontal distance between end points
- p.dy - vertical distance between end points
- p.style - line style
- p.width - line width
- p.color - line color

# Examples

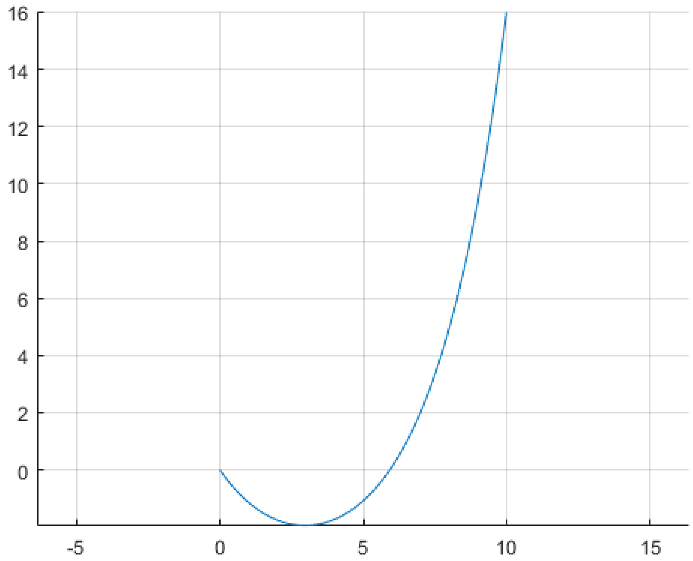
## Example 1

Example from [2] (pp 132-133).

```
%Data
a = 10;
b = 16;
L = 24;
%Values from book
%c  = 2.530;  % characteristic length
%xa = 2.96;  % appex x position
%ya = 4.470;
figure
axis equal
hold on
p=drawCatenary(0,0,a,b,L)
```

```
p = struct with fields:
      type: 'Catenary'
         x: [100×1 double]
         y: [100×1 double]
        xa: 2.9622
        ya: -1.9394
    lambda: 2.5323
        s1: -3.6856
        s2: 20.3144
         L: 24
        dx: 10
        dy: 16
        h1: 1.9394
        h2: 17.9394
        d1: -2.9622
        d2: 7.0378
        xk: [0 10 2.9622]
        yk: [0 16 -1.9394]
     color: [0 0.4470 0.7410]
```

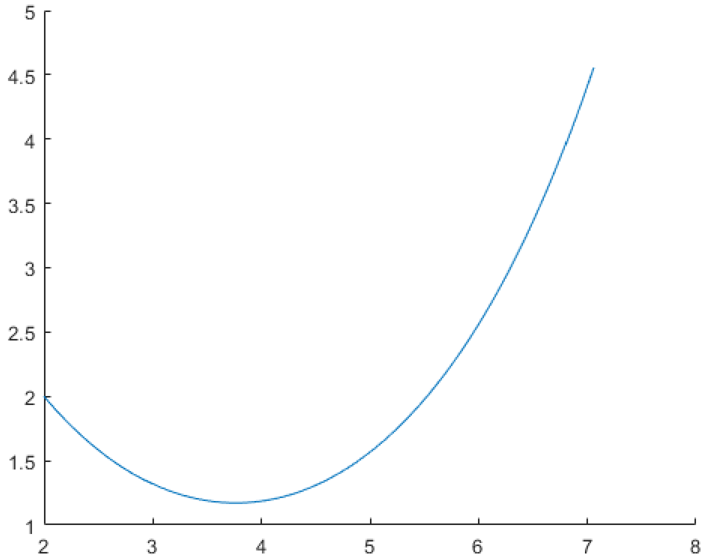
```
grid on
```



## Example 2

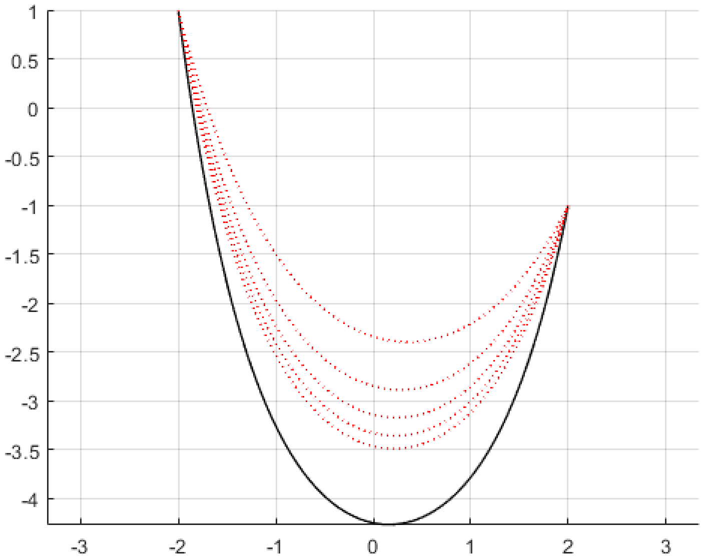
```
figure
hold on
c = 2; % characteristic length
x1 = 2; y1 = 2; % starting point
L1 = 2; % lengt of left part of catenary
L2 = 5; % lenfth of right part of catenary
[x,y] = evalCatenary(c,x1,y1,-L1,linspace(-L1,L2));
```

```
plot(x,y)
```



## Example 3

```
 figure
 hold on
 axis equal       % !!!
 L = 10;          % catenaary length
 x1 = -2; y1 = 1; % start point
 dx = 4; dy = -2; % horizontal and verticale distance between suports
 % draw curves
 drawCatenary(x1,y1,dx,dy,L,'LineWidth',1,'Color','k');
 for k = 2:6

drawCatenary(x1,y1,dx,dy,L*k/(k+1),'LineWidth',1,'LineStyle',':','Color','r');
 end
 grid on
```



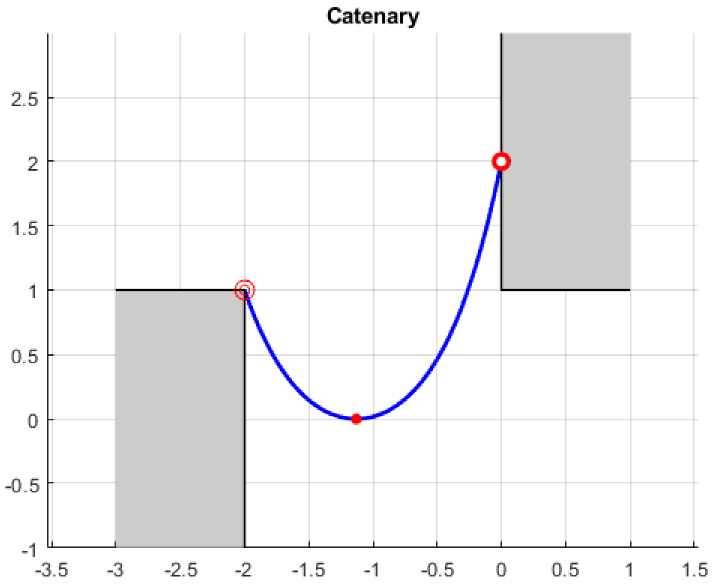
## Example 4

```
drawInit
title('Catenary')
col = 0.8*[1 1 1]; % light gray
h1 = 1; % sag of first point
x1 =-2; y1 = 1; % start point location
dx = 2; dy = 1; % horizontal and vertical distance between supports
c = drawCatenary(x1,y1,dx,dy,'-h',h1,'LineWidth',2,'Color','b')
```

```
c = struct with fields:
      type: 'Catenary'
         x: [100×1 double]
         y: [100×1 double]
        xa: -1.1301
        ya: -2.2204e-16
    lambda: 0.4892
        s1: -1.4065
        s2: 2.4406
         L: 3.8471
        dx: 2.0000
        dy: 1.0000
        h1: 1.0000
        h2: 2.0000
        d1: -0.8699
        d2: 1.1301
        xk: [-2 -4.4409e-16 -1.1301]
        yk: [1 2.0000 -2.2204e-16]
     color: [0 0 1]
```

```
% mark appex
scatter(c.xa,c.ya,30,'r','filled')
```

```
% draw wals
b1=fillRect(col,1,2,c.x(end),c.y(end),0,'-pos',4,'EdgeColor','none');
drawRect(1,2,c.x(end),c.y(end),0,'-v',4,2,'-pos',4,'k','LineWidth',1)
b2=fillRect(col,1,2,c.x(1),c.y(1),0,'-pos',9,'EdgeColor','none');
drawRect(1,2,c.x(1),c.y(1),0,'-v',2,4,'-pos',9,'k','LineWidth',1);
% draw pins
drawDonut(0.15,0.075,c.x(1),c.y(1),'r')
fillDonut('r','w',0.15,0.075,c.x(end),c.y(end))
grid on
```



## See Also

## References

[1] WikipediA, Catenary

[2] J.Prescott, Mechanics of Particls and Rigid Bodies, Longmans, 1966, Ch 8

# drawCircle

# fillCircle

Draw or fill the Circular arc

## Description

Parametric equations of the circular arc are

$$x = x_c + r\cos\theta,\ y = y_c + r\sin\theta, \quad \theta_0 \le \theta \le \theta_0 + \Delta\theta.$$

where *r* is radius.

## Syntax

drawCircle( xc, yc, r)

drawCircle( xc, yc, r, sang, theta)

drawCircle( xc, yc, r, sang, theta, '-pie')

drawCircle( xc, yc, r, sang, theta, '-seg')

drawCircle( __, '-np',np)

drawCircle( __,  LineSpec)

p = drawCircle(__)

fillCircle(c, __)

## Description

drawCircle( xc, yc, r) draw the circle with default number of points 360 and current line specification.

drawCircle( xc, yc, r, sang, theta) draw the circular arc with  number of points set to fix(theta) and current line specification.

drawCircle( xc, yc, r, sang, theta,'-pie') draw a pie i.e. connect end points with the center.

drawCircle( xc, yc, r, sang, theta,'-seg') draw a segment i.e. connect end points.

drawCircle( __, LineSpec)  sets the line style, line width, and color.

drawCircle( __,'-np',np) set the number of points on the output curve.

p = drawCircle(__) returns structure with fields contains coordinates of end points.

## Method

For calculation of the coordinates of the curve **drawCircle** call function **evalCircle( xc,yc,r,th)** which returns coordinates *x* and *y* of the curve at given angles th . For usage of **evalCircle** see Example 1.

The curve is plotted by MATLAB function plot.

# Arguments

## Input Arguments

**c** - fill color

**xc -** x-coordinate of the center (real scalar)

**yc** - y-coordinate of the center (real scalar)

**r** - circle radius (real scalar)

## Optional Input Argumnts

**sang** - start angle in degrees

**theta** - central angle in degrees: > 0 if CCW, <0 if CW

**'-pie'|'-sec'** - draw pie (section)

**'-seg'** - segment

## Optional Name-Value Pair Input Arguments

**'-np', np** - number of points along the curve ( scalar integer value > 2)

***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

**p** - structure with fields

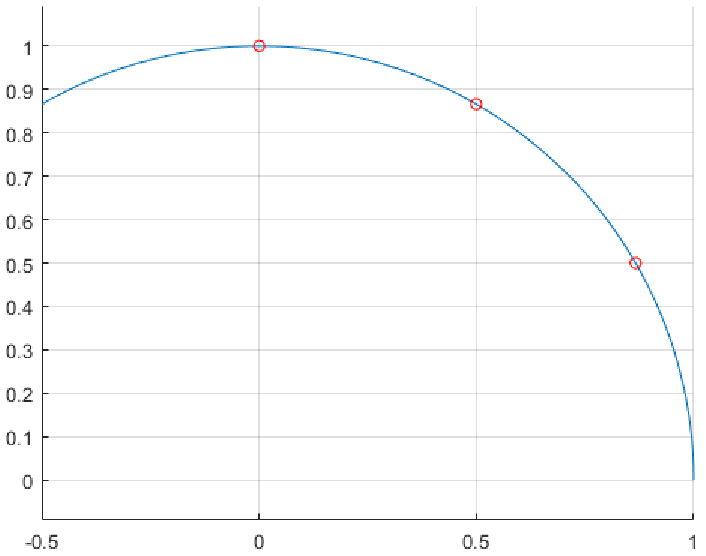
- p.xk, p.yk - key points: 1=start,2=end,3=center
- p.th - tangent angle: 1=start,2=end
- p.color - line color

# Examples

## Example 1

```
%Data
r = 1;
sang = 0;
theta = 120;
% plot
figure
```

```
hold on
axis equal
[x,y] = evalCircle(0,0,r,linspace(sang,sang + theta));
plot(x,y)
[x,y] = evalCircle(0,0,r,[30,60,90]); % points on circle
scatter(x,y,30,'r')
grid on
```



## Example 2

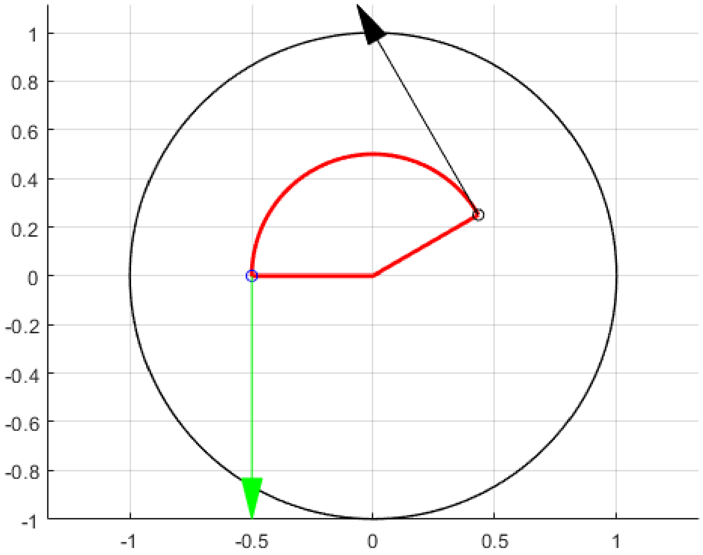
```
%Data
xc = 0;
yc = 0;
r = 1;
sang = 30;
theta = 150;
figure
hold on
axis equal
c1 = drawCircle(xc,yc,r)
```

```
c1 = struct with fields:
       xk: [-1 1 0 0 0]
       yk: [0 0 -1 1 0]
       th: [90 90]
    color: 'k'
```

```
c2 = drawCircle(xc,yc,r/2,sang,theta,'-pie','LineWidth',2,'Color','r','-
np',100)
```

```
c2 = struct with fields:
        xk: [0.4330 -0.5000 0]
        yk: [0.2500 0 0]
        th: [120.0000 -90]
     color: [1 0 0]
```

```
scatter(c2.xk(1),c2.yk(1),30,'k')
scatter(c2.xk(2),c2.yk(2),30,'b')
drawArrow(3,0.5/3,0.25/3,c2.xk(1),c2.yk(1),'-rtheta',1,c2.th(1),'k')
drawArrow(3,0.5/3,0.25/3,c2.xk(2),c2.yk(2),'-rtheta',1,c2.th(2),'g')
grid on
```



## Example 3

```
%Data
xc = 0;
yc = 0;
r = 1;
sang = 30;
theta = 150;
figure
hold on
```

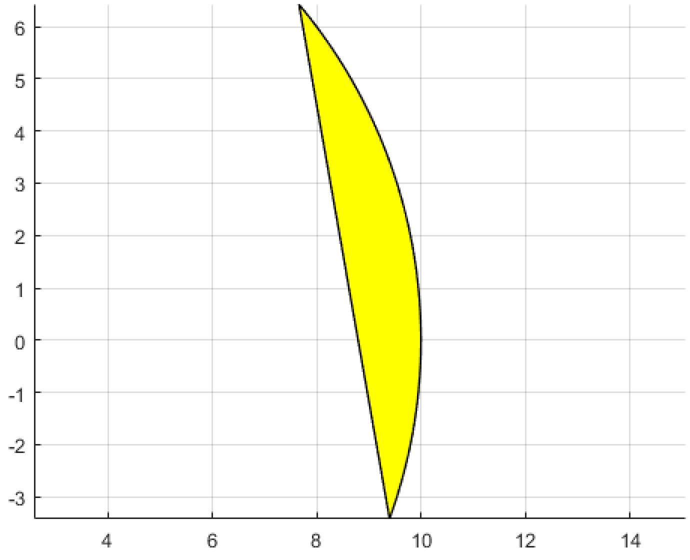
```
axis equal
c1 = drawCircle(xc,yc,r);
c2 = drawCircle(xc,yc,r/2,sang,theta,'-pie','LineWidth',2,'Color','r','-np',100);
c3 = fillCircle('g',xc,yc,r/3,-sang/3,-theta,'-seg','LineWidth',1,'-np',100);
grid on
```

## Example 4

```
% intercept errors
figure
hold on
axis equal
p = drawCircle(0,0,10,2,i,'-np',16)
grid on
```

## Example 6

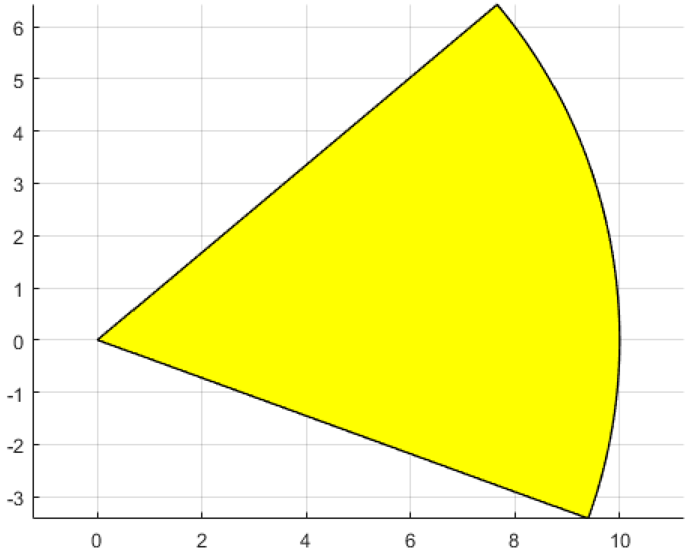
```
figure
hold on
axis equal
p = fillCircle('y',0,0,10,-20,60,'-seg');
p = drawCircle(0,0,10,-20,60,'-seg');
grid on
```



## Example 7

```
figure
hold on
```

```
axis equal
p = fillCircle('y',0,0,10,-20,60,'-sec');
p = drawCircle(0,0,10,-20,60,'-sec');
grid on
```



## See Also

## References

# drawCOG

Draw center of gravity symbol

## Description

## Syntax

drawCOG(d,xc,yc)

drawCross(d,xc,yc,rot)

drawCross(d,xc,yc,FillSpec)

### Description

drawCOG(d,xc,yc)  draw COG symbol with diameter d and center at  (xc,yc).

drawCross(d1,d2,xc,yc,rot)  set rotation angle.

p = drawCross(__) returns an output data.

### Method

The function drawCOG call function **filleCircle**.

## Arguments

### Input Arguments

**d** - diameter

**xc**  -  center point

**yc**  -

### Optional Input Arguments

**rot** - rotation angle in degrees, >0 is CCLW direction and <0 is CLW direction

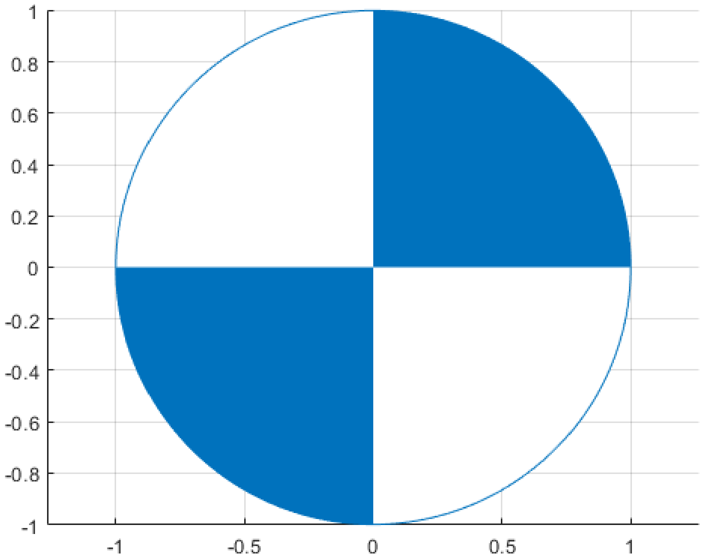
### Optional Name-Value Pair Input Arguments

***FillSpec*** - specifies fill properties

## Examples

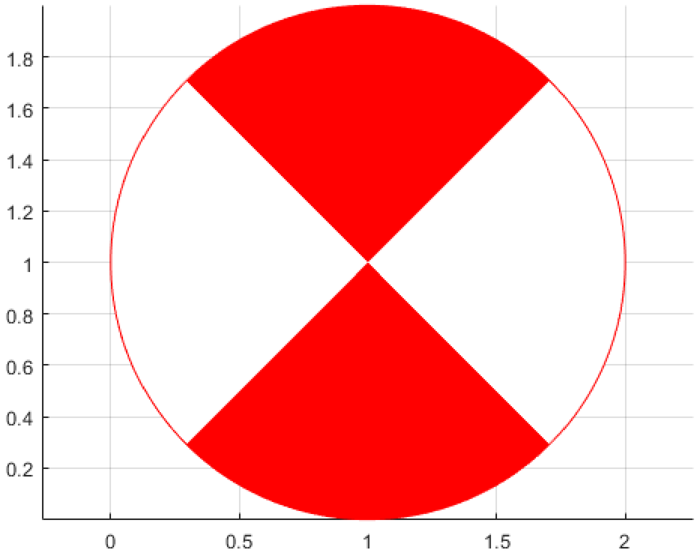
## Example 1

```
figure
hold on
axis equal
drawCOG( 2, 0, 0);
grid on
```



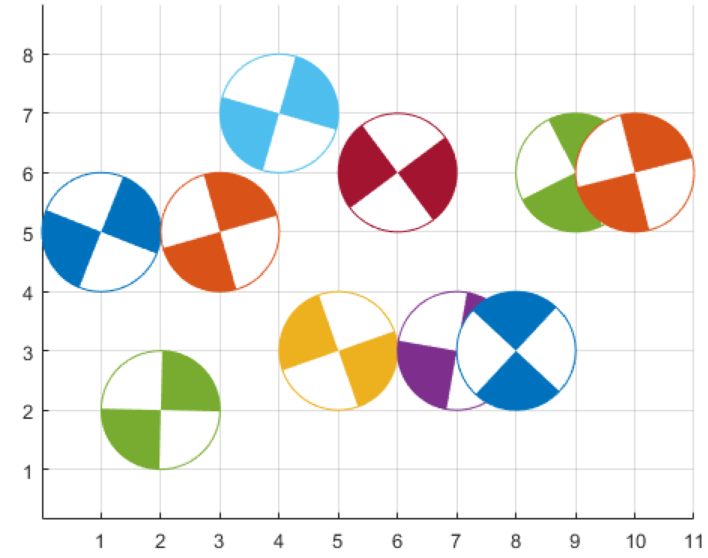
## Example 2

```
figure
axis equal
hold on
drawCOG(2,1,1,45,'r')
grid on
```



### Example 3

```
figure
axis equal
hold on
for k = 1:10
    drawCOG(2,k,randi(10,1,1),randn*45,'LineWidth',1);
end
grid on
```



## See also

## References

# drawCross

Draw two intersecting lines perpendicular to each other.

## Description

## Syntax

drawCross(d1,d2,xc,yc)

drawCross(d1,d2,xc,yc,rot)

drawCross(d1,d2,xc,yc,rot,LineSpec)

p = drawCross(__)

### Description

drawCross(d1,d2,xc,yc)  draw cross with center (xc,yc) and length of arms  d1 and d2.

drawCross(d1,d2,xc,yc,rot)  draw cross at center (xc,yc) with length of arms  d1 and d2 and rotated by *rot* about the center.

p = drawCross(__) returns an output data.

### Method

The function **drawCross** call function **drawLine**.

The curve is plotted by MATLAB function plot.

## Arguments

### Input Arguments

**d1** - length of cross arm in major direction

**d2** - length of cross arm in minor direction

**xc**     -  center point

**yc**     -

### Optional Input Arguments

**rot** - rotation angle in degrees, >0 is CCLW direction and <0 is CLW direction

### Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

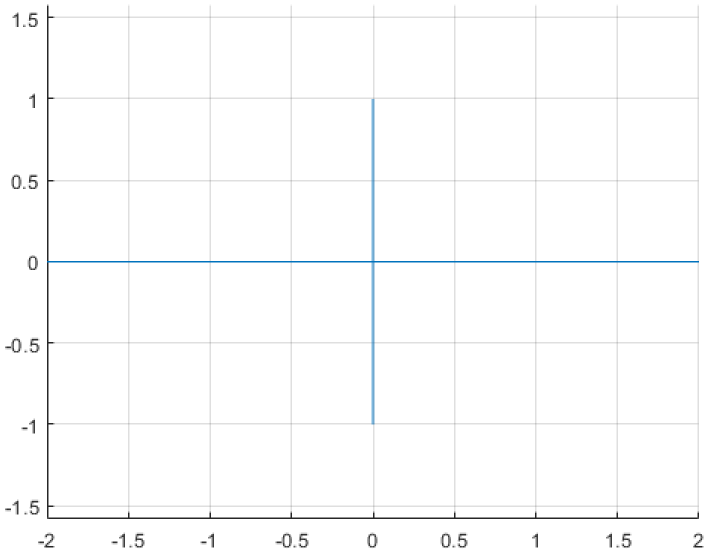
**p** - structure with fields

- p.style - line style
- p.width - line width
- p.color - line color

# Examples

## Example 1

```
figure
hold on
axis equal
drawCross( 2, 1, 0, 0);
grid on
```

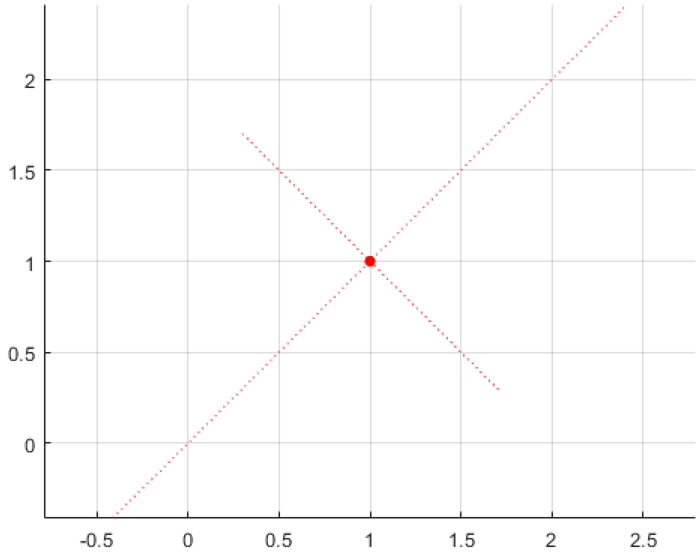


## Example 2

```
figure
axis equal
hold on
p=drawCross(2,1,1,1,45,'r:')
```

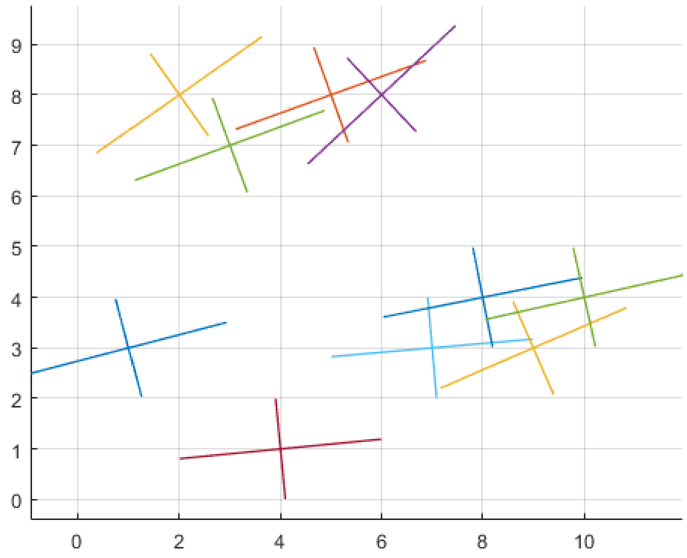
```
p = struct with fields:
       xk: [-0.4142 2.4142 1.7071 0.2929 1]
       yk: [-0.4142 2.4142 0.2929 1.7071 1]
    color: [1 0 0]
```

```
% plot referebnce points
scatter(1,1,30,'r','filled')
grid on
```



### Example 3

```
figure
axis equal
hold on
for k = 1:10
    drawCross(2,1,k,randi(10,1,1),45*rand,'LineWidth',1);
end
grid on
```



## Example 4

Principal components analysis in two-dimensions

```
drawInit
np = 1000;
xx = 10*randn(np,1);
yy = randn(np,1);
th = 180*rand(np,1);
x = xx.*cosd(th) - yy.*sind(th);
y = xx.*sind(th) + yy.*cosd(th);
scatter(x,y,20,'.')
p=PCA2d(x,y)
```

```
p = struct with fields:
     xm: -0.0716
     ym: 0.1032
     s1: 7.5353
     s2: 7.0537
    th1: -17.2350
    th2: 72.7650
      S: [2×2 double]
      C: [2×2 double]
```

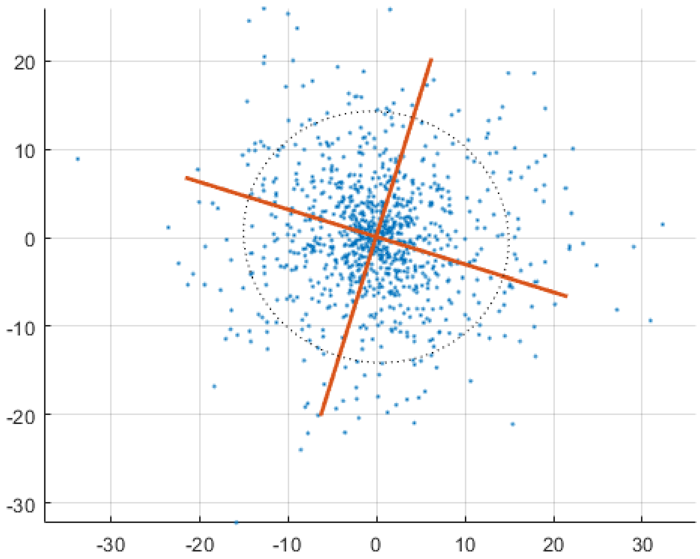
```
p.C
```

```
ans = 2×2
    0.9551   -0.2963
    0.2963    0.9551
```

```
[coeff]=pca([x,y])
```

```
coeff = 2×2
    0.9551    0.2963
   -0.2963    0.9551
```

```
r1=drawCross(3*p.s1,3*p.s2,p.xm,p.ym,p.th1,'LineWidth',2);
drawEllipse(2*p.s1,2*p.s2,p.xm,p.ym,p.th1,'k:')
grid on
```



## See also

## References

# drawDim

Draw parallel dimension

## Description

Parallel dimension measures  the  distance between two points.

## Syntax

drawDim(form,d1,d2,x1,y1,x2,y2,xt,yt)

drawDim(__,'-str',str)

drawDim(_,LineSpec)

### Description

drawDim(form,d1,d2,x1,y1,x2,y2,xt,yt) draw  dimension between points (x1,y1) and (x2,y2) and locate text, i.e. value of distnce x2-x1, at point (xt,yt). If x2 < x1 than the points are swaped.

drawDim(__,'-str',str) draw  dimension between points (x1,y1) and (x2,y2) and locate text given by variable str at point (xt,yt).

drawDim(__,LineSpec)  sets the line style, line width, and color.

### Method

drawDim use drawLine, drawArrohwhead, and gkText to draw a horizontal dimension . For drawing text the function drawDim use current text attrubtes. They can be changed by function drawSet.

## Arguments

### Input Arguments

**form** - arrowhead form

**d1,d2** - arrow head width and height

**x1,y1** - start point. x1 colud be > x2

**x2,y2** - end point

**xt,yt**    - text location. If xt is inside interval (x1,x2) then text is located at the center of the interval

### Optional Name-Value Pair Input Arguments

**'-str'|'-txt',str** - dimension text

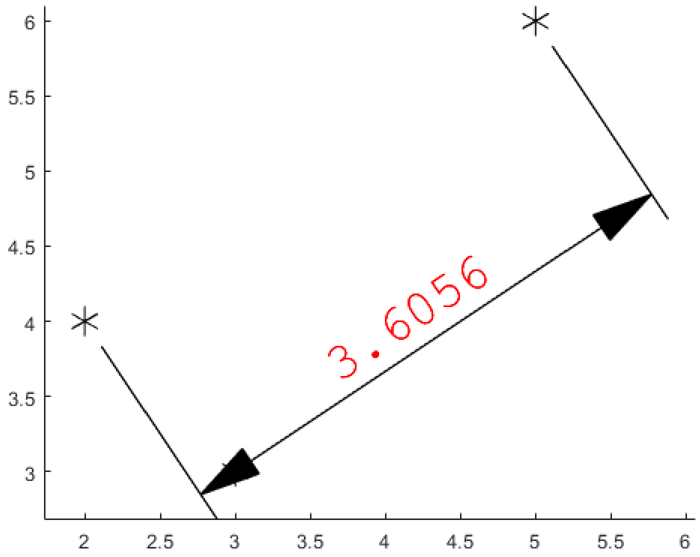
***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

# Examples

## Example 1

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 4;
x2 = 5; y2 = 6;
xt = 3; yt = 3;
gkSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawPoint(1,ad1/2,x2,y2)
drawPoint(1,ad1/2,xt,yt)
drawDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
```

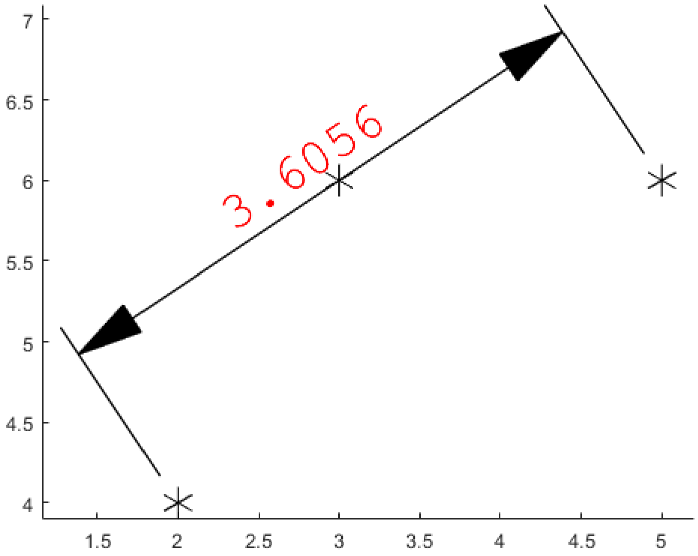


```
%grid on
```

## Example 2

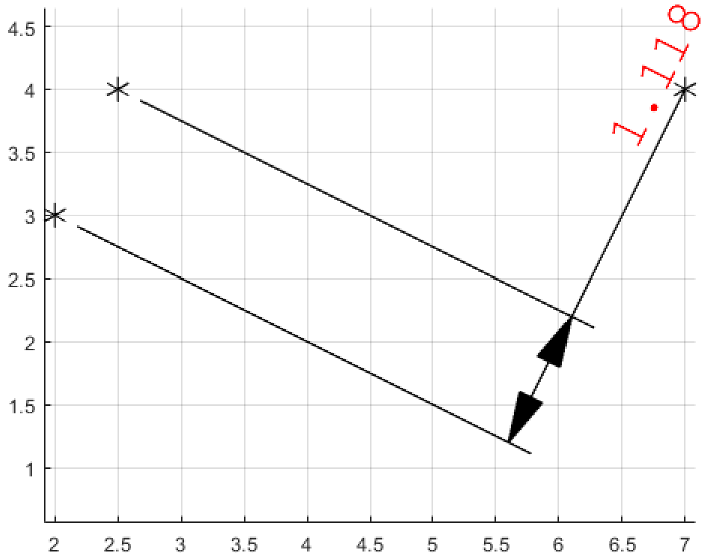
```
drawInit
ad1 = 0.4; ad2 = ad1/2;
```

```
x1 = 2; y1 = 4;
x2 = 5; y2 = 6;
xt = 3; yt = 6;
gkSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawPoint(1,ad1/2,x2,y2)
drawPoint(1,ad1/2,xt,yt)
drawDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
```



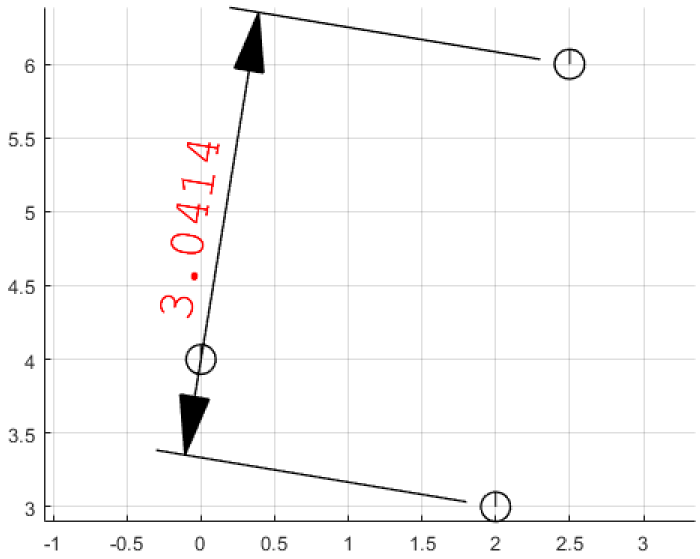
## Example 3

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 2.5; y2 = 4;
xt = 7; yt = 4;
drawSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawPoint(1,ad1/2,x2,y2)
drawPoint(1,ad1/2,xt,yt)
drawDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
grid on
```



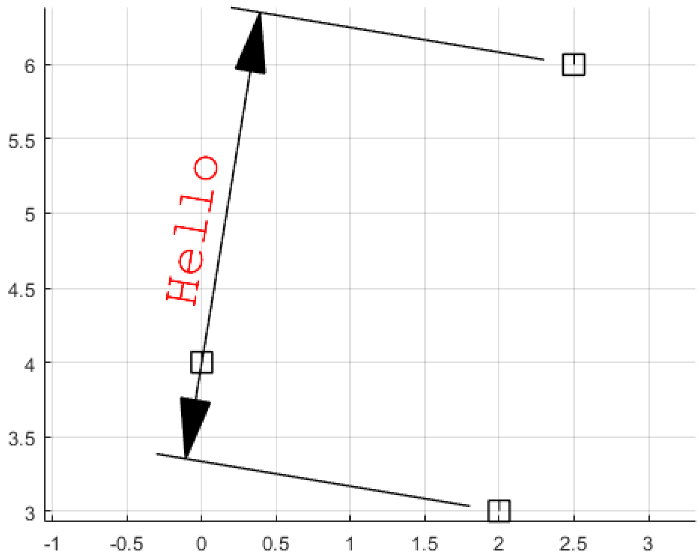
## Example 4

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 2.5; y2 = 6;
xt = 0; yt = 4;
drawSet('FontSize',26)
drawPoint(3,ad2,x1,y1)
drawPoint(3,ad2,x2,y2)
drawPoint(3,ad2,xt,yt)
drawDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
grid on
```



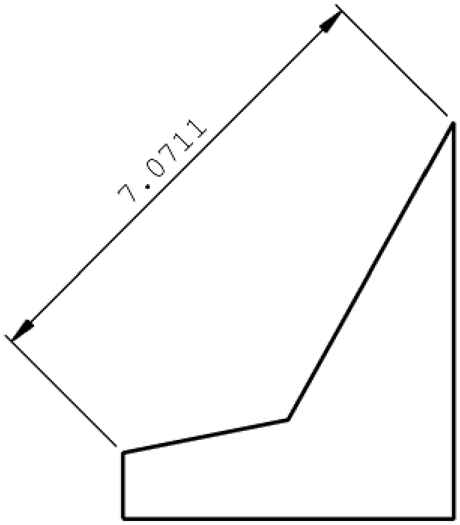
## Example 5

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 2.5; y2 = 6;
xt = 0; yt = 4;
drawSet('FontSize',26,'textColor','r')
drawPoint(2,ad1/2,x1,y1)
drawPoint(2,ad1/2,x2,y2)
drawPoint(2,ad1/2,xt,yt)
drawDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt,'-str','Hello')
grid on
```



## Example 6

```
drawInit
ad1 = 0.4; ad2 = ad1/3;
a = 5; b = 1; c = 3;
p = drawPolygon([0 a a a/2 0],[0 0 2*c c/2 b],'LineWidth',2,'Color','k');
drawSet('FontSize',16,'textColor','k')
drawDim(3,ad1,ad2,p.xk(3),p.yk(3),p.xk(5),p.yk(5),-a/8,1.25*c)
axis off
```



```
%grid on
```

## See also

## References

# drawDonut

# fillDonut

Draw or fill donut i.e, two concentric circles with given diameter.

## Description

Donut is IGES flash entity (type 125),  form number 3 (see [1], pp 120-123).

## Syntax

drawDonut(d1,d2,xc,yc)

drawDonut(d1,d2,xc,yc,LineSpec)

drawDonut(d1,d2,xc,yc,'-np',np)

p = drawDonut(__)

fillDonut(c1,c2,__)

## Description

drawDonut((d1,d2,xc,yc) draw concentric circles wit diameters d1 and d2 and center in point (xc,yc).

drawDonut(wd,ht,LineSpec)  sets the line style, line width, and color.

p = drawDonut(__) returns structure with output data.

## Method

drawDonut call the function **evalCircle**.

The curve is plotted by MATLAB function plot.

## Arguments

### Input Arguments

**c1, c2** - fill color

**d1** - diameter of outer circle

**d2** - diameter of inner circle

**xc** - circle center

**yc** -

### Optional Name-Value Pair Input Arguments

**'-np',np** - number of points on each circle

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

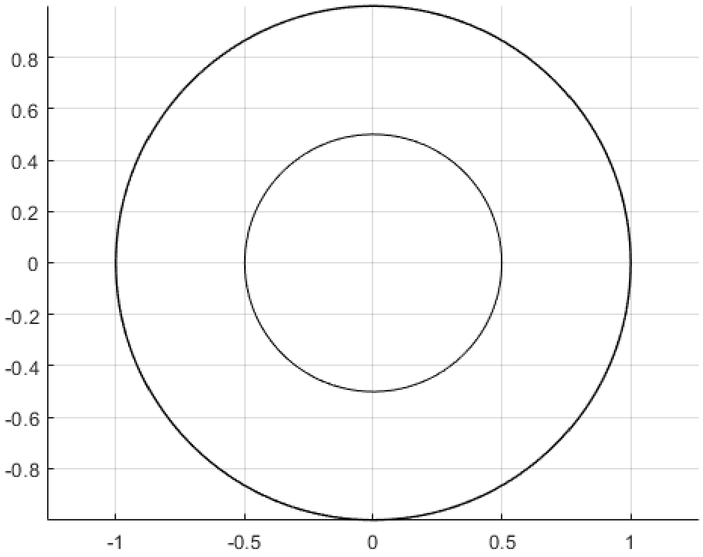
**p** - structure  with fields

- p.style - line style
- p.width - line width
- p.color - line color

## Examples

### Example 1

```
figure
hold on
axis equal
% close polygon
drawDonut( 2, 1,0,0);
grid on
```



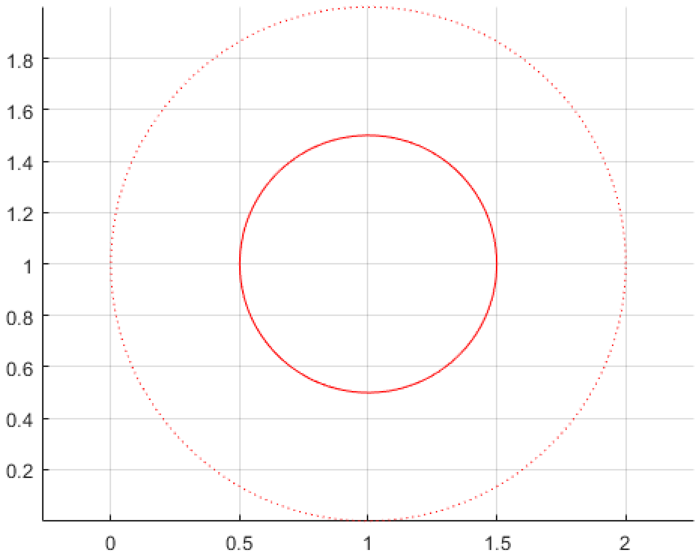
### Example 2

```
figure
axis equal
hold on
```

```
p=drawDonut(2,1,1,1,'r:')
```

```
p = struct with fields:
       xk: [0 2 1 1 1]
       yk: [1 1 0 2 1]
    color: [1 0 0]
```

```
grid on
```

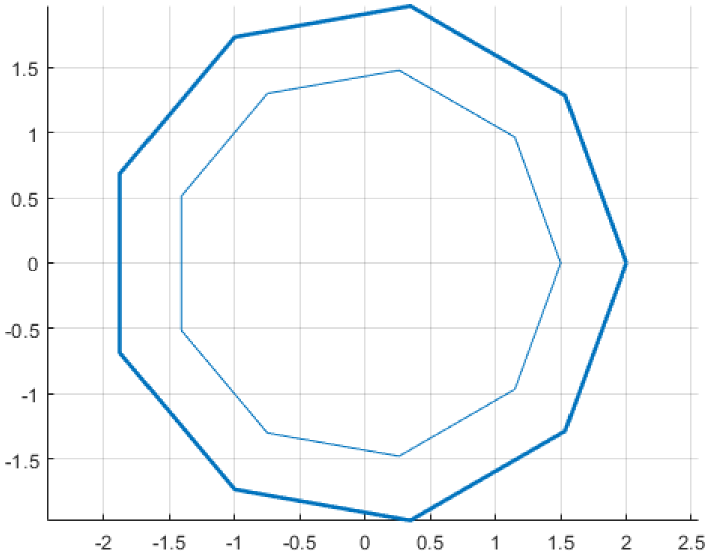


### Example 3

```
figure
axis equal
hold on
p=drawDonut(4,3,0,0,'LineWidth',2,'-np',10)
```

```
p = struct with fields:
       xk: [-2 2 0 0 0]
       yk: [0 0 -2 2 0]
    color: [0 0.4470 0.7410]
```

```
grid on
```

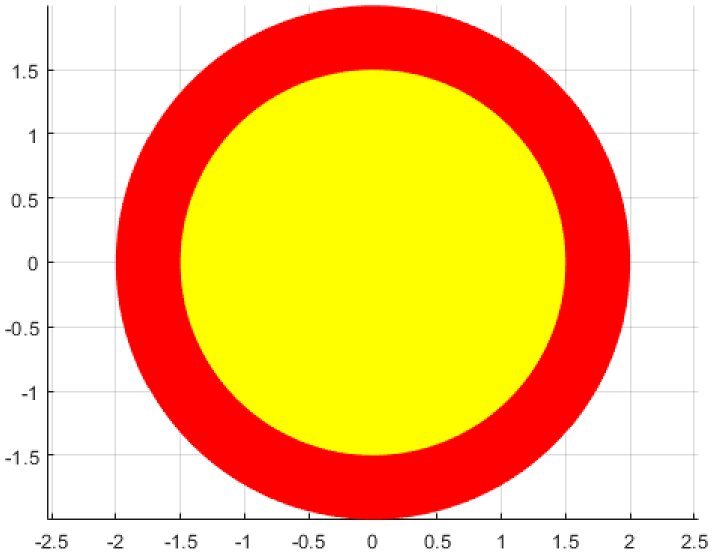


### Example 3

```
figure
axis equal
hold on
p=fillDonut('r','y',4,3,0,0)
```

```
p = struct with fields:
    xk: [-2 2 0 0]
    yk: [0 0 -2 2]
```

```
grid on
```



## See also

## References

[1] IGES Initial Graphics Exchange Specification IGES 5.3

# drawEllipse

# fillEllipse

Draw the Ellipse arc

## Description

Parametric equation of an elipse is

$$x = x_c + a\cos t,\ y = y_c + b\sin t, \quad 0 \le t < 2\pi\,.$$

where *a, b* are the ellipse semi axes, and *t* is a sweep angle.

## Syntax

drawEllipse( a, b, xc, yc)

fillEllipse( c, a, b, xc, yc)

drawEllipse( a, b, xc, yc, rot)

drawEllipse( a, b, xc, yc, rot, t1, t2)

drawEllipse( a, b, xc, yc, rot, t1, t2,'-pie'|'-seg')

drawEllipse(__, '-np', np)

drawEllipse(__, LineSpec)

p = drawEllipse(__)

## Description

drawEllipse( a, b,xc,yc) draw the elipse with major axis a , minor axis b with enter at (xc,yc).

drawEllipse( a, b, xc, yc, rot) rotated by the angle *rot* in CCLW direction around center.

drawEllipse( a, b, xc, yc, rot, t1, t2) draw elliptic arc

drawEllipse( a, b, xc, yc, rot, t1, t2,'-pie'|'-seg') draw elliptic pie or segment

drawEllipse( __, LineSpec)  sets the line style, line width, and color.

drawEllipse( __,'-np',np) set the number of points on the output curve.

p = drawEllipse(__) returns structure with output data.

## Method

For calculation of the coordinates of the curve **drawEllipse** call function **evalEllipse( a,b,xc,yc,rot,t)** which returns coordinates *x* and *y* of the curve at given parameters t . For usage of **evalEllipse** see Example 1.

The curve is plotted by MATLAB function plot.

# Arguments

## Input Arguments

**c -** fill color

**a, b** - semi major axis (real scalar)

**xc -** x-coordinate of the center (real scalar)

**yc** - y-coordinate of the center (real scalar)

## Optional Input Argumnts

**rot** - rotation angle about the center in degrees, CCLW direction >0, CLW direction is < 0.

**t1** - start sweep angle in degrees)

**t2** - central sweep angle in degrees)

**'-pie'|'-seg'** - draw pie or segment

## Optional Name-Value Pair Input Arguments

**'-np', np** - number of points along the curve ( scalar integer value > 2)

***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

**p** - structure with the fields

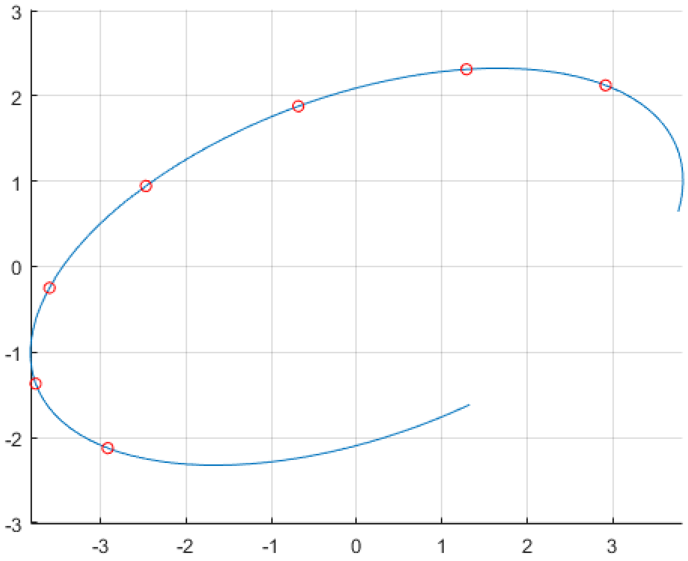
- p.xk, p.yk - key points: 1=start,2=end,3=center
- p.th - tangent angle: 1=start,2=end
- p.color - line color

# Examples

## Example 1

```
%Data
a = 4;
b = 2;
% plot
figure
hold on
```

```
axis equal
[x,y] = evalEllipse(a,b,0,0,20,linspace(-20,280));
plot(x,y)
[x,y] = evalEllipse(a,b,0,0,20,[30,60,90,120,150,180,210]);
scatter(x,y,30,'r')
grid on
```



## Example 2

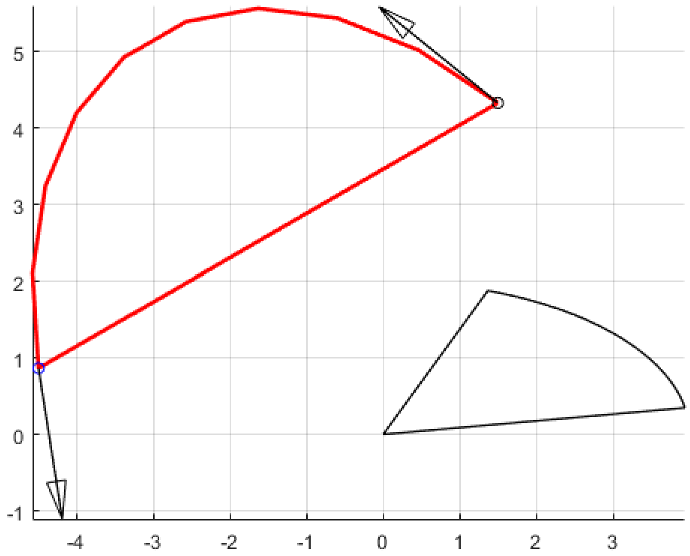
```
%Data
xc = 0;
yc = 0;
a = 4;
b = 2;
t1 = 30;
t2 = 120;
figure
hold on
axis equal
E1 = drawEllipse(a,b,0,0,0,10,60,'-pie')
```

```
E1 = struct with fields:
        xk: [3.9392 1.3681 0]
        yk: [0.3473 1.8794 0]
        th: [109.4254 169.6859]
     color: 'k'
```

```
E2 = drawEllipse(a,b*3,xc,yc,30,t1,t2,'-seg','LineWidth',2,'Color','r','-np',10)
```

```
E2 = struct with fields:
       xk: [1.5000 -4.5000 0]
       yk: [4.3301 0.8660 0]
       th: [141.0517 -81.0517]
    color: [1 0 0]
```

```
scatter(E2.xk(1),E2.yk(1),30,'k')
scatter(E2.xk(2),E2.yk(2),30,'b')
drawArrow(2,0.5,0.25,E2.xk(1),E2.yk(1),'-rtheta',2,E2.th(1))
drawArrow(2,0.5,0.25,E2.xk(2),E2.yk(2),'-rtheta',2,E2.th(2))
grid on
```

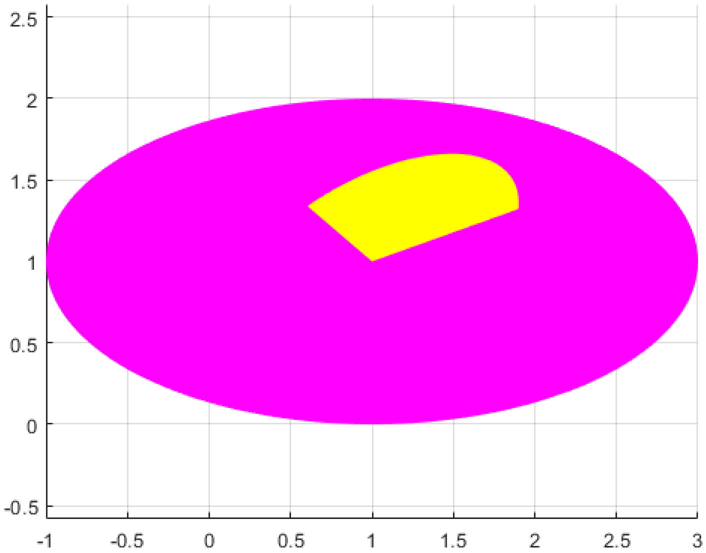


## Example 3

```
figure
hold on
axis equal
p = fillEllipse('m',2,1,1,1,'-np',215);
p = fillEllipse('y',1,0.5,1,1,30,-20,120,'-sec','-np',215)
```

```
p = struct with fields:
    xk: [1.8993 0.6034 1]
    yk: [1.3217 1.3396 1]
    th: [83.9476 -144.9616]
```

```
grid on
```



## Example 4

```
% intercept errors
figure
hold on
axis equal
p = drawEllipse(2,1,0,0,20,'-np',15)
grid on
```

## See Also

## References

# drawForce

Draw a force

## Description

## Syntax

drawForce(F,th,d,xr,yr)

drawForce(F,th,d,xr,yr,rot)

drawForce(__,'-ad',ad)

drawSupport(__,LineSpec)

p = drawSpring(__)

## Description

## Method

## Arguments

### Input Arguments

**F,th** -- force and its local inclination angle in degrees

**d** -- disatnce from reference point

**xr, yr** -- reference point

**Optional input:**

**rot** -- rotation of distance about reference point in degrees

### Optional Name-Value Pair Input Arguments

**'-ad',ad** - arowhead width (default is 0.2*F). Height is set to width/2. Arrowhead form is 3 (filled arrow)

***LineSpec*** - specifies line properties, see Line Properties.

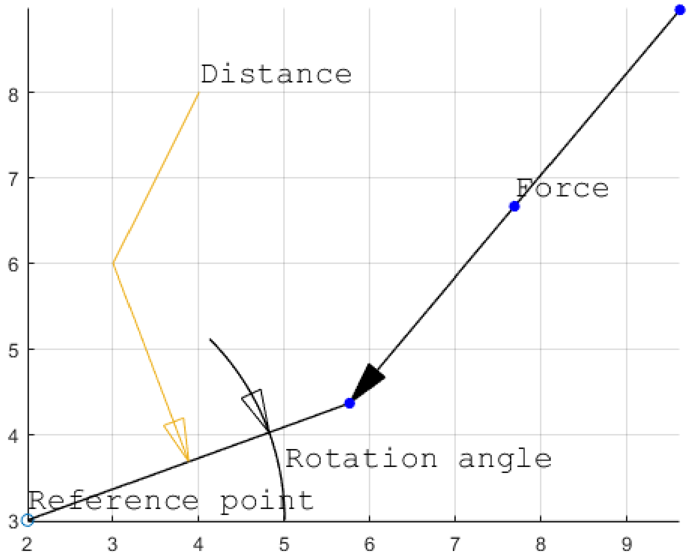
### Optional Output Arguments

**p** - structure with key points

# Examples

## Example 1

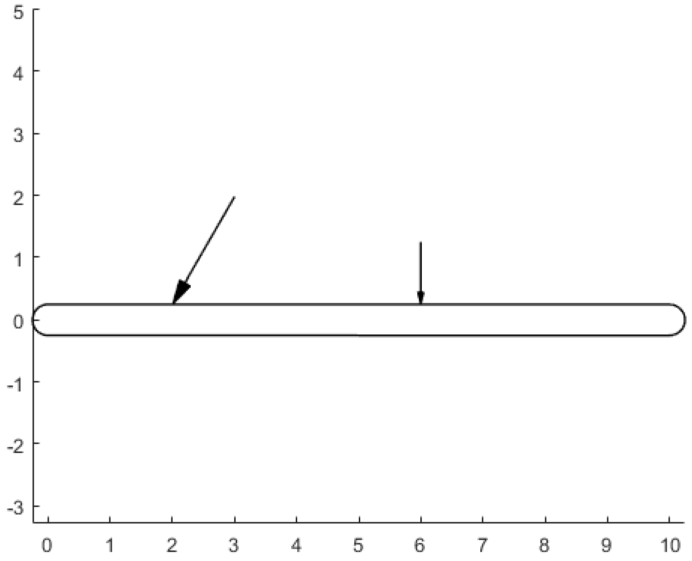
```
drawInit
gkInit
xr = 2;yr = 3;d = 4; th = 20;
scatter(xr,yr)
drawText(xr,yr,'Reference point')
L=drawLine(xr,yr,'-rtheta',d,th);
L1=drawLeader(2,0.5,0.25,(L.xk(1)+L.xk(2))/2,(L.yk(1)+L.yk(2))/2,[3,4],[6,8]);
drawText(L1.xk(end),L1.yk(end),'Distance')
% draw force inclinaded by 30 degrees to direction
p = drawForce(6,30,d,xr,yr,th,'-ad',0.5);
scatter(p.xk,p.yk,30,'b','filled')
% angular dimension
drawCircle(xr,yr,3,0,th,'-np',40)
drawArcArrow(2,0.5,0.25,xr,yr,3,45,-45+th)
drawText(5,3.5,'Rotation angle')
drawText(p.xk(3),p.yk(3),'Force')
grid on
```



```
%axis off
gkClose
```

## Example 2

```
drawInit
gkInit
c=drawCanoe1(0.5,0,0,10,0);
drawForce(2,60,2,c.xk(7),c.yk(7))
drawForce(1,90,6,c.xk(7),c.yk(7))
```



## See Also

## References

# drawGet

Get attribute of drawing entities

## Description

## Syntax

val = drawGet(str)

## Description

drawGet returns tthe value saved in global variable gkdata.  The variable (table) is initialized by calling drawInit or gkInit.

## Method

## Arguments

### Input Arguments

**str** -- name of propery (case insensitive)

for plot: 'linestyle', 'linecolor', 'linewidth'

for fill: 'facecolor', 'edgecolor', 'edgewidth', 'edgestyle'

for text:  'fontname', 'fontsize', 'fontweight', 'textcolor', 'horizontalalignment', 'verticalalignment', 'rotation'

### Output Arguments

**val** - value of  property

## Examples

### Example 1

```
gkInit
drawGet('lineColor')
```

```
ans = 'k'
```

```
drawGet('rotation')
```

```
ans = 0
```

```
gkClose
```

## Example 2

```

%drawInit
drawGet('lineColor')
```

```
Error using gkGet (line 5)
Data table is empty.

Error in drawGet (line 6)
    value = gkGet(name);
```

```
drawGet('rotation')
gkClose
```

## See Also

## References

# drawHDim

Draw horizontal dimension

## Description

Horizontal dimension measures  the horizontal distance between two points.

## Syntax

drawHDim(form,d1,d2,x1,y1,x2,y2,xt,yt)

drawHDim(__,'-str',str)

drawHDim(_,LineSpec)

### Description

drawHDim(form,d1,d2,x1,y1,x2,y2,xt,yt) draw horizontal dimension between points (x1,y1) and (x2,y2) and locate text, i.e. value of distnce x2-x1, at point (xt,yt). If x2 < x1 than the points are swaped.

drawHDim(__,'-str',str) draw horizontal dimension between points (x1,y1) and (x2,y2) and locate text given by variable str at point (xt,yt).

drawHDim(__,LineSpec)  sets the line style, line width, and color.

### Method

drawHDim use drawLine, drawArrohwhead, and gkText to draw a horizontal dimension . For drawing text the function drawHDim use current text attrubtes. They can be changed by function drawSet.

## Arguments

### Input Arguments

**form** - arrowhead form

**d1,d2** - arrow head width and height

**x1,y1** - start point. x1 colud be > x2

**x2,y2** - end point

**xt,yt**    - text location. If xt is inside interval (x1,x2) then text is located at the center of the interval

### Optional Name-Value Pair Input Arguments

**'-str'|'-txt',str** - dimension text

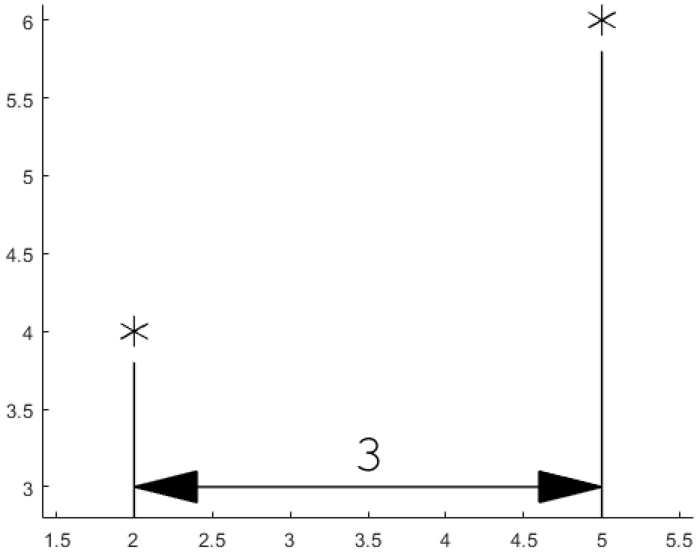
***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

## Examples

### Example 1

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 4;
x2 = 5; y2 = 6;
xt = 3; yt = 3;
gkSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawPoint(1,ad1/2,x2,y2)
drawHDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
```

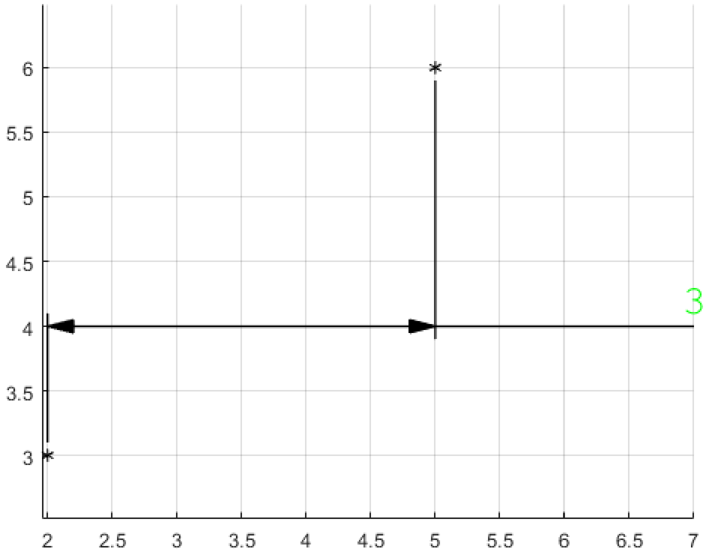


```
%grid on
```

### Example 2

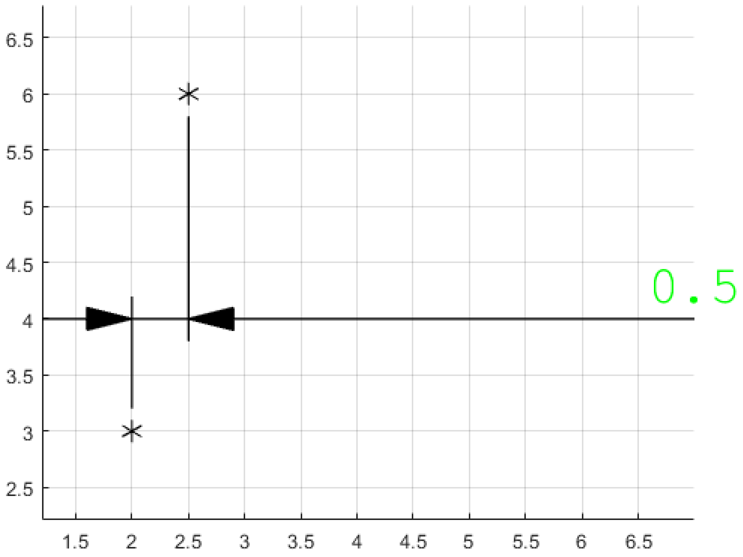
```
drawInit
```

```
ad1 = 0.2; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 5; y2 = 6;
xt = 7; yt = 4;
drawSet('FontSize',20,'textColor','g')
drawPoint(1,ad1/2,x1,y1)
drawPoint(1,ad1/2,x2,y2)
drawHDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
grid on
```



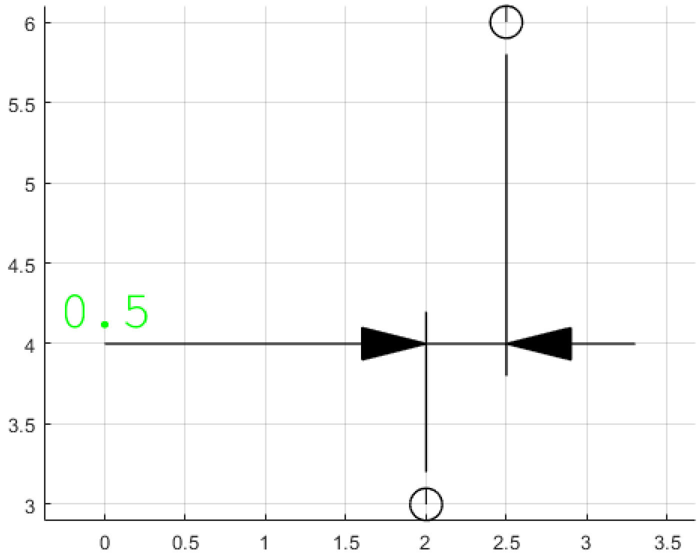
## Example 3

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 2.5; y2 = 6;
xt = 7; yt = 4;
drawSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawPoint(1,ad1/2,x2,y2)
drawHDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
grid on
```



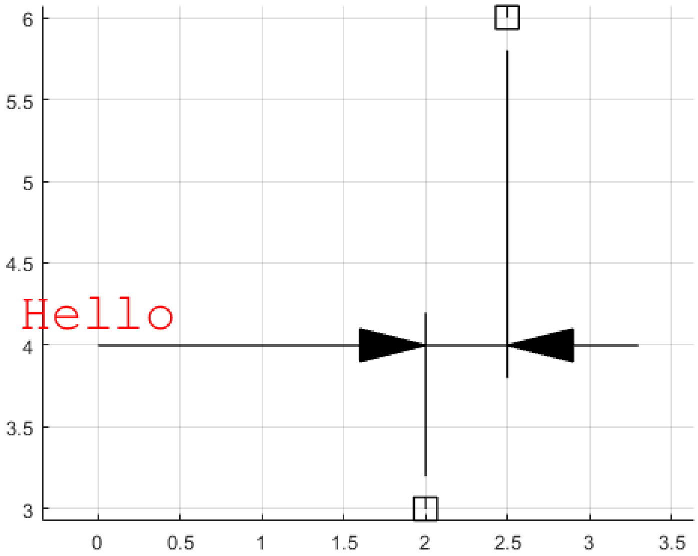
## Example 4

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 2.5; y2 = 6;
xt = 0; yt = 4;
drawSet('FontSize',26)
drawPoint(3,ad2,x1,y1)
drawPoint(3,ad2,x2,y2)
drawHDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
grid on
```



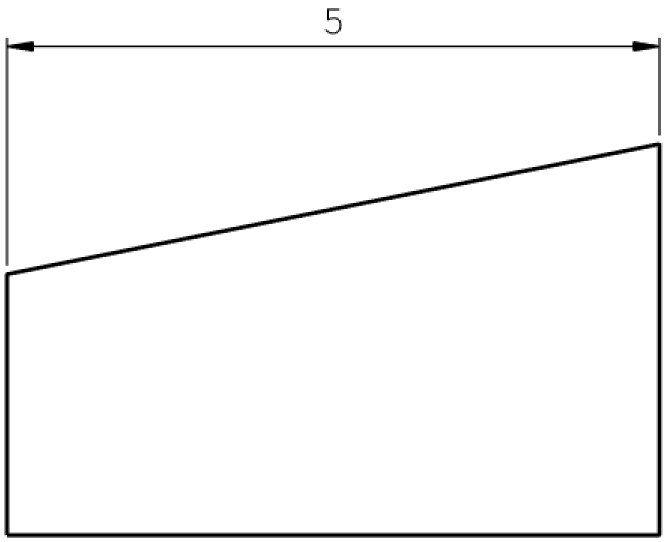
## Example 5

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 2.5; y2 = 6;
xt = 0; yt = 4;
drawSet('FontSize',26,'textColor','r')
drawPoint(2,ad1/2,x1,y1)
drawPoint(2,ad1/2,x2,y2)
drawHDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt,'-str','Hello')
grid on
```



### Example 6

```
drawInit
ad1 = 0.2; ad2 = ad1/3;
a = 5; b = 2; c = 3;
p = drawPolygon([0 a a 0],[0 0 c b],'LineWidth',2,'Color','k');
drawSet('FontSize',20,'textColor','k')
drawHDim(3,ad1,ad2,p.xk(3),p.yk(3),p.xk(4),p.yk(4),a/2,1.25*c)
axis off
```



```
%grid on
```

## See also

## References

# drawHyperbola

Draw the arc of hyperbola

## Description

Parametric equation of the hyperbola is

$$x = a\cosh t, \qquad y = b\sinh t, \qquad -\infty < t < \infty.$$

where a*,* b *are semi axes*.

## Syntax

drawHyperbola( t1, t2, a, b, xc, yc)

drawHyperbola( t1, t2, a, b, xc, yc, rot)

drawHyperbola( t1, t2, a, b, '-np', np)

drawHyperbola( t1, t2, a, b, LineSpec)

p = drawHyperbola(__)

## Description

drawHyperbola( t1, t2, a, b,xc,yc) draw the arc with default number of points 360 and current line specification.

drawHyperbola( t1, t2, a, b, xc, yc, rot) rotated by the angle *rot* in CCLW direction around center.

drawHyperbola( __, LineSpec)  sets the line style, line width, and color.

drawHyperbola( __,'-np',np) set the number of points on the output curve.

p = drawHyperbola(__) returns structure with output data.

## Method

For calculation of the coordinates of the curve **drawHyperbola** call function **evalHyperbola(a, b,xc,yc,rot,t)**  which returns coordinates *x* and *y* of the curve at given parameters t . For usage of **evalHyperbola** see Example 1.

The curve is plotted by MATLAB function plot.

## Arguments

### Input Arguments

**t1** - initial parameter

**t2** - final parameter

**a** - focal distance (real scalar)

**b** - focal distance (real scalar)

**xc -** x-coordinate of the center (real scalar)

**yc** - y-coordinate of the center (real scalar)

### Optional Input Argumnts

**rot** - rotation angle about the center in degrees

**'-pie'|'-seg'** - draw pie or segment

### Optional Name-Value Pair Input Arguments

**'-np', np** - number of points along the curve ( scalar integer value > 2)

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

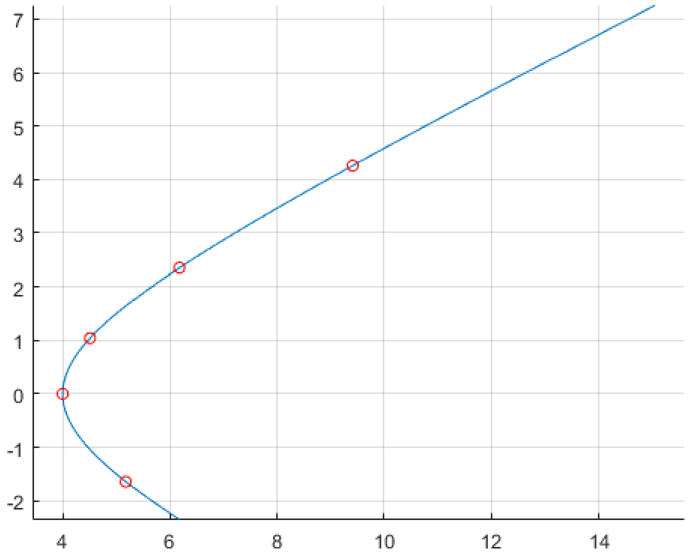
**p** - structure with the fields

- p.xk, p.yk - key points: 1=start,2=end,3=center
- p.th - tangent angle: 1=start,2=end
- p.color - line color

## Examples

### Example 1

```
%Data
a = 4;
b = 2;
% plot
figure
hold on
axis equal
[x,y] = evalHyperbola(a, b,0,0,0,linspace(-1,2));
plot(x,y)
[x,y] = evalHyperbola(a, b,0,0,0,[-0.75, 0, 0.5,1, 1.5]);
scatter(x,y,30,'r')
grid on
```



## Example 2

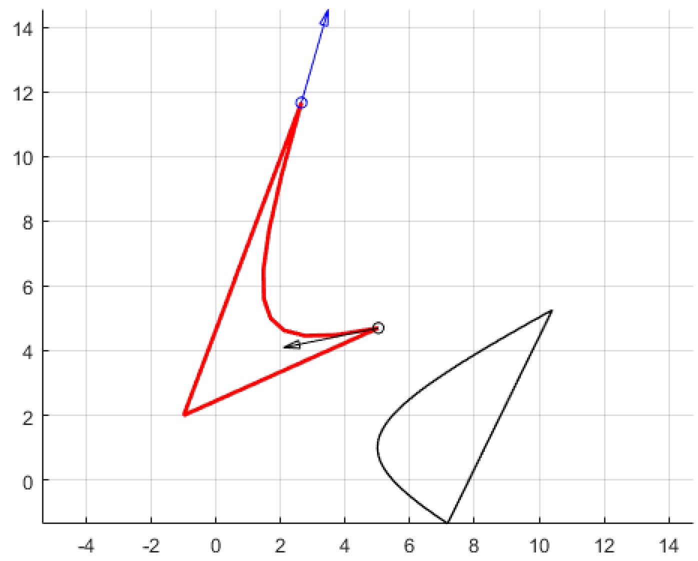
```
%Data
xc = -1;
yc = 2;
a = 4;
b = 2;
t1 = -1;
t2 = 1.5;
figure
hold on
axis equal
H1 = drawHyperbola(t1,t2,a,b,1,1,'-seg')
```

```
H1 = struct with fields:
       xk: [7.1723 10.4096 1]
       yk: [-1.3504 5.2586 1]
       th: [146.7144 28.9161]
    color: 'k'
```

```
H2 = drawHyperbola(t1,t2,a,b,xc,yc,45,'-sec','LineWidth',2,'Color','r','-
np',10)
```

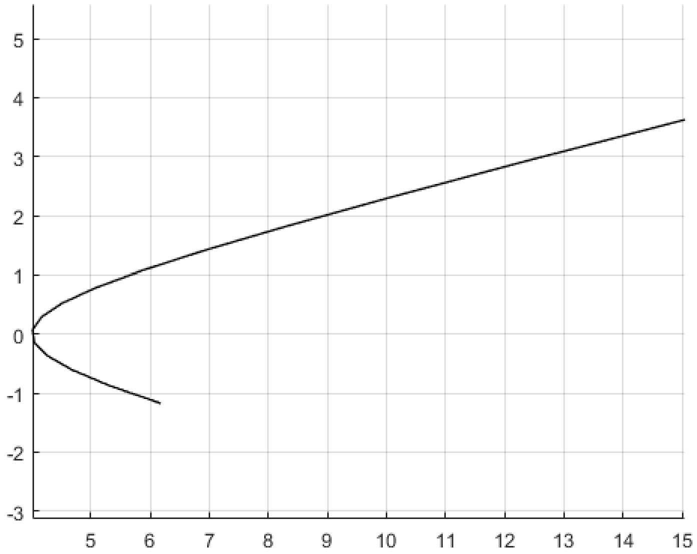
```
H2 = struct with fields:
       xk: [5.0265 2.6424 -1]
       yk: [4.7025 11.6649 2]
       th: [191.7144 73.9161]
    color: [1 0 0]
```

```
scatter(H2.xk(1),H2.yk(1),30,'k')
scatter(H2.xk(2),H2.yk(2),30,'b')
drawArrow(2,0.5,0.25,H2.xk(1),H2.yk(1),'-rtheta',3,H2.th(1),'k')
drawArrow(2,0.5,0.25,H2.xk(2),H2.yk(2),'-rtheta',3,H2.th(2),'b')
grid on
```



## Example 3

```
figure
hold on
axis equal
drawHyperbola(2,-1,4,1,0,0,'-np',15)
grid on
```



## See Also

## References

# drawInit

Initialize figure.

## Description

The function calls:

clf % clear current figure

hold on % retains plots in the current axes

axis equal % use the same length for the data units along each axis

This function is not essential for the rest of the functions from draw2d library, however if one omit its call than setting **axis equal** is essentila in ordet to obtain correct shapes.

## Syntax

drawInit

drawInit( figNum)

## Description

drawInit - creates new figure

drawInit( figNum) - initialize figure 'figNum'. Use this also to clear the figure.

p = drawInit(__) - figure number

## Method

## Arguments

### Optional Input Arguments

**figNum** - figure number (>0)

### Optional Output Arguments

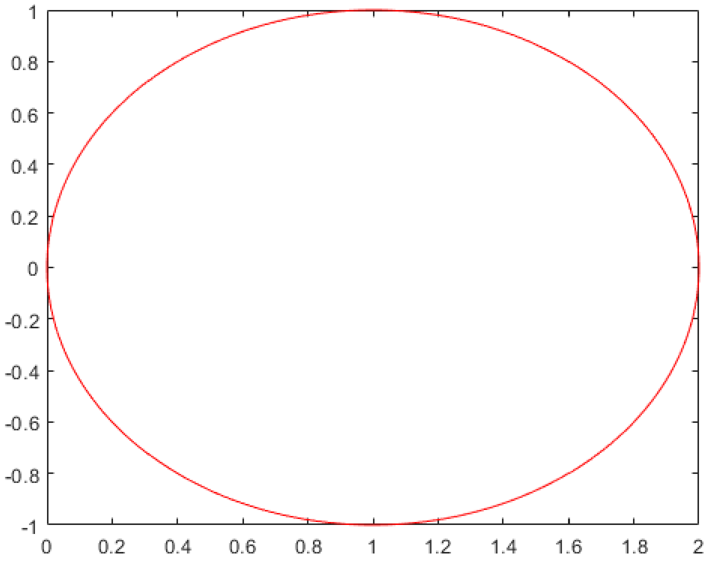
**p** - figure number.

## Examples

### Example 1

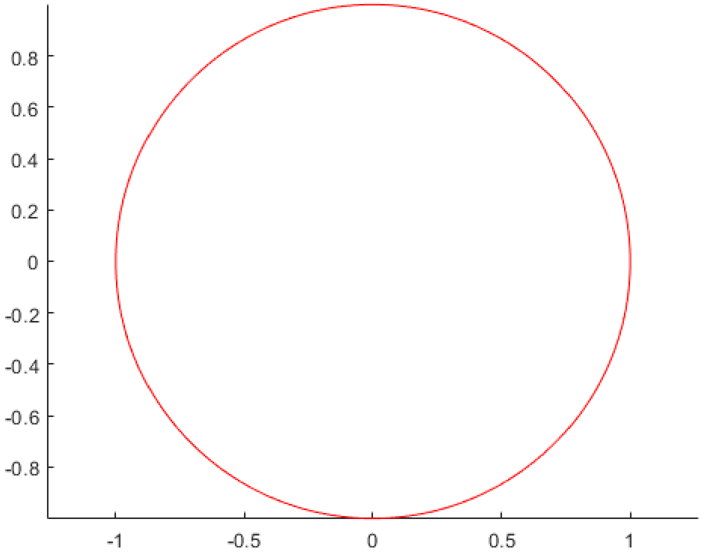
```
figure
clf
```

```
drawCircle(1,0,1,'r')  % circle is distorted
```



## Example 2

```
drawInit
drawCircle(0,0,1,'r')
```



## See Also

## References

[1] IGES Initial Graphics Exchange Specification IGES 5.3

# drawLeader

Draw one or more lines where the first line begin with arrowhead.

## Description

See IGES 5.3,  4.62 Leader (arrow) entity, pp 259-251 ([1])

## Syntax

drawLeader( form, ad1, ad2, xh, yh, xt, yt)

drawLeader( form, ad1, ad2, xh, yh, xt, yt, LineSpec)

## Description

drawLeader( form, ad1, ad2, xh, yh, xt, yt) draw leader with arrow head type *form* and arrowhead point at *xh*,*yh*.

drawLeader( form, ad1, ad2, xh, yh, xt, yt, LineSpec) set line specification for lines. Color of arrowhead is the same as color of lines.

## Method

The arrowhead is plotted by function **drawArrowhead**.

The leader is plotted by MATLAB function plot.

## Arguments

### Input Arguments

**form** - arrowhead type number: 1,...,12 ([1]).

Form Meaning

1. Wedge
2. Triangle
3. Filled Triangle
4. No Arrowhead
5. Circle
6. Filled Circle
7. Rectangle
8. Filled Rectangle
9. Slash
10. Integral Sign

11. Open Triangle
12. Dimension Origin

**ad1** - arrowhead height (real scalar >0)

**ad2** - arrowhead width (real scalar >0)

**xh, yh** - arrowhead coordinates (real scalar)

**xt, yt** - tail coordinates (real vector)

### Optional Name-Value Pair Input Arguments

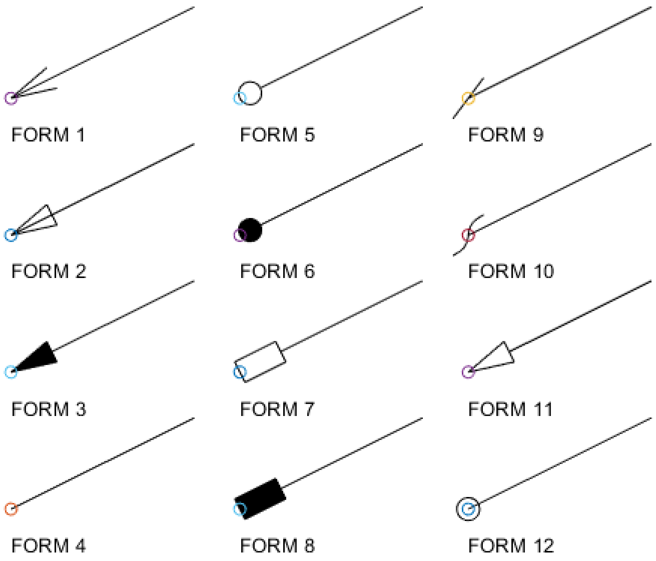
***LineSpec*** - specifies line properties, see Line Properties.

## Examples

### Example 1

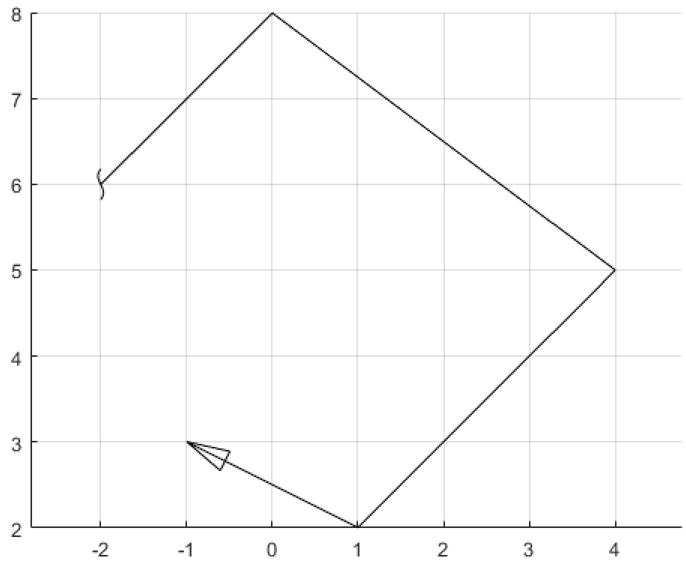
```
%Arrowhead types (IGES 5.3)
figure
hold on
axis equal
dx = 2.5;
dy = 1.5;
k = 0;
xh = 0;
yh = 3*dy;
ad2 = 0.5/2;
for n = 1:12
    yh = yh - dy;
    if n == 5 || n == 6 || n == 12
        ad1 = ad2;
    else
        ad1 = 2*ad2;
    end
    % x,y is arrow head point
    drawLeader(n,ad1,ad2,xh,yh,xh+2,yh+1)
    scatter(xh,yh)
    text(xh,yh-dy/4,sprintf('FORM %d',n))
    k = k + 1;
    if k > 3
        k = 0;
        xh = xh + dx;
        yh = 3*dy;
    end
end
```

```
axis off
```



## Example 2

```
%Arrowhead types (IGES 5.3)
figure
hold on
axis equal
ad1 = 0.5;
ad2 = ad1/2;
xt = [1 2 3 4 0];
yt = [2 3 4 5 8];
drawLeader( 2,ad1,ad2,-1,3,xt,yt,'k')
drawLeader(10,ad2,ad2,-2,6,xt(end),yt(end),'k')
grid on
```



## See Also

**drawArrow**

## References

[1] IGES Initial Graphics Exchange Specification IGES 5.3

# drawLimits

Set the clipping boundaries for the curent axes.

## Description

The function calls xlim and ylim to set clipping bounaries for current axes. The function does not check the current figure number.

## Syntax

drawLimits

drawLimits( xmin, xmax, ymin, ymax)

## Description

drawLimits - set limits to auto

drawLimits( xmin, xmax, ymin, ymax) - set axis limits

p = drawInit(__) - figure number

## Method

## Arguments

### Optional Input Arguments

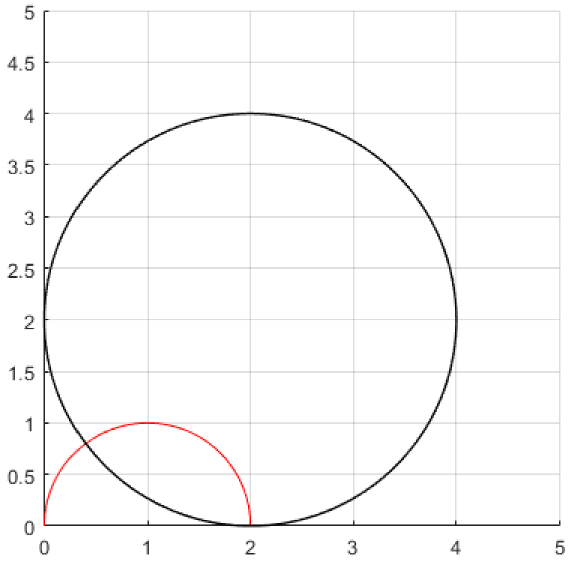
**xmin,xmax** - the minimum and the maximum boundaries in x-direction (xmin < xmax)

**ymin,ymax** - the minimum and the maximum boundaries in y-direction (ymin < ymax)

## Examples

### Example 1

```
drawInit
drawCircle(1,0,1,'r')
drawCircle(2,2,2)
drawLimits(0,5,0,5)
grid on
```



## See Also

## References

[1] IGES Initial Graphics Exchange Specification IGES 5.3

# drawLine

Draw the line between two points

## Description

Parametric equations of the line are

$$x = x_1 + t(x_2 - x_1), \qquad y = y_1 + t(y_2 - y_1), \quad t_1 \le t \le t_2.$$

where *r* is radius.

## Syntax

drawLine( x1, y1, x2, y2)

drawLine( x1, y1, x2, y2, t1, t2)

drawLine( x1, y1, x2, y2,LineSpec)

drawLine( x1, y1, '-rtheta',r,th)

drawLine( x1, y1, '-delta',dx,dy)

p = drawLine(__)

## Description

drawLine( x1, y1, x2, y2) draw the line betwe two points using current line specification.

drawLine( x1, y1, x2, y2, t1, t2) draw the line betwe two points from point given by t1 to point given by t2

drawLine( x1, y1, x2, y2,LineSpec) sets the line style, line width, and color.

drawLine( x1, y1, '-polar',r,th) draw line from start point to point given by polar coordinates from start point.

drawLine( x1, y1, '-delta',dx,dy) draw line from start point to point given by shift vector from start point.

p = drawLine(__) returns structure with input and output parameters of the line.

## Method

For calculation of the coordinates of the line **drawLine** use **evalLine( x1,y1,x2,y2,t)** which returns coordinates *x* and *y* of the curve at given parameter t . For usage of **evalLine** see Example 2.

The curve is plotted by MATLAB function plot.

## Arguments

## Input Arguments

**x1, y1 -** start point (real scalar)

**x2, y2 -** end point (real scalar)

or instaed of x2,y2

**'-rtheta',r,th** - polar coordinates of end point from start point

or

**'-delta',dx,dy** - shift vector

## Optonal Input Arguments

**t1,t2** - start and end parameter (defaults are 0 and 1)

## Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

**p** - structure with the fields

- p.x1 - start point x-coordinate (real scalar)
- p.y1 - start point y-coordinate (real scalar)
- p.x2 - end point x-coordinate (real scalar)
- p.y2 - end point y-coordinate (real scalar)
- p.d  - distance between end points
- p.style - line style
- p.width - line width
- p.color - line color

# Examples

## Example 1

```
%Data
x1 = 1; y1 = 1;
x2 = 2; y2 = 2;
figure
hold on
axis equal
L1 = drawLine(x1,y1,x2,y2)
```

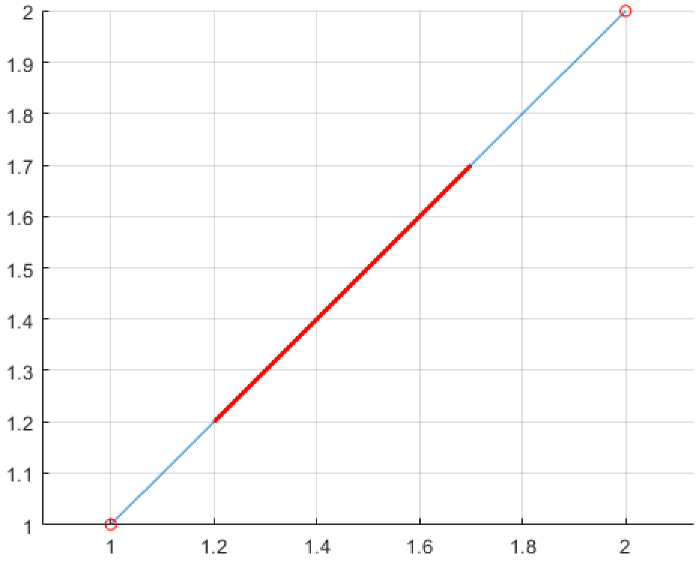
```
L1 = struct with fields:
        x1: 1
        y1: 1
        x2: 2
```

```
         y2: 2
          d: 1.4142
      style: '-'
      width: 0.5000
      color: [0 0.4470 0.7410]
```

```
scatter(x1,y1,30,'r')
scatter(x2,y2,30,'r')
L2 = drawLine(x1,y1,x2,y2,0.2,0.7,'LineWidth',2,'Color','r')
```

```
L2 = struct with fields:
         x1: 1.2000
         y1: 1.2000
         x2: 1.7000
         y2: 1.7000
          d: 0.7071
      style: '-'
      width: 2
      color: [1 0 0]
```

```
grid on
```



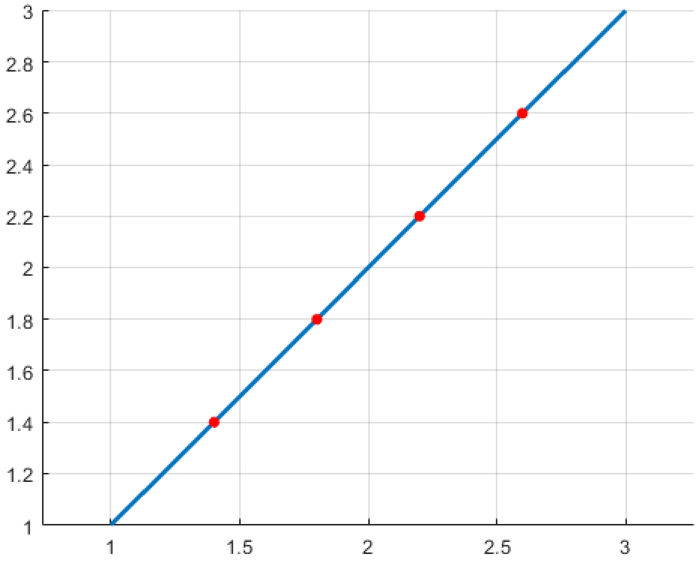
## Example 2

```
%Data
x1 = 1; y1 = 1;
dx = 2; dy = 2;
figure
```

```
hold on
axis equal
L1 = drawLine(x1,y1,'-delta',dx,dy,'LineWidth',2)
```

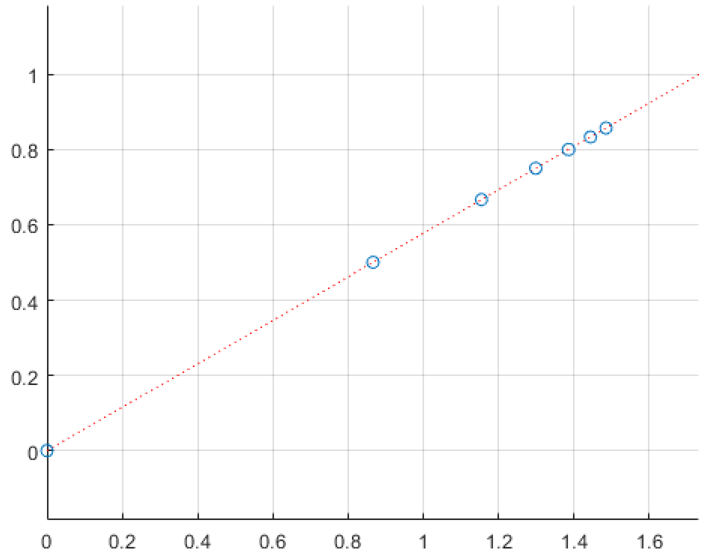
```
L1 = struct with fields:
       x1: 1
       y1: 1
       x2: 3
       y2: 3
        d: 2.8284
    style: '-'
    width: 2
    color: [0 0.4470 0.7410]
```

```
[x, y] = evalLine( L1.x1, L1.y1, L1.x2, L1.y2, [0.2,0.4,0.6,0.8]);
scatter(x,y,30,'r','filled')
grid on
```



## Example 3

```
figure
hold on
axis equal
L=drawLine(0,0,'-polar',2,30,'r:');
[x,y] = evalLine(L.x1,L.y1,L.x2,L.y2,[0,1/2,2/3,3/4,4/5,5/6,6/7]);
scatter(x,y)
grid on
```



## See Also

## References

# drawParabola

Draw the arc of paramola

## Description

Parametric equation of the parabola is

$$x = 4f\,t^2, \quad y = 2f\,t, \quad 0 \le t < \infty.$$

where *f is focal distance*are elispse semi axes.

## Syntax

drawParabola( t1, t2, f, xc, yc)

drawParabola( t1, t2, f, xc, yc, rot)

drawParabola( __, '-np', np)

drawParabola(__, LineSpec)

p = drawParabola(__)

## Description

drawParabola( t1, t2, f, xc, yc) draw the parabola with default number of points 360 and current line specification.

drawParabola( t1, t2,f, xc, yc, rot) rotated by the angle *rot* in CCLW direction around

drawParabola( __, LineSpec) sets the line style, line width, and color.

drawParabola( __,'-np',np) set the number of points on the output curve.

p = drawParabola(__) returns structure with output data.

## Method

For calculation of the coordinates of the curve drawParabola call function **evalParabola( f,xc,yc,rot,t)** which returns coordinates *x* and *y* of the curve at given parameters t . For usage of **evalParabola** see Example 1.

The curve is plotted by MATLAB function plot.

## Arguments

### Input Arguments

**t1** - initial parameter

**t2** - final parameter

**f** - focal distance (real scalar)

**xc -** x-coordinate of the center (real scalar)

**yc** - y-coordinate of the center (real scalar)

### Optional Input Argumnts

**rot** - rotation angle about the center in degrees

**'-seg'** - draw  segment

### Optional Name-Value Pair Input Arguments

**'-np', np** - number of points along the curve ( scalar integer value > 2)

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

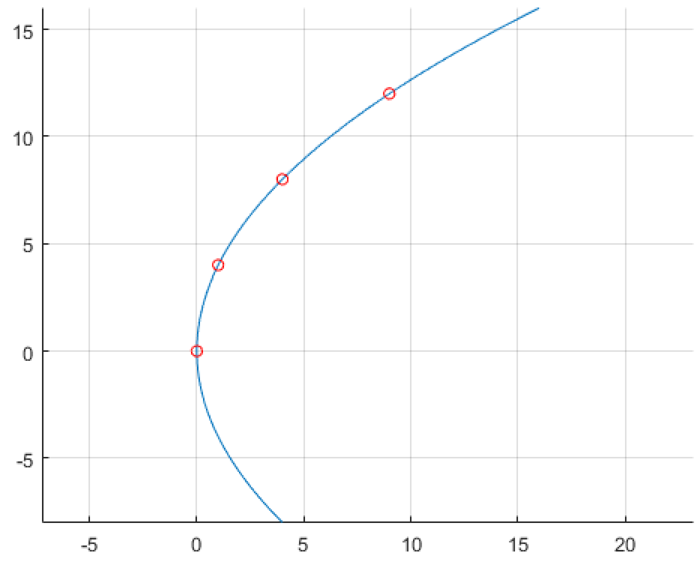
**p** - structure with the fields

- p.xk, p.yk - key points: 1=start,2=end,3=center
- p.th - tangent angle: 1=start,2=end
- p.color - line color

## Examples

### Example 1

```
%Data
f = 4;
% plot
figure
hold on
axis equal
[x,y] = evalParabola(f,0,0,0,linspace(-1,2));
plot(x,y)
[x,y] = evalParabola(f,0,0,0,[0, 0.5,1, 1.5]);
scatter(x,y,30,'r')
grid on
```



## Example 2

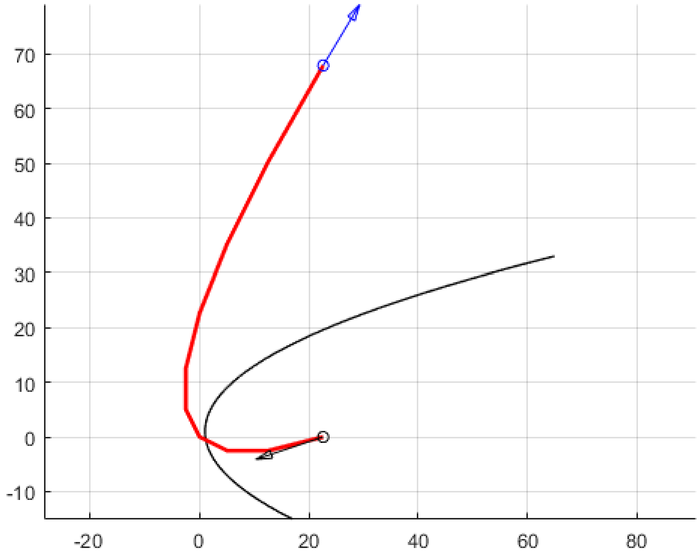
```
%Data
xc = 0;
yc = 0;
f = 4;
t1 = -2;
t2 = 4;
figure
hold on
axis equal
P1 = drawParabola(t1,t2,f,1,1)
```

```
P1 = struct with fields:
       xk: [17 65 1]
       yk: [-15 33 1]
       th: [153.4349 14.0362]
    color: 'k'
```

```
P2 = drawParabola(t1,t2,f,xc,yc,45,'LineWidth',2,'Color','r','-np',10)
```

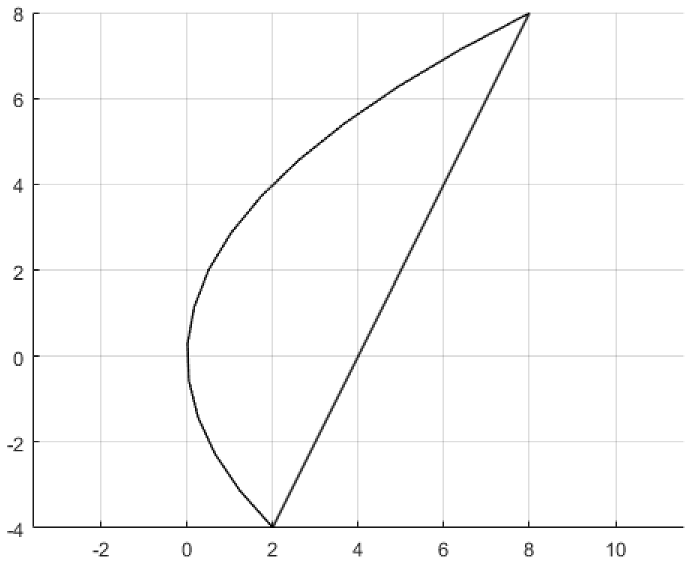
```
P2 = struct with fields:
       xk: [22.6274 22.6274 0]
       yk: [0 67.8823 0]
       th: [-161.5651 59.0362]
    color: [1 0 0]
```

```
scatter(P2.xk(1),P2.yk(1),30,'k')
scatter(P2.xk(2),P2.yk(2),30,'b')
drawArrow(2,3,1.5,P2.xk(1),P2.yk(1),'-rtheta',13,P2.th(1),'k')
drawArrow(2,3,1.5,P2.xk(2),P2.yk(2),'-rtheta',13,P2.th(2),'b')
grid on
```



## Example 3

```
figure
hold on
axis equal
drawParabola(2,-1,2,0,0,'-seg','-np',15)
grid on
```



## See Also

## References

# drawPoint

Draw apoint in plane.

## Description

## Syntax

drawPoint(form,d,,xp,yp)

drawPoint(__,LineSpec)

## Description

drawPoint(form,d,xp,yp) draw point.

p = drawPoint(__) returns some output data.

## Method

For calculation of the coordinates of the symbol representin the point  the function **evalNgon**.

The curve is plotted by MATLAB function plot.

## Arguments

## Input Arguments

**form** - form number; 1- star, 2-square

**d** - symbol dymension.

**xp -** x-coordinate of the point

**yp** - y-coordinate of the point

## Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

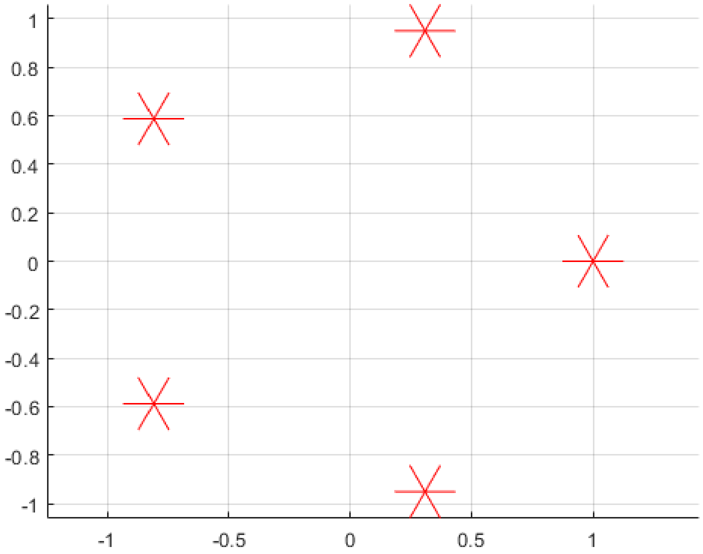
**p** - structure with fields

- p.xk,p.yk - point
- p.color - line color

## Examples

## Example 1

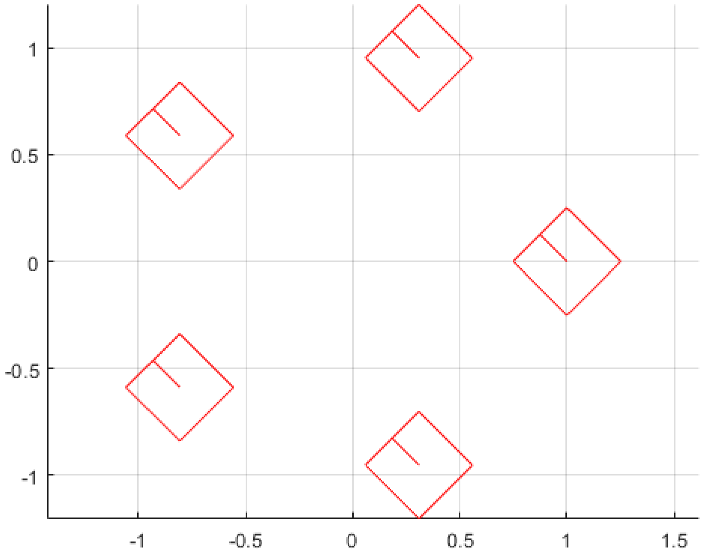
```
figure
hold on
axis equal
% Data
n = 5;
r = 1;
% close polygon
[xv,yv] = evalNgon( n, r, 0, 0, 0, 1, 1);
% plotvertices
for k = 1:n
    drawPoint(1,0.25, xv(k),yv(k),30,'r')
end
grid on
```



## Example 2

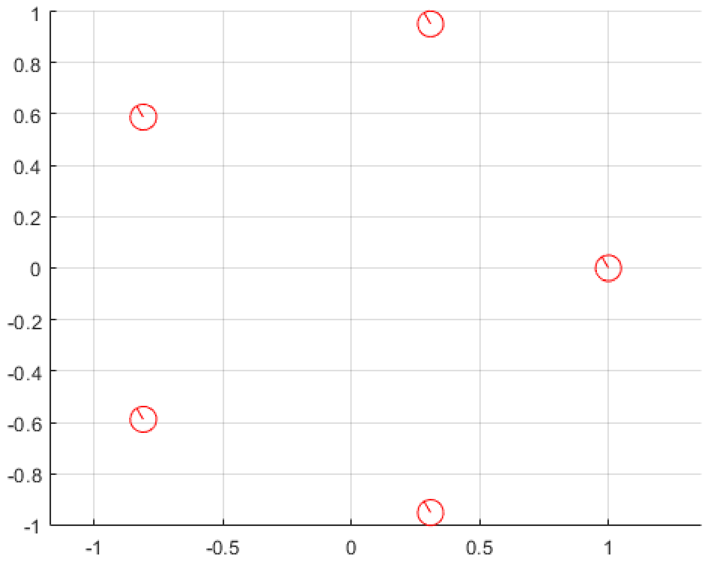
```
figure
hold on
axis equal
% Data
n = 5;
r = 1;
% close polygon
[xv,yv] = evalNgon( n, r, 0, 0, 0, 1, 1);
% plotvertices
for k = 1:n
    drawPoint(2,0.5, xv(k),yv(k),45,'r')
```

```
end
grid on
```



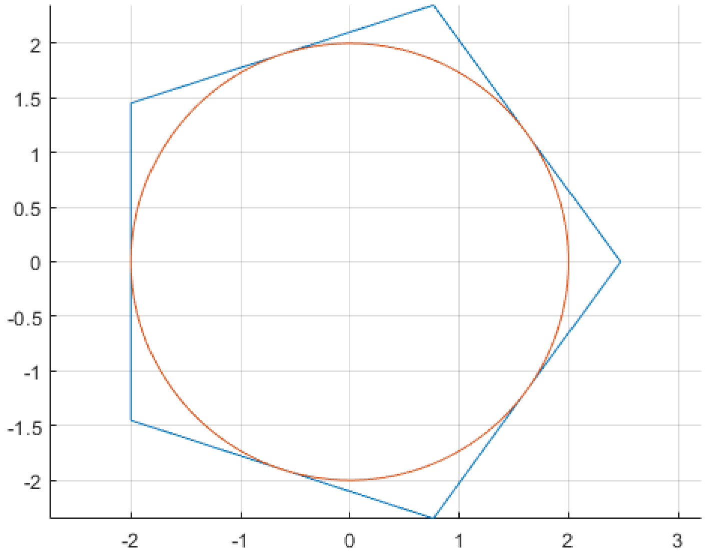
## Example 3

```
figure
hold on
axis equal
% Data
n = 5;
r = 1;
% close polygon
[xv,yv] = evalNgon( n, r, 0, 0, 0, 1, 1);
% plotvertices
for k = 1:n
    drawPoint(3,0.1, xv(k),yv(k),30,'r')
end
grid on
```



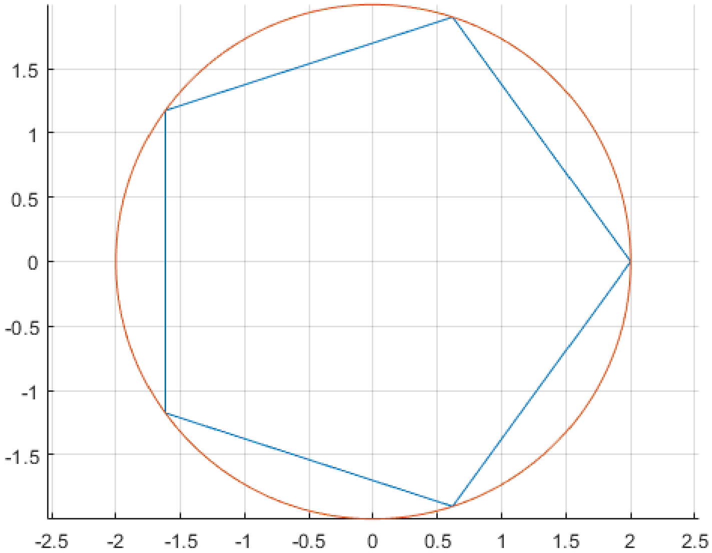
## Example 4

```
figure
axis equal
hold on
p=drawNgon(5,2,0,0,'-in');
drawCircle(p.xc,p.yc,2)
grid on
```



## Example 5

```
figure
axis equal
hold on
p=drawNgon(5,2,0,0,'-out');
drawCircle(p.xc,p.yc,2)
grid on
```



## See also

**drawCircle**

## References

# drawPolygon

# fillPolygon

Draw or fill a closed 2D shape with straight sides.

## Description

## Syntax

drawPolygon(xp,yp)

drawPolygon(xp,yp,LineSpec)

p = drawPolygon(__)

fillPolygon(c,__)

### Method

The curve is plotted by MATLAB function plot.

## Arguments

### Input Arguments

**c** - fill color

**xp -** x-coordinate of polygon vertices

**yp** - y-coordinate of polygon vertices

xp, yp must be of the same size.

### Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

**p** - structure .

- p.color - line color

## Examples

### Example 1

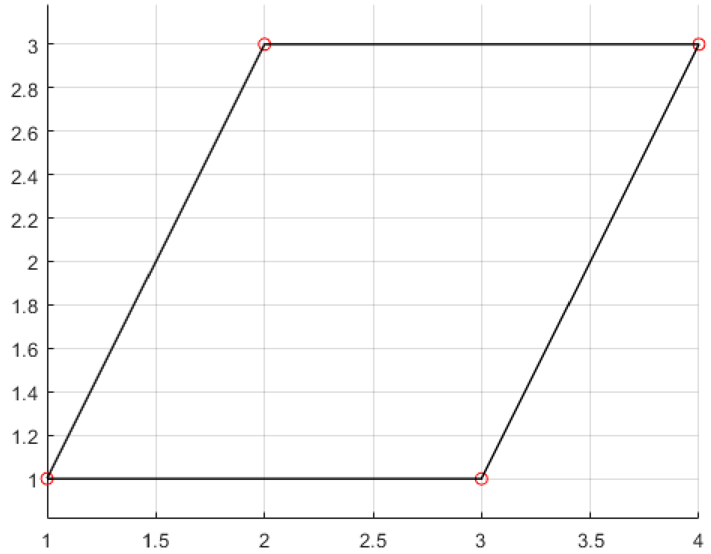
Example 5-3 from [1] (pp 143-144).

## Example 2

```
figure
hold on
axis equal
% coordinates of polygon vertices
xp=[1 2 4 3];
yp=[1 3 3 1];
% plot definiting polygon vertices
scatter(xp,yp,'r')
% plot definiting polygon
plot(xp,yp,'r')
% draw curve
p = drawPolygon(xp,yp)
```

```
p = struct with fields:
       xk: [1 2 4 3 1]
       yk: [1 3 3 1 1]
    color: 'k'
```

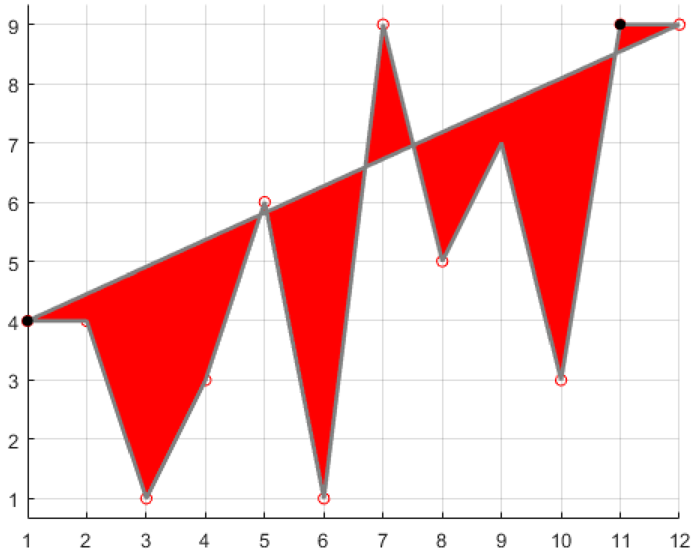
```
grid on
```



## Example 3

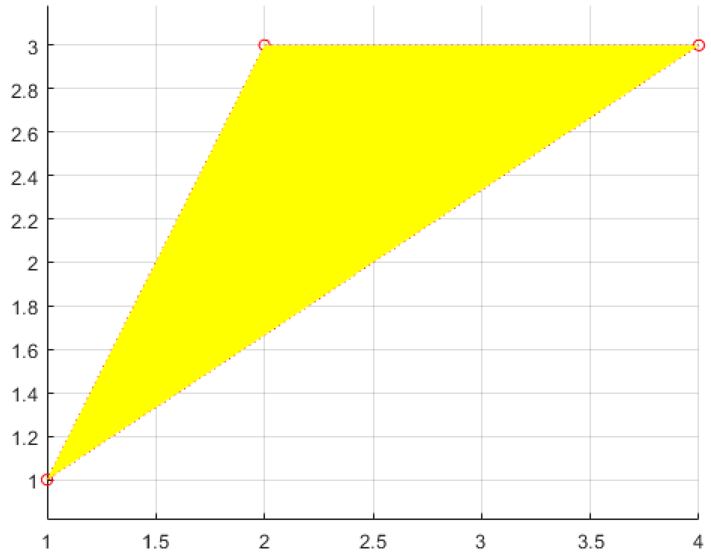
```
figure
axis equal
hold on
% coordinates of polygon vertices
nv = 12; % number of vertices
```

```
xv=1:nv; %randi(10,nv,1);
yv=randi(10,nv,1);
% plot vertices
scatter(xv,yv,30,'r')
% plot definiting polygon
plot(xv,yv,'r:')
% plot calculated points
% draw curve using 50 points in gray color
b = drawPolygon(xv,yv,'LineWidth',2,'Color',[1 1 1]*0.5,'-fill','r');
% label every 10th point
scatter(xv(1:10:end),yv(1:10:end),30,'k','filled')
grid on
```



### Example 4

```
figure
axis equal
hold on
% coordinates of polygon vertices
xv = [ 1 2 4 1];
yv = [ 1 3 3 1];
% plot vertices
scatter(xv,yv,30,'r')
% plot definiting polygon
plot(xv,yv,'r:')
% plot in gray color
b = fillPolygon('y',xv,yv);
grid on
```



# References

# drawPolyline

Draw an open 2D shape with straight sides.

## Description

Just a wrapper to MATLAB's plot.

## Syntax

drawPolyline(xp,yp)

drawPolyline(xp,yp,LineSpec)

p = drawPolygon(__)

### Method

## Arguments

### Input Arguments

**xp -** x-coordinate of polygon vertices

**yp** - y-coordinate of polygon vertices

xp, yp must be of the same size.

### Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

**p** - structure .

- p.color - line color

## Examples

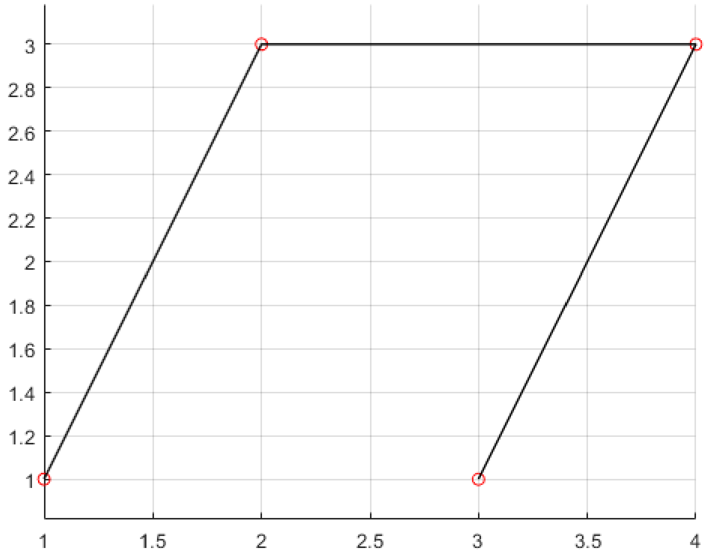
### Example 1

```
figure
hold on
axis equal
% coordinates of polygon vertices
```

```
xp=[1 2 4 3];
yp=[1 3 3 1];
% plot definiting polygon vertices
scatter(xp,yp,'r')
% plot definiting polygon
plot(xp,yp,'r')
% draw curve
p = drawPolyline(xp,yp)
```

```
p = struct with fields:
        xk: [1 3]
        yk: [1 1]
     color: 'k'
```

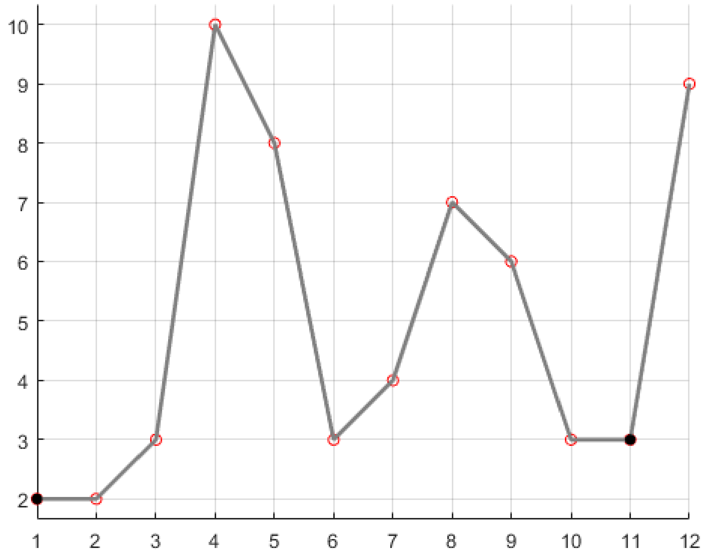
```
grid on
```



### Example 3

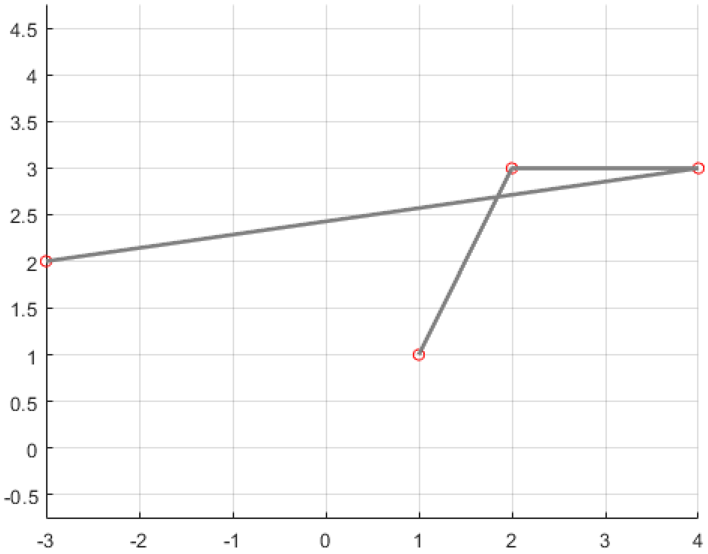
```
figure
axis equal
hold on
% coordinates of polygon vertices
nv = 12; % number of vertices
xv=1:nv; %randi(10,nv,1);
yv=randi(10,nv,1);
% plot vertices
scatter(xv,yv,30,'r')
% plot definiting polygon
plot(xv,yv,'r:')
```

```
% plot calculated points
% draw curve using 50 points in gray color
b = drawPolyline(xv,yv,'LineWidth',2,'Color',[1 1 1]*0.5);
% label every 10th point
scatter(xv(1:10:end),yv(1:10:end),30,'k','filled')
grid on
```



### Example 4

```
figure
axis equal
hold on
% coordinates of polygon vertices
xv = [ 1 2 4 -3];
yv = [ 1 3 3 2];
% plot vertices
scatter(xv,yv,30,'r')
% plot definiting polygon
plot(xv,yv,'r:')
% plot in gray color
b = drawPolyline(xv,yv,'LineWidth',2,'Color',[1 1 1]*0.5);
grid on
```



# References

# drawRect

# fillRect

Draw or fill rectangle.

## Description

## Syntax

drawRect(wd,ht,xr,yr)

drawRect(wd,ht,xr,yr,rot)

drawRect(__,'-pos',ip)

drawRect(__,'-v',v1,v2)

drawRect(__,LineSpec)

p = drawRect(__)

fillRect(c,__)

## Description

drawRect(wd,ht,xr,yr) draw rectangle with reference point at position 1.

drawRect(wd,ht,xr,yr,rot) draw rectangle rotated by given angle about reference point.

drawRect(__,'-pos',ip) draw rectangle with reference point at position ip:1,...,9, (default is 1)  (see Example 2)

drawRect(__,'-v',v1,v2) draw part of rectangle from vertex number v1 to vertex number v2 in CCLW direction. v2 is not neccesery > v1.

drawRect(_,LineSpec)  sets the line style, marker symbol, and color.

p = drawRect(__) returns structure with fields contain x-value and y-value for the rectangle.

## Method

For calculation of the coordinates  the function use **evalRect( xr,yr,wd,ht,rot,ip,v1,v2)**  which returns coordinates *x* and *y* of the vertices v1 to v2 in CCW direction. If v1=v2 the rectangle is closed. For usage of **evalRect** see Example 1.

The curve is plotted by MATLAB function plot.

## Arguments

## Input Arguments

**c** - fill color

**wd** - width

**ht** - height

**xr, yr** - reference point

## Optional Input Arguments

**rot** - rotation angle about the reference point in degrees

**'-pos',ip** - position of reference point:1,...,9 (default is 1) (see Example 2)

**'-v',v1,v2** -- start and end vertices in CCLW direction, 1<=v1,v2 <= 4. It is not neccesary that v1 < v2 (see Example 2)

## Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

**p** - structure with the fields

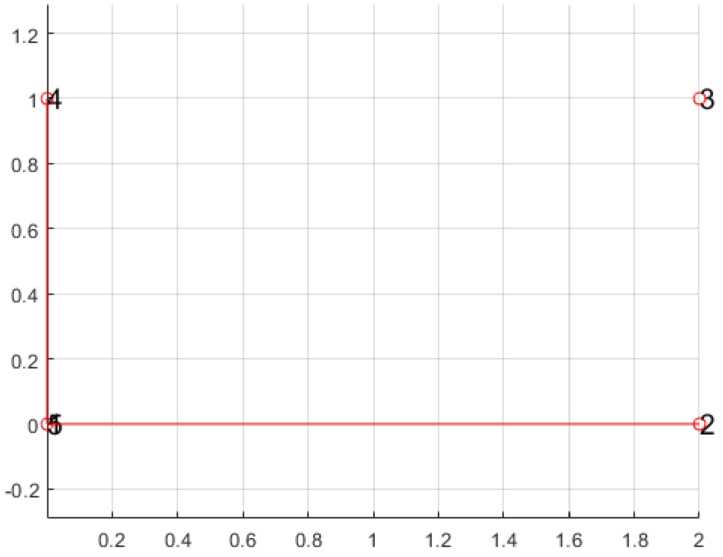
- p.xk - coordinates of 9 characteristic points
- p.yk
- p.color - line color

# Examples

## Example 1

```
figure
hold on
axis equal
% Data
n = 5;
r = 1;
% close polygon
[xv,yv] = evalRect( 2, 1, 0, 0, 0, 1, 1, 1);
% plotvertices
scatter(xv,yv,'r')
for k = 1:n
    text(xv(k),yv(k),num2str(k),'FontSize',14)
end
```

```
% polyline between two vertices
v1 = 4;
v2 = 2;
[xv,yv] = evalRect( 2, 1, 0, 0, 0, 1, v1, v2);
% plot  polyline
plot(xv,yv,'r')
grid on
```



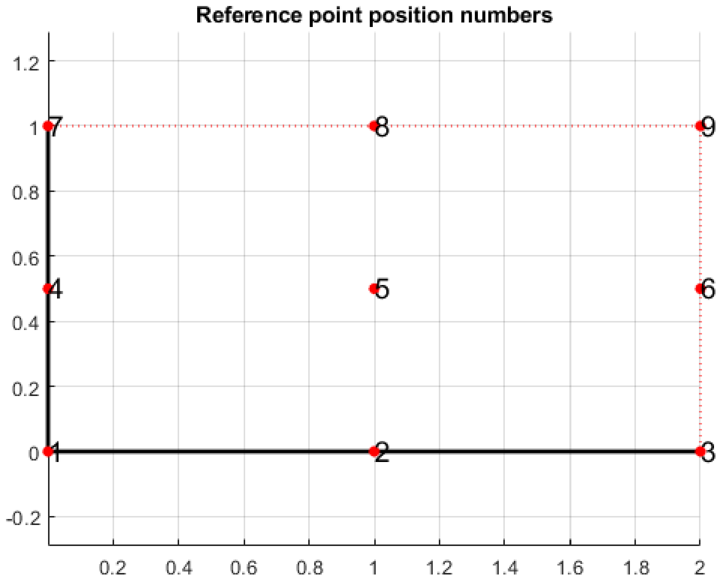
## Example 2

```
figure
axis equal
hold on
% coordinates of polygon vertices
nv = 12; % number of vertices
% plot definiting polygon
p=drawRect(2,1,0,0,'r:')
```

```
p = struct with fields:
       xk: [9×1 double]
       yk: [9×1 double]
    color: [1 0 0]
```

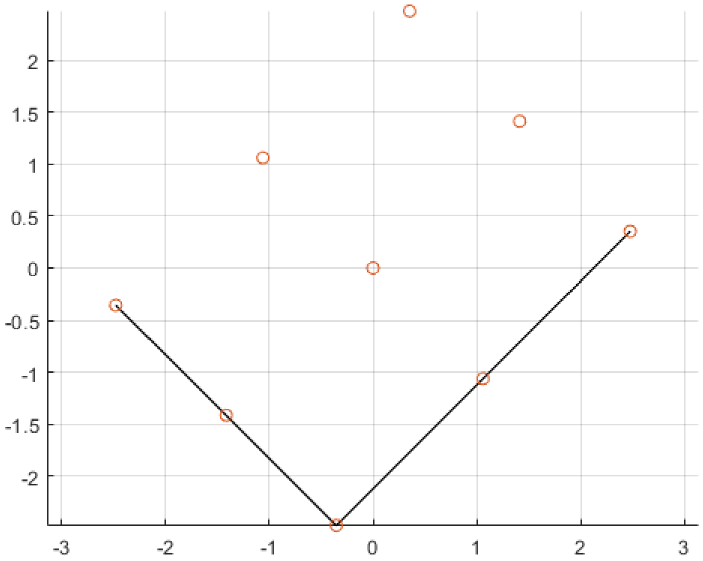
```
% Polyline between vertex 4 and 3
drawRect(2,1,0,0,'-v',4,2,'k','LineWidth',2);
%plot(x,y,'k','LineWidth',2)
% plot referebnce points
scatter(p.xk,p.yk,30,'r','filled')
for k = 1:9
```

```
    text(p.xk(k),p.yk(k),num2str(k),'FontSize',14)
end
title('Reference point position numbers')
grid on
```



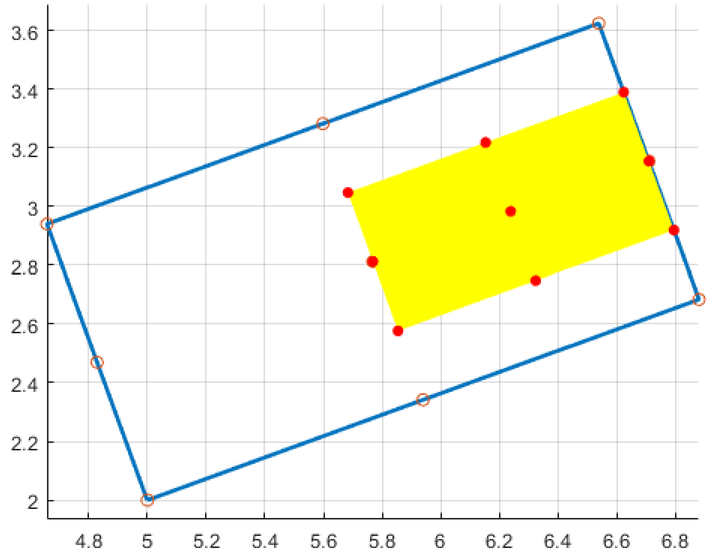
### Example 3

```
figure
axis equal
hold on
p=drawRect(4,3,0,0,45,'-v',4,2,'-pos',5);
scatter(p.xk,p.yk)
grid on
```



## Example 4

```
figure
axis equal
hold on
r1=drawRect(2,1,5,2,20,'LineWidth',2);
scatter(r1.xk,r1.yk)
r2=fillRect('y',1,0.5,r1.xk(5),r1.yk(5),20,'-pos',4);
scatter(r2.xk,r2.yk,30,'r','filled')
grid on
```



## See also

## References

# drawSave

Save figure.

## Description

Save current figure as bitmap image in jpg 24-bit format.

## Syntax

drawSave

drawSave( fileName)

drawSave(__,'-r',res)

drawSave(__,'-f',fileName)

## Description

drawSave - save current figure to file FigNN.jpg, where NN is figure number, in medium resolution ('-r300'). The file is saved in current folder.

drawSave( fileName) - save current figure to file fileName.jpg.

drawSave(__,'-r',res) - set figure resolution

drawSave(__,'-f',fileName) - set figure output file

## Method

## Arguments

### Optional Input Arguments

**fileName** - output file name (without extension !!!)

**'-r',res** - resolution 'low'w'medium'|'high', corespond to '-r100','-r300,'-r600'. or number between 10 and 1200

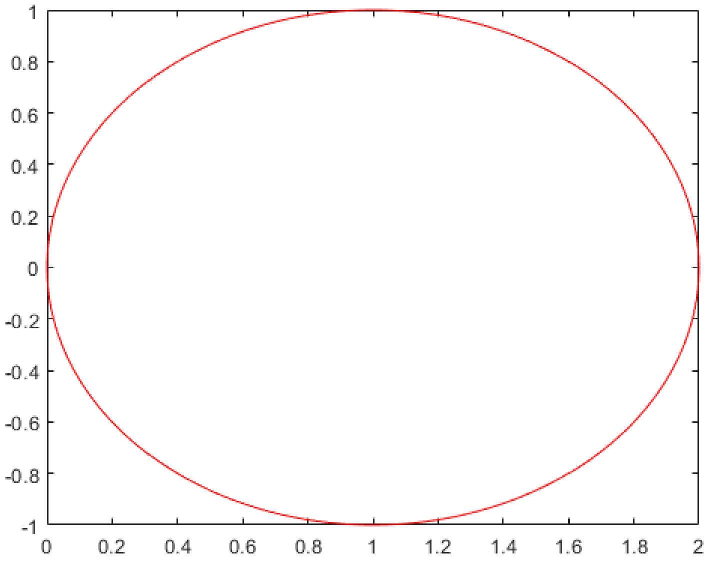
**'-f',filename** - output file name

## Examples

### Example 1

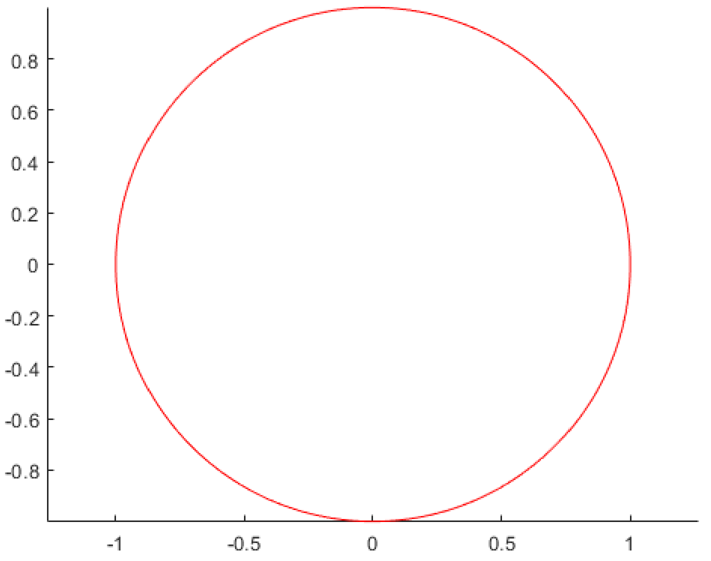
```
figure
clf
drawCircle(1,0,1,'r')  % circle is distorted
```

```
drawSave
```



## Example 2

```
drawInit
drawCircle(0,0,1,'r')
drawSave('circle','-r','low')
```



## See Also

# References

# drawSet

Set attribute of drawing entities

## Description

## Syntax

drawSet(varargin)

## Description

drawSet set the value of global variable gkdata.  The variable (table) is initialized by calling drawInit or gkInit.

## Method

## Arguments

### Optional Name-Value Pair Input Arguments

**name,value** -- name is propery (case insensitive)

for plot: 'linestyle', 'linecolor', 'linewidth'

for fill: 'facecolor', 'edgecolor', 'edgewidth', 'edgestyle'

for text:  'fontname', 'fontsize', 'fontweight', 'textcolor', 'horizontalalignment', 'verticalalignment', 'rotation'

## Examples

### Example 1

```
gkInit
drawGet('lineColor')
```

```
ans = 'k'
```

```
drawSet('linecolor','r','rotation',45)
```

```
drawGet('lineColor')
```

```
ans = 'r'
```

```
drawGet('rotation')
```

```
ans = 45
```

```
gkClose
```

## See Also

## References

# drawShow

Show current figure.

## Description

Wraper tha call the MATLAB function shg

# drawSpiral

Draw the Archimedean spiral

## Description

The Archimedean spiral is the curve that corresponding to the locations over time of a point moving away from a fixed point with a constant speed along a line that rotates with constant angular velocity [1]. Its equation in polar coordiantes $(r,\theta)$ is

$$r = c\theta$$

where *c* is real parameter. If *a* is distance between succesive turnings and $\theta$ is in degrees then the equation is

$$r = \frac{a\,\theta^0}{360}$$

## Syntax

drawSpiral(a,sang,eang)

drawSpiral(a,sang,eang,xc,yc)

drawSpiral(a,sang,eang,xc,yc,rot)

drawSpiral(a,sang,eang,LineSpec)

drawSpiral(a,sang,eang,'-np',np)

p = drawSpiral(__)

## Description

drawSpiral(a,sang,eang) draw spiral with default number of points fix(eang-sang) and current line specification.

drawSpiral(a,sang,eang,xc,yc) draw spiral with given center

drawSpiral(a,sang,eang,xc,yc,rot) draw spiral with given center and rotation

drawSpiral(a,sang,eang,LineSpec) sets the line style, line width, and color.

drawSpiral(a,sang,eang,'-np',np) set the number of points on the output curve.

p = drawSpiral(__) returns structure with fields contain x-value and y-value for the curve.

## Method

For calculation of the coordinates of the curve **drawSpiral** call function **evalSpiral( xc,yc,rot,a,th)** which returns coordinates *x* and *y* of the curve at given angles th . For usage of **evalSpiral** see Example 1.

Catenary is plotted by MATLAB function plot.

# Arguments

## Input Arguments

**a** - distance between succesive turns

**sang** - start angle in degrees

**eang** - end angle in degrees

## Optional arguments

**xc -** x-coordinate of the center

**yc** - y-coordinate of the center

**rot** - rotation angle about the center in degrees

## Optional Name-Value Pair Input Arguments

**'-np', np** - number of points along the curve ( scalar integer value > 2)

***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

**p** - structure with the fields

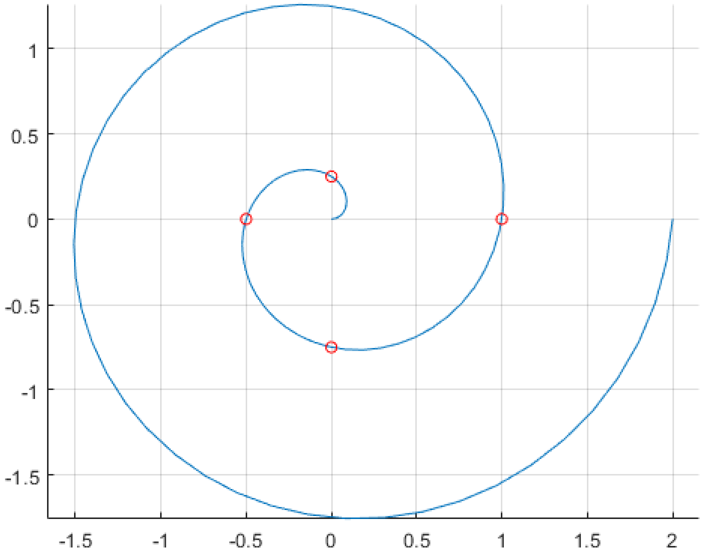
- p.xk,p.yk - key points: 1=start,2=end,3=center
- p.color - line color

# Examples

## Example 1

```
%Data
b = 1;
sang = 0;
eang = 2*360;
figure
hold on
axis equal
[x,y] = evalSpiral(b,0,0,0,linspace(sang,eang));
plot(x,y)
[x,y] = evalSpiral(b,0,0,0,[90,180,270,360]);
```

```
scatter(x,y,30,'r')
grid on
```

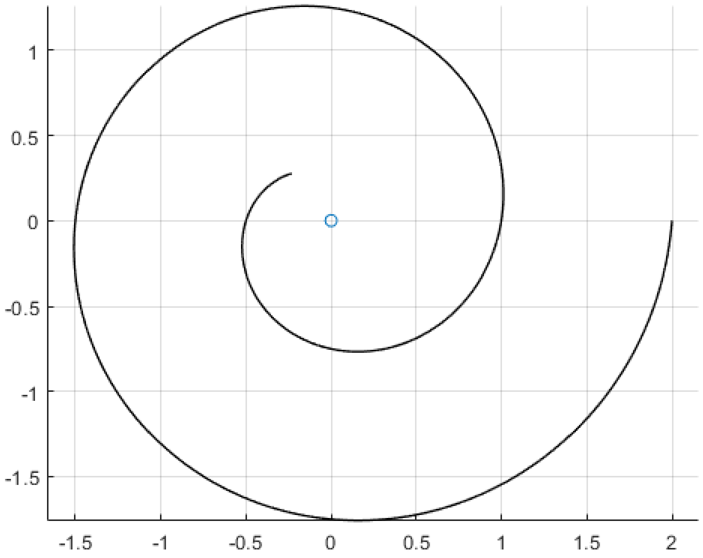


## Example 2

```
%Data
b = 1;
sang = 130;
eang = 2*360;
figure
hold on
axis equal
scatter(0,0)
p = drawSpiral(b,sang,eang)
```

```
p = struct with fields:
       xk: [-0.2321 2 0]
       yk: [0.2766 0 0]
    color: 'k'
```

```
grid on
```

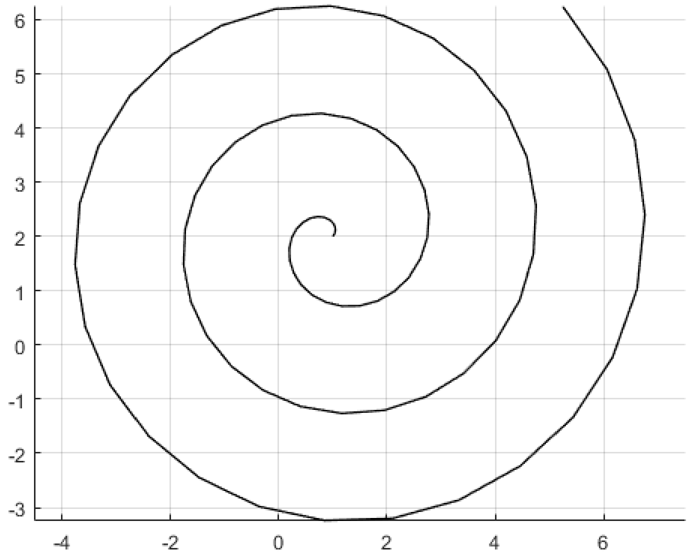


## Example 3

```
figure
hold on
axis equal
b = 2;
eang = 3*360;
x0 = 1;
y0 = 2;
rot = 45;
p = drawSpiral(b,0,eang,x0,y0,rot,'-np',80)
```

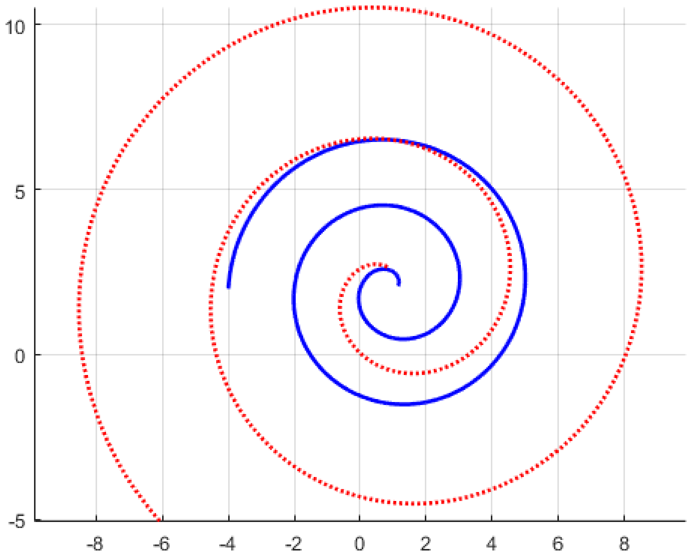
```
p = struct with fields:
       xk: [1 5.2426 1]
       yk: [2 6.2426 2]
    color: 'k'
```

```
grid on
```



## Example 4

```
figure
hold on
axis equal
a = 2;
b = 2;
sang = 30;
eang = 2.5*360;
x0 = 1;
y0 = 2;
rot = 45;
drawSpiral(b,sang,eang,x0,y0,'LineWidth',2,'Color','b')
drawSpiral(2*b,2*sang,eang,x0,y0,rot,'LineWidth',2,'Color','r','LineStyle',':')
grid on
```



## See Also

## References

[1] WikipediA, Archimedean spiral

# drawSpline

Draw 2-D cubic spline curve fits  given points.

## Description

## Syntax

drawSpline(ctype,xp,yp)

drawSpline(ctype,xp,yp,'-s1',u,'-s2',v)

drawSpline(__,LineSpec)

p = drawSpline(__)

## Description

drawSpline(ctype,xp,yp) draw cubic spline with default number of points.

drawSpline(ctype,xp,yp,LineSpec)  sets the line style, marker symbol, and color.

drawSpline(ctypexp,yp,'-np',np) draw curve using np points.

p = drawSpline(__) returns structure with fields contain x-value and y-value for the curve.

## Method

For calculation of the coordinates of the curve **drawSpline** call function **evalSpline( xp, yp,np)** which returns coordinates *x* and *y* of the curve at given values of parameter *t*.  Function **evalSpline** use MATLAB function **spline**.

The function **evalSpline** calculates the parameter range based upon chord distance between

data points rather than using normalized splines.

The curve is plotted by MATLAB function plot.

## Arguments

### Input Arguments

**ctype -** Spline type:

- 1= cubic:
- 2=Wilson-Fowler: the chord length as the independent parameter

**xp -** x-coordinate of polygon vertices (real vector)

**yp** - y-coordinate of polygon vertices (real vector)

xp, yp must be of the same size.

### Optional Name-Value Pair Input Arguments

**'-s1',u1** - slope vector at start point (if u2 is not given than u2=[0 0])

**'-s2',u2** - slope vector at end point (if u1 is not given than u1=[0 0])

**'-np', np** - number of points along the curve, np is scalar integer value > 2.

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

**p** - structure with fields

- p.xk, p.yk - spline end points
- p.th - tangen angle in end points in degrees
- p.x - spline coordinates
- p.y
- p.color - line color

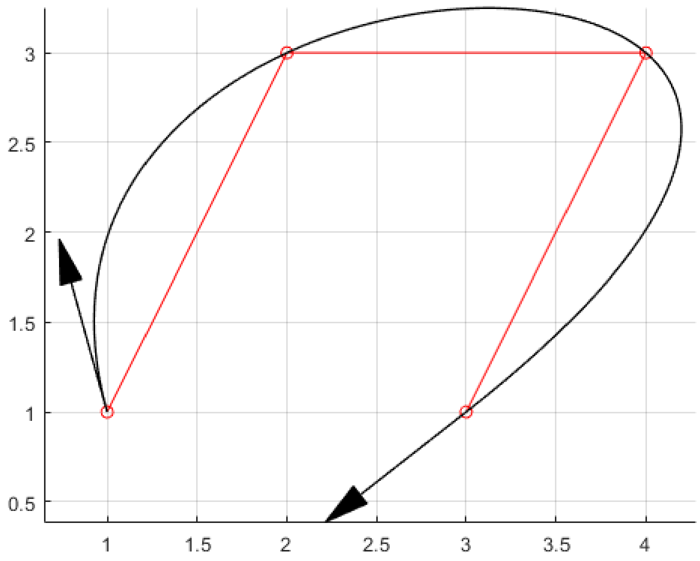
## Examples

### Example 1

```
drawInit
% coordinates of polygon vertices
xp=[1 2 4 3];
yp=[1 3 3 1];
% plot definiting polygon vertices
scatter(xp,yp,'r')
% plot definiting polygon
plot(xp,yp,'r')
% draw curve
p = drawSpline(1,xp,yp)
```

```
p = struct with fields:
       th: [105.5241 -141.9530]
        x: [1×100 double]
        y: [1×100 double]
       xk: [1 3]
       yk: [1 1]
    color: 'k'
```

```
drawArrow(3,0.25,0.125,xp(1),yp(1),'-rtheta',1,p.th(1))
drawArrow(3,0.25,0.125,xp(4),yp(4),'-rtheta',1,p.th(2))
grid on
```



## Example2

```
figure
hold on
axis equal
% coordinates of polygon vertices
xp=[1 2 4 3];
yp=[1 3 3 1];
u = [2-1, 3-1];
v = [3-4,1-3];
% plot definiting polygon vertices
scatter(xp,yp,'r')
% plot definiting polygon
plot(xp,yp,'r')
% draw curve
p = drawSpline(1,xp,yp,'-s1',u,'-s2',v);
p = drawSpline(1,xp,yp,'-s1',u,'k')
grid on
```

### Example 3

```
drawInit
% coordinates of polygon vertices
nv = 12; % number of vertices
```

```
xv=1:nv; %randi(10,nv,1);
yv=randi(10,nv,1);
% plot vertices
scatter(xv,yv,30,'r')
% plot definiting polygon
%plot(xv,yv,'r:')
% plot calculated points
% draw curve using 50 points in gray color
b1 = drawSpline(1,xv,yv,'-np',250,'LineWidth',1,'Color','b');
b = drawSpline(2,xv,yv,'-np',250,'LineWidth',2,'Color',[1 1 1]*0.5);
% label every 10th point
%scatter(b.x(1:10:end),b.y(1:10:end),30,'k','filled')
legend({'data','cubic','Wilson-Fowler'},'Location','best')
grid on
```

### Example 4

```
figure
axis equal
hold on
% coordinates of polygon vertices
xv = [ 1 2 4 1];
yv = [ 1 3 3 1];
% plot vertices
scatter(xv,yv,30,'r')
% plot definiting polygon
plot(xv,yv,'r:')
% plot calculated points
% draw curve using 50 points in gray color
b = drawSpline(1,xv,yv,'-np',50,'LineWidth',2,'Color',[1 1 1]*0.5);
% label every 10th point
scatter(b.x(1:10:end),b.y(1:10:end),30,'k','filled')
grid on
```

## See also

Spline

## References

# drawSpring1

Draw a coil spring

## Description

## Syntax

drawSpring1(form,r,nc,x1,y1,x2,y2)

drawSpring1(form,r,nc,x1,y1,'-delta',dx,dy)

drawSpring1(form,r,nc,x1,y1,'-polar',r,th)

drawSpring1(__,'-type','flat'|'ext')

drawSpring1(__,LineSpec)

p = drawSpring1(__)

## Description

## Method

## Arguments

### Input Arguments

**form** --- spring type: 1=flat,2=ext

**r ---** radius

**nc** --- number of circuts

**x1, y1** --- start point

**x2, y2** -- end point

or

**'-polar'|'-rtheta', r, th** -- polar coordiantes of end point

or

**'-delta', dx, dy** -- delta coordinates of end point

### Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

**p** - structure with the fields

- p.xk, p.yk - key points:1=start,2=end
- p.color - line color

## Examples

### Example 1

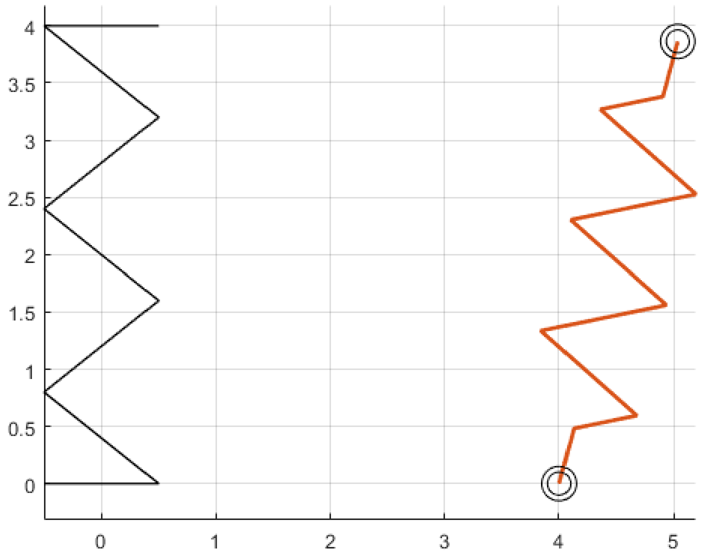
```
drawInit
x0 = 0; y0=0;nc=3;r=1/2,
```

```
r = 0.5000
```

```
p0=drawSpring1(1,r,nc,x0,y0,x0,4);
p= drawSpring1(2,r,nc,x0+4,y0,'-rtheta',4,75,'LineWidth',2)
```

```
p = struct with fields:
       xk: [4 5.0353]
       yk: [0 3.8637]
    color: [0.8500 0.3250 0.0980]
```

```
drawDonut(0.3,0.2,p.xk(1),p.yk(1),'k')
drawDonut(0.3,0.2,p.xk(2),p.yk(2),'k')
grid on
```



## See Also

## References

# drawSupport

Draw a beam support

## Description

## Syntax

drawSupport(type,x0,y0,ht,th)

drawSupport(__,LineSpec)

p = drawSpring(__)

## Description

## Method

## Arguments

### Input Arguments

**type -** 1=fixed,2=simple

**x0, y0** --- base point (bottom, center)

**ht** --- height

**rot** --- support inclination angle in deg (0, default)

### Optional Name-Value Pair Input Arguments

***LineSpec*** - specifies line properties, see Line Properties.

### Optional Output Arguments

**p** - structure with the fields

- p.xk,p.yk -- key points
- p.color - line color

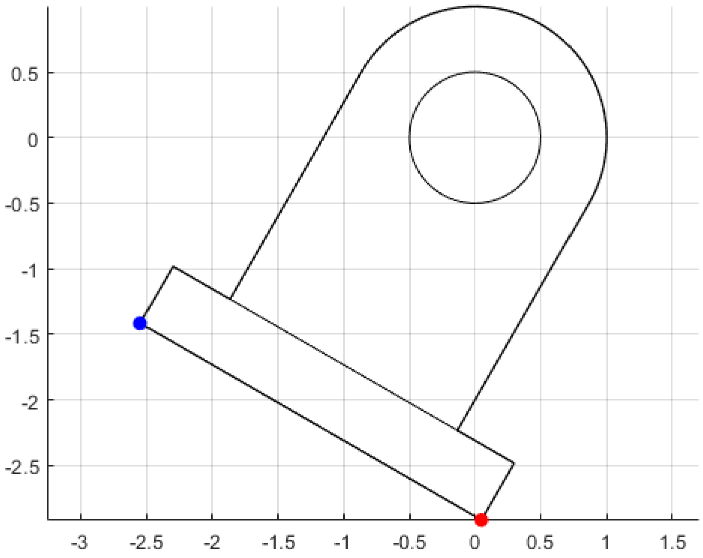
## Examples

### Example 1

```
drawInit
x0 = 0; y0=0; r=1;ht=1;th=-30
```

```
th = -30
```

```
p = drawSupport(1,r,x0,y0,th,'b');
scatter(p.xk(1),p.yk(1),50,'b','filled')
scatter(p.xk(2),p.yk(2),50,'r','filled')
grid on
```

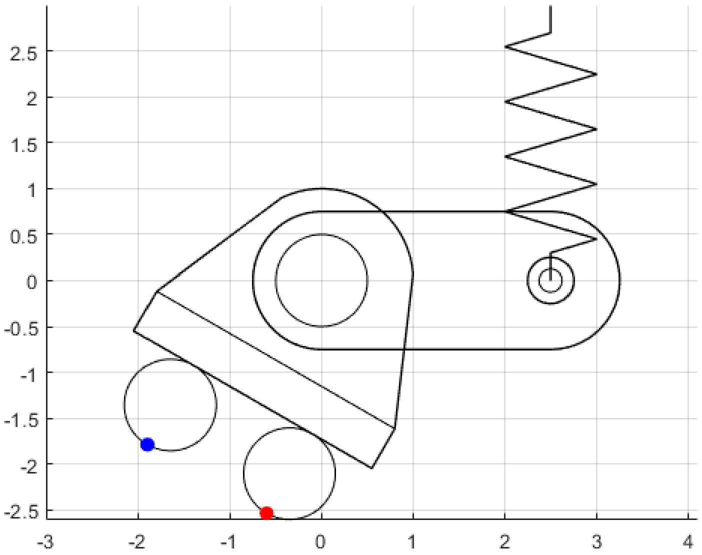


## Example 2

```
drawInit
x0 = 0; y0=0; r=1;ht=1;th=-30
```

```
th = -30
```

```
p = drawSupport(2,r,x0,y0,th,'b');
c=drawCanoe(4,1.5,0,0,0,'-pos',4);
drawSpring(c.xk(6),c.yk(6),0.5,3,4,90,'-type','ext')
drawDonut(0.5,0.25,c.xk(6),c.yk(6))
scatter(p.xk(1),p.yk(1),50,'b','filled')
scatter(p.xk(2),p.yk(2),50,'r','filled')
grid on
```



## See Also

## References

# drawVDim

Draw vertical dimension

## Description

Vertical dimension measures  the vertical distance between two points.

## Syntax

drawVDim(form,d1,d2,x1,y1,x2,y2,xt,yt)

drawVDim(__,'-str',str)

drawVDim(_,LineSpec)

### Description

drawVDim(form,d1,d2,x1,y1,x2,y2,xt,yt) draw horizontal dimension between points (x1,y1) and (x2,y2) and locate text, i.e. value of distnce y2-y1, at point (xt,yt). If y2 < y1 than the points are swaped.

drawVDim(__,'-str',str) draw horizontal dimension between points (x1,y1) and (x2,y2) and locate text given by variable str at point (xt,yt).

drawVDim(__,LineSpec)  sets the line style, line width, and color.

### Method

drawVDim use drawLine, drawArrohwhead, and gkText to draw a vertical dimension . For drawing text the function drawVDim use current text attrubtes. They can be changed by function drawSet.

## Arguments

### Input Arguments

**form** - arrowhead form

**d1,d2** - arrow head width and height

**x1,y1** - start point. y1 colud be > y2

**x2,y2** - end point

**xt,yt**    - text location. If xt is inside interval (x1,x2) then text is located at the center of the interval

## Optional Name-Value Pair Input Arguments

**'-str'|'-txt',str** - dimension text

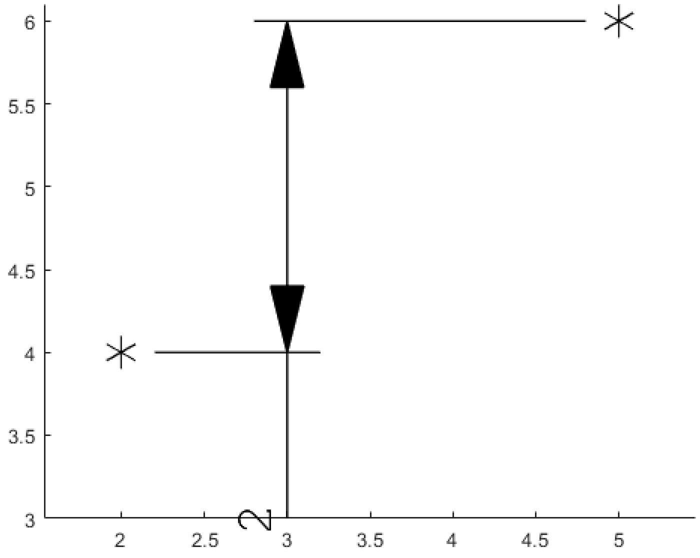
***LineSpec*** - specifies line properties, see Line Properties.

## Optional Output Arguments

# Examples

## Example 1

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 4;
x2 = 5; y2 = 6;
xt = 3; yt = 3;
gkSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawPoint(1,ad1/2,x2,y2)
drawVDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
```

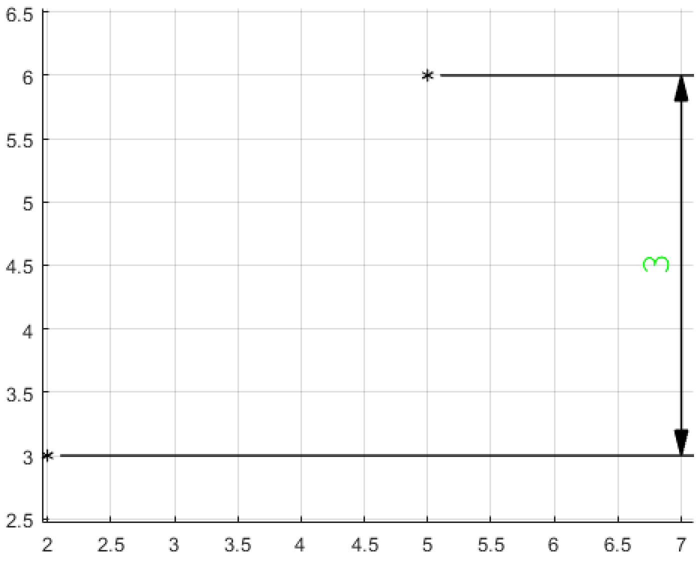


```
%grid on
```

## Example 2

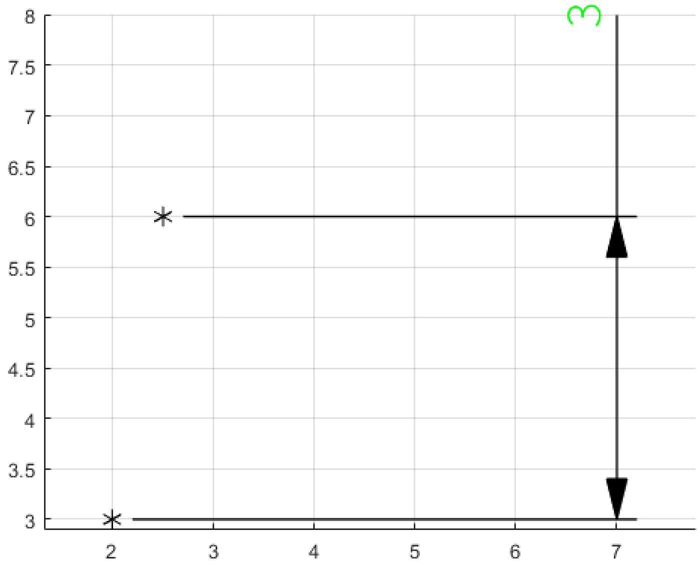
```
drawInit
```

```
ad1 = 0.2; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 5; y2 = 6;
xt = 7; yt = 4;
drawSet('FontSize',20,'textColor','g')
drawPoint(1,ad1/2,x1,y1)
drawPoint(1,ad1/2,x2,y2)
drawVDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
grid on
```



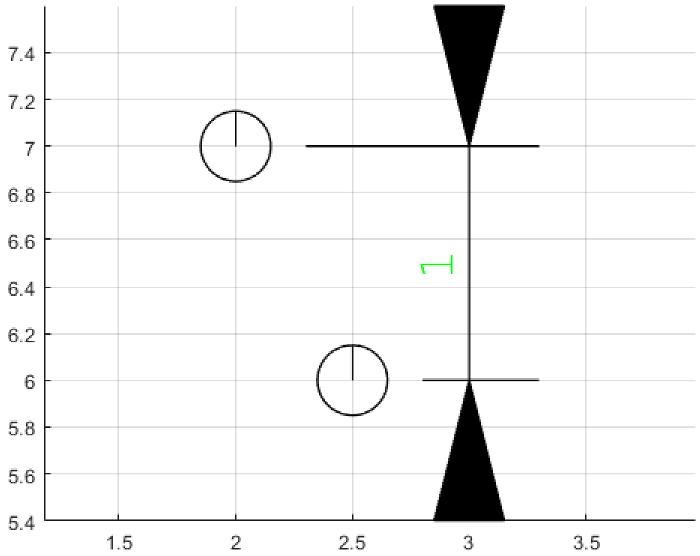
## Example 3

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 2.5; y2 = 6;
xt = 7; yt = 8;
drawSet('FontSize',26)
drawPoint(1,ad1/2,x1,y1)
drawPoint(1,ad1/2,x2,y2)
drawVDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
grid on
```



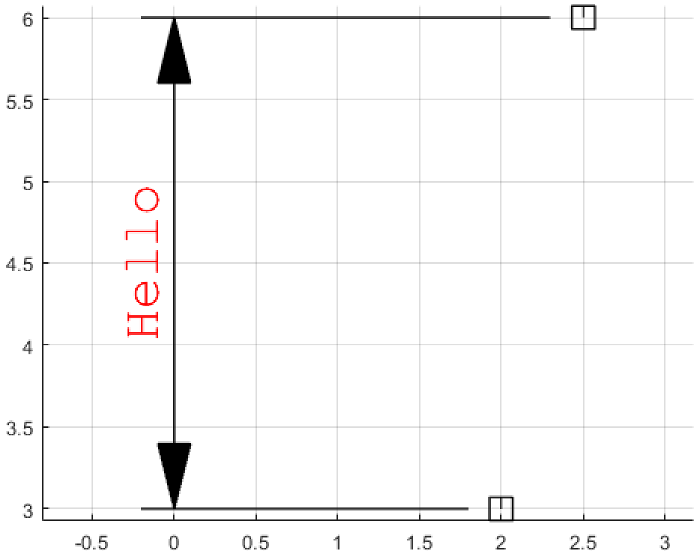
## Example 4

```
drawInit
ad1 = 0.6; ad2 = ad1/2;
x1 = 2; y1 = 7;
x2 = 2.5; y2 = 6;
xt = 3; yt = 7;
drawSet('FontSize',26)
drawPoint(3,ad2,x1,y1)
drawPoint(3,ad2,x2,y2)
drawVDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt)
grid on
```



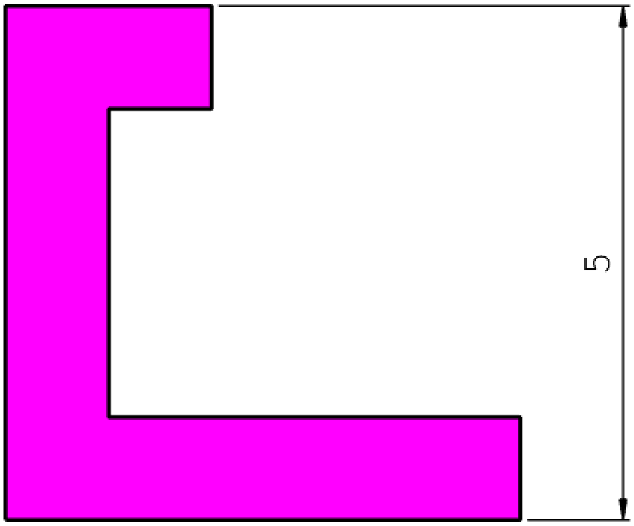
## Example 5

```
drawInit
ad1 = 0.4; ad2 = ad1/2;
x1 = 2; y1 = 3;
x2 = 2.5; y2 = 6;
xt = 0; yt = 4;
drawSet('FontSize',26,'textColor','r')
drawPoint(2,ad1/2,x1,y1)
drawPoint(2,ad1/2,x2,y2)
drawVDim(3,ad1,ad2,x1,y1,x2,y2,xt,yt,'-str','Hello')
grid on
```



## Example 6

```
 drawInit
 ad1 = 0.2; ad2 = ad1/3;
 a = 5; b = 3; c = 5; d = 1;
 p = drawPolygon([0 a a d d   b-d b-d 0],...
                 [0 0 d d c-d c-d c   c ],'LineWidth',2,'Color','k','-
fill','m');
 drawSet('FontSize',20,'textColor','k')
 drawVDim(3,ad1,ad2,p.xk(2),p.yk(2),p.xk(7),p.yk(7),a*1.2,c/2)
 axis off
```



## See also

## References