PROBLEM 1

Consider the following sequence of rotations:

1. Rotate by φ about the world x-axis.
2. Rotate by θ about the world z-axis.
3. Rotate by ψ about the current x-axis.
4. Rotate by α about the world z-axis.

Write the matrix product that will give the resulting rotation matrix.

SOLUTION:

Remember the two basic rules for the composition of transformations:

- When we use "moving axes" (that is, each new rotation is relative to the new coordinate frame), we compose the transformations from left to right.

- When we use "fixed axes" (that is, each new rotation is relative to the original coordinate frame), we compose the transformations from right to left.

According to these rules, we can obtain the sought transformation using the Robotics

```
%%% First we clean the workspace

clc;
clear;

%%% We declare phi, theta, psi, and alpha as symbolic variables

phi   = sym('phi');
theta = sym('theta');
psi   = sym('psi');
alpha = sym('alpha');

%%% We compute the sequence of rotations

R = rotz(alpha)*rotz(theta)*rotx(phi)*rotx(psi)

Eva_R=subs(R, [phi,theta,psi,alpha],[pi/4,pi/4,pi/4,pi/4])


%%% To see an animation
A=eye(4)
phi=pi/4
theta=pi/4
psi=pi/4
alpha=pi/4
B=trotx(phi)*A
C=trotz(theta)*B
D=C*trotx(psi)
E=trotz(alpha)*D
%%% To see an
tranimate(A,B)
tranimate(B,C)
tranimate(C,D)
tranimate(D,E)
%% Chek that the figure match with the Eva_R rotation Matrix
```

PROBLEM 2

Consider the wedge-shaped object in the following drawing.

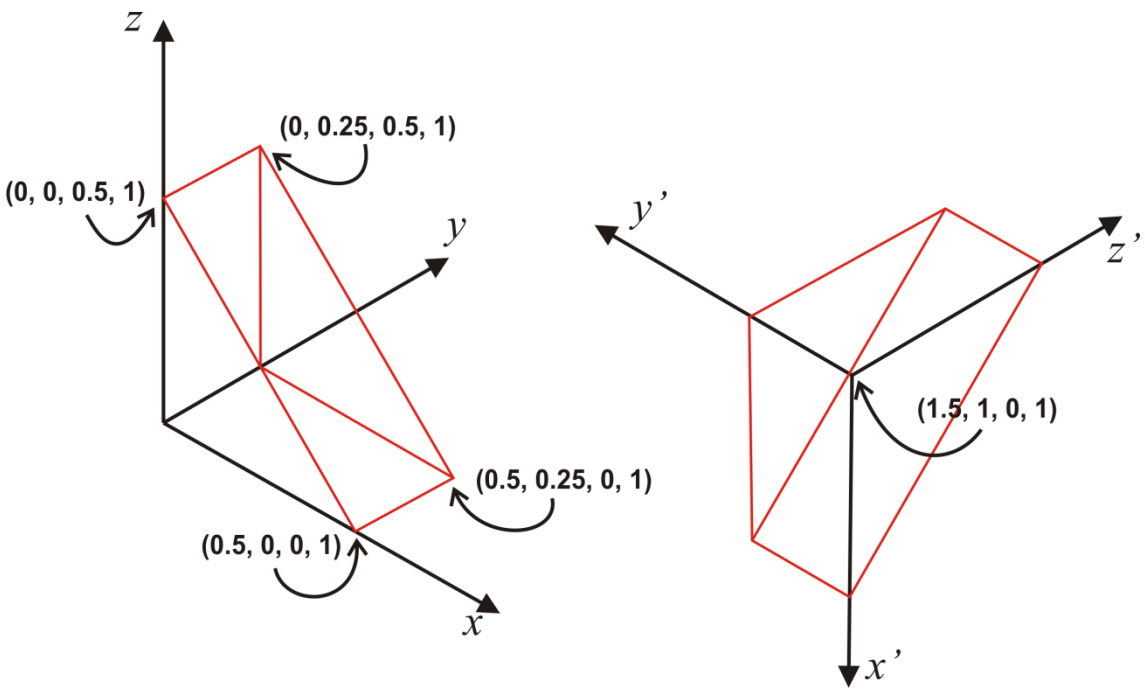
1) Obtain the transformation that should be applied to take it from the origin (left) to its final location (right). Represent this transformation as a translation followed by a rotation and vice versa. Verify that the solution is the same.

2) Compute the coordinates of the vertices of the translated and rotated wedge with respect to the original frame.

3) Decompose the rotation obtained in question 1) as a sequence of rotations about the axes X, Y, and Z.

4) The same as above but as a sequence of rotations about the axes Z, Y, and Z.

5) Obtain the equivalent axis of rotation, and the angle rotated about it, for the rotation obtained in question 1).

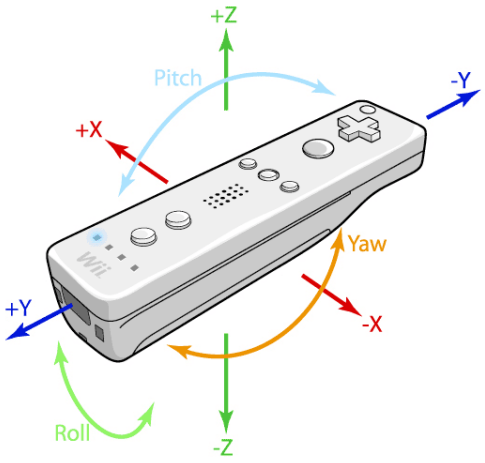


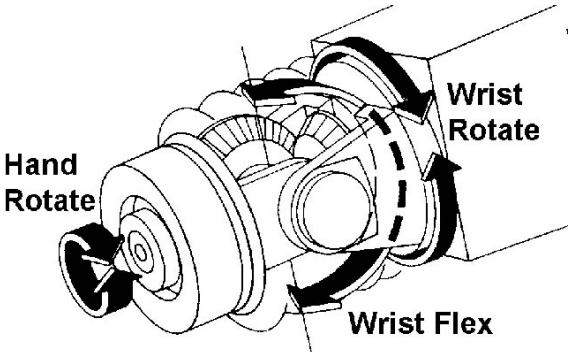
SOLUTION:

```
%%% First we clean the workspace

clc;
clear;
%%% QUESTION 1:
%%% The rotation is given by the transformation

TR = [0 -1 0 0; 0 0 1 0; -1 0 0 0; 0 0 0 1];

%%% If we translate with respect to the original reference frame, the
%%% transformation is obtained by post-multiplying the rotation

T1 = transl(1.5, 1, 0)*TR;

%%% If we translate with respect to the transformed reference frame,
%%% the transformation is obtained by pre-multiplying the rotation

T2 = TR*transl(0, -1.5, 1);

%%% It can be checked that T1 and T2 are identical

%%% QUESTION 2:
%%% The coordinates of the six vertices of the wedge can be organized
%%% as the columns of a matrix as follows:

Vertices = [0   0.5 0   0    0.5   0;    ...
            0   0   0   0.25 0.25  0.25; ...
            0   0   0.5 0    0     0.5;  ...
            1   1   1   1    1     1];

%%% Then, the coordinates of the six vertices of the wedge after it
%%% is translated and rotated are:

NewVertices = T1*Vertices

%%% QUESTION 3:
%%% The decomposition of a rotation into a sequence of rotations
%%% about axes X, Y, and Z, corresponds to the computation of
%%% the roll, pitch, and yaw angles

q = tr2rpy(TR)

%%% Then, we can verify that TR is identical to

trotx(q(1))*troty(q(2))*trotz(q(3))

%%% QUESTION 4:
%%% The decomposition of a rotation into a sequence of rotations
%%% about axes Z, Y, and Z, corresponds to the computation of
%%% the Euler angles

q = tr2eul(TR)

%%% Then, we can verify that TR is identical to

trotz(q(1))*troty(q(2))*trotz(q(3))

%%% QUESTION 5:

[ang vec] = tr2angvec(TR);
```

PROBLEM 3

Let us suppose that we want to make a robot hand to follow the orientation of a WII™ remote controller. The problem is that the orientation of the remote controller is given in terms of roll-pitch-yaw (RPY) angles while the orientation of the robot's hand is set using ZYZ Euler angles.





1) Using the Robotics Toolbox, write a routine that returns the Euler angles corresponding to a set of RPY angles.

2) There are two solutions to determine Euler angles from a rotation matrix. The Robotics Toolbox makes the solution unique by constraining the second rotation to lie between 0° and 180°. Modify the routine written above to provide both solutions.

SOLUTION:

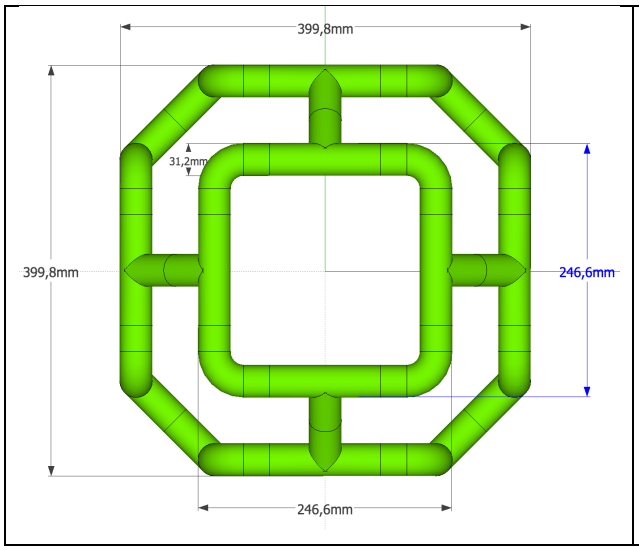
```
%%% QUESTION 1:

function euler = rpy2eul(rpy)
% RPY2EUL Converts a RPY angles to Euler angles form
%
%    [PHI THETA PSI] = RPY2EUL([ROLL PITCH YAW])
%
% Returns a vector of 3 angles corresponding to rotations about
% the Z, Y and Z axes, respectively, from roll/pitch/yaw angles.
%
% See also:  RPY2TR, TR2EUL

    euler =  tr2eul(rpy2tr(rpy));

end
```

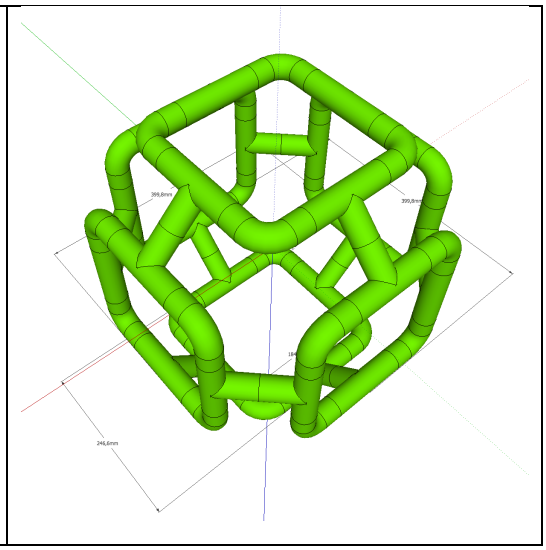
```
%%% QUESTION 2:
%% For example, to compute the equivalent rotation matrix for
R = (0.1, 0.2, 0.3) using euler angles we write
```
R = rotz(0.1) * roty(0.2) * rotz(0.3);
```
%% or more conveniently
```
R = eul2r(0.1, 0.2, 0.3)
```
%% R =
%%0.9021 -0.3836 0.1977
%%0.3875 0.9216 0.0198
%%-0.1898 0.0587 0.9801
The inverse problem is finding the Euler angles that correspond to a
given rotation matrix
```
gamma = tr2eul(R)
```
%%gamma =
%%0.1000 0.2000 0.3000
%%However if θ is negative
```
R = eul2r(0.1 , -0.2, 0.3)
```
%%R =
%%0.9021 -0.3836 -0.1977
%%0.3875 0.9216 -0.0198
%%0.1898 -0.0587 0.9801
%% The inverse function
```
tr2eul(R)
```
%%ans =
%%-3.0416 0.2000 -2.8416
%%Returns a positive value for θ and quite different values for φ and
%% ψ. However the corresponding rotation matrix
```
eul2r(ans)
```
%%ans =
%%0.9021 -0.3836 -0.1977
%%0.3875 0.9216 -0.0198
%%0.1898 -0.0587 0.9801
%%is the same – the two different sets of Euler angles correspond to
%%the one rotation matrix. The mapping from rotation matrix to Euler
angles is not unique and always returns a positive angle for θ.

%%The rotation matrix given by a set of Euler angles (phi, theta, psi)
%%can be expressed as

R = rotz(phi)*roty(theta)*rotz(psi)

%% Now observe that

roty(theta) = rotz(pi)*roty(-theta)*rotz(pi)

%%Then,

R = rotz(phi)*rotz(pi)*roty(-theta)*rotz(pi)*rotz(psi)

%%In other words,

R = rotz(phi+pi)*roty(-theta)*rotz(psi+pi)

Thus, we conclude that, if (phi, theta, psi) is a set of Euler angles, (phi+pi, -theta, psi+pi) is another set of Euler angles that represent the same orientation. If theta is constrained to be in the interval [0, pi], then –theta is in the interval [-pi, 0].

The above function can be modified to provide both solutions as follows:
```

```
function euler = rpy2eul(rpy, sigma)
% RPY2EUL Converts a RPY angles to Euler angles form
%
%   [PHI THETA PSI] = RPY2EUL([ROLL PITCH YAW])
%
% Returns a vector of 3 angles corresponding to rotations about
% the Z, Y and Z axes, respectively, from roll/pitch/yaw angles.
%
% If sigma=-1 theta is in the interval [-pi, 0]. Otherwise is in
% the interval [0, pi]
%
% See also:  RPY2TR, TR2EUL

euler =  tr2eul(rpy2tr(rpy));

if sigma == -1
        euler(1) = euler(1) + pi;
        euler(2) = - euler(2);
        euler(3) = euler(3) + pi;
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

PROBLEM 4

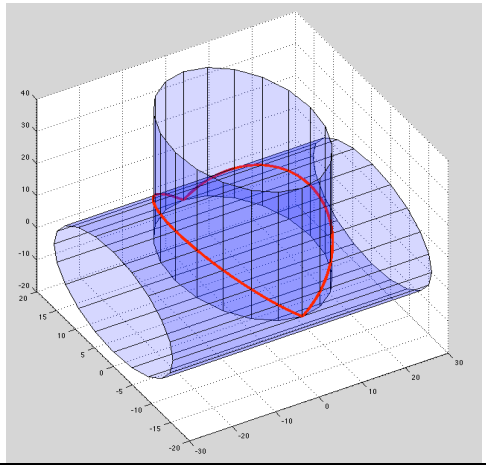
A Unimation Puma 560 robot is used to weld a folded tubes frame as it is shown in the next figures.





The task for the Puma 560 consists in welding the six folded squared tube among them. The welding trajectory can be assumed to as two orthogonal and intersecting cylinders with radius $r = 15.6$mm. The trajectory to be followed by the welder can be parameterized as follows and it can be shown below:

$$p(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} r\cos(t) \\ r\sin(t) \\ [\![r\cos(t)]\!] \end{pmatrix}; t \in [0, 2\pi]$$



1. Write the code[1] to obtain 32 welding points of two intersecting tubes. See Ilustración 1 Points at reference frame.
2. Place on site all necessary reference frames to perform the welding of the 24 locations. Draw two intersecting cylinder in any of the 24 sites. See Ilustración 2 References frames and intersecting cylinders.[2]

[1] Use the Robotics Tool Box functions: trotxyz, transl(x,y,z), trplot, …..

3. Write the code[1] to obtain the welding points for all the intersecting squared

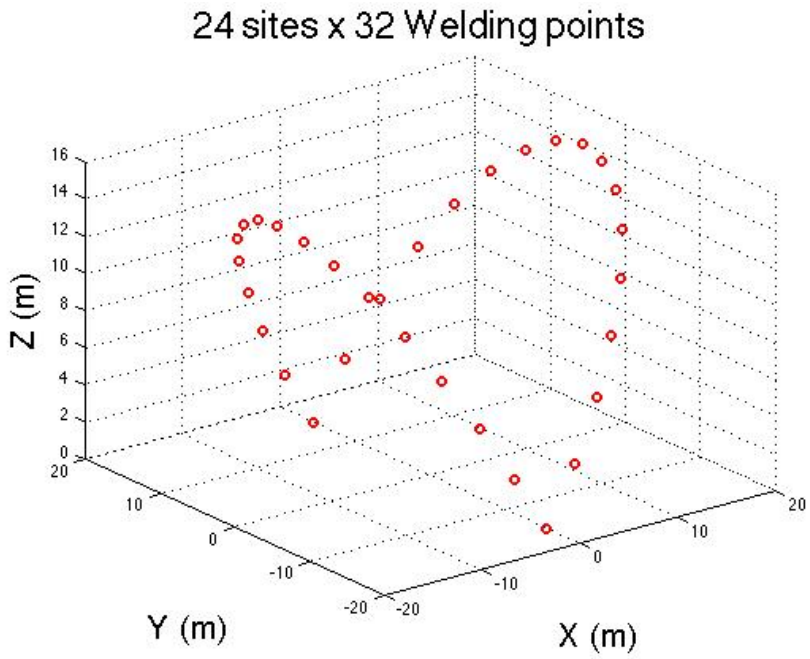


Ilustración 1 Points at reference frame

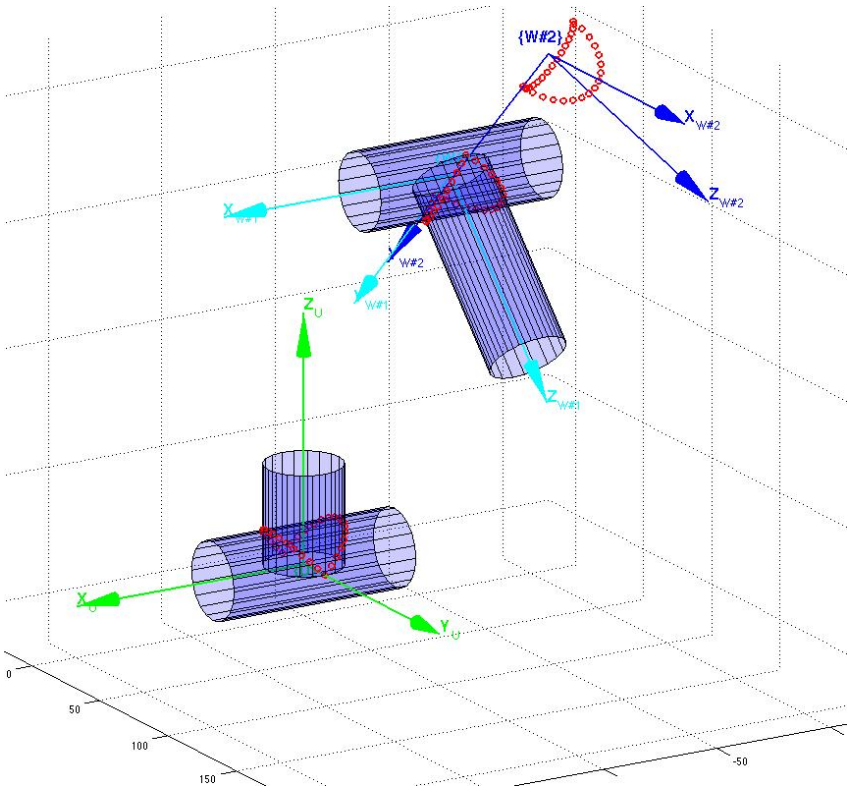


Ilustración 2 References frames and intersecting cylinders

[2] Use the following Matlab functions: Cylinder, hold on, alpha, ….

```
%% RKD&C - Exam 20th of January 2016. Problem 1. Kinematics
%
clear all
%% First question:
% Write the code to obtain the vector with 32 points of the two
intersecting tubes.
t=0:pi/16:2*pi;
r=15.6;
pt=[r*cos(t);r*sin(t);abs(r*cos(t));ones(1,length(t))]

%% Plotting the points in the space
scatter3(pt(1,:),pt(2,:),pt(3,:),'r','LineWidth',2)
% Create xlabel
xlabel('X (m)','FontSize',20);
% Create ylabel
ylabel('Y (m)','FontSize',20);
% Create zlabel
zlabel('Z (m)','FontSize',20);
% Create title
title('24 sites x 32 Welding points','FontSize',24);

grid on;
axis equal;
hold on;

%% Plotting two intersecting cylinders. This was no asked
% Vertical cylinder
[X,Y,Z] = cylinder
xyz_down=[r*X(1,:);r*Y(1,:);Z(1,:);ones(1,length(X))]
xyz_up=transl(0,0,40)*xyz_down

Xv=[xyz_down(1,:);xyz_up(1,:)]
Yv=[xyz_down(2,:);xyz_up(2,:)]
Zv=[xyz_down(3,:);xyz_up(3,:)]

surf(Xv, Yv, Zv);
axis([-50 50 -50 50 -20 20])
view(136,26)
alpha(0.2);
colormap([1 0 1]);

% Horizontal cylinder
xyz_down_rot=troty(pi/2)*transl(0,0,-40)*xyz_down
xyz_up_rot=transl(80,0,0)*xyz_down_rot

hold on

Xh=[xyz_down_rot(1,:);xyz_up_rot(1,:)]
Yh=[xyz_down_rot(2,:);xyz_up_rot(2,:)]
Zh=[xyz_down_rot(3,:);xyz_up_rot(3,:)]

surf(Xh, Yh, Zh);
%axis([-50 50 -50 50 -20 20])

alpha(0.2);
colormap([0 0 1]);
view(136,26)



%% Plotting the pipes
% Translating the weld point
t_z=(399.8/2)-r
t_y=(246.6/2)-r
```

```
xyz_down_t_r=transl(0,t_y,t_z)*trotx(-3*pi/4)*xyz_down
xyz_up_t_r=transl(0,t_y,t_z)*trotx(-3*pi/4)*transl(0,0,80)*xyz_down

X_t_r=[xyz_down_t_r(1,:);xyz_up_t_r(1,:)]
Y_t_r=[xyz_down_t_r(2,:);xyz_up_t_r(2,:)]
Z_t_r=[xyz_down_t_r(3,:);xyz_up_t_r(3,:)]

surf(X_t_r, Y_t_r, Z_t_r)

%axis([-200 200 -200 200 -20 200])

alpha(0.2);
colormap([0 0 1]);
view(90,0)

xyz_down_t_r=transl(0,t_y,t_z)*xyz_down_rot
xyz_up_t_r=transl(80,0,0)*xyz_down_t_r

X_t_r=[xyz_down_t_r(1,:);xyz_up_t_r(1,:)]
Y_t_r=[xyz_down_t_r(2,:);xyz_up_t_r(2,:)]
Z_t_r=[xyz_down_t_r(3,:);xyz_up_t_r(3,:)]

surf(X_t_r, Y_t_r, Z_t_r);
axis([-120 120 -20 220 -20 220])

alpha(0.2);
colormap([0 0 1]);

view(90,0)

%% Plotting the n_Welding
pt_t_r=transl(0,t_y,t_z)*trotx(-3*pi/4)*pt
scatter3(pt_t_r(1,:),pt_t_r(2,:),pt_t_r(3,:),'r','LineWidth',2)
pt_t_r=trotz(pi/2)*transl(0,t_y,t_z)*trotx(-3*pi/4)*pt
scatter3(pt_t_r(1,:),pt_t_r(2,:),pt_t_r(3,:),'r','LineWidth',2)

U=eye(4)
trplot(U, 'frame','U', 'text_opts', {'FontSize',16,
'FontWeight','bold'},...
       'color','g','arrow','length',100,'width', 2 )
U_W_1=transl(0,t_y,t_z)*trotx(-3*pi/4)*U
trplot(U_W_1, 'frame','W#1', 'text_opts', {'FontSize',16,
'FontWeight','bold'},...
       'color','c','arrow','length',100,'width', 2 )
U_W_2=trotz(pi/2)*transl(0,t_y,t_z)*trotx(-3*pi/4)*U
trplot(U_W_2, 'frame','W#2', 'text_opts', {'FontSize',16,
'FontWeight','bold'},...
       'color','b','arrow','length',100,'width', 2 )

%axis tight;
axis square
```