The 23rd ECMI Modelling Week
European Student Workshop
on Mathematical Modelling
in Industry and Commerce

Wrocław University of Technology, Wrocław, Poland, August 23-30, 2009

Report of project group 7 on

How to connect two pipes of circular and rectangular profiles?

Project: Connecting two pipes of circular and rectangular profiles

A.A. Arara K. Ermakov M. Friedrich T. Gross
M. Kollmann P. Uść J. Wulcan

October 30, 2009

Contents

1 Introduction

Connecting pipes of different profiles is an important technique in industry. It is essential to design the connecting elements such that they are easy to be made by means of turning down and/or rolling sheet-metal pieces of the appropriate shapes (two-dimensional figures) and then welding them. The welding edge should be situated on some flat surface if possible. The connecting elements should also provide perfect air-tightness.
The main aim of the project is to find and design an element which connects two pipes, one of circular profile and the other one of rectangular profile, with different symmetry axes, see Figure 1.

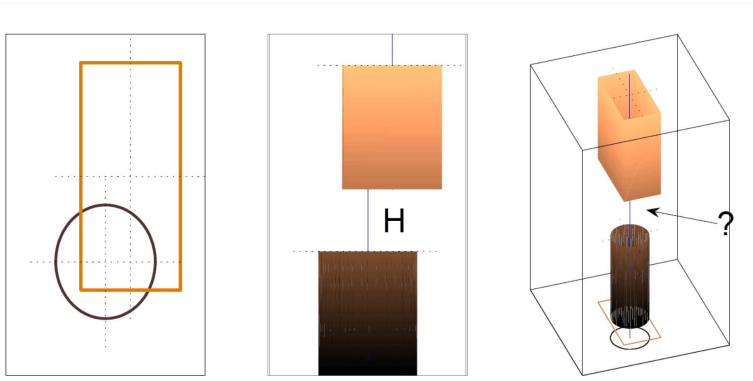

Figure 1: The main problem

Let us begin with considering two simpler, well-known cases: connection of two rectangular pipes with the same symmetry axis, and the case of two circular pipes.

1.1 Connecting two rectangular pipes

Suppose we want to connect two pipes with square profiles with the same symmetry axis. The simplest 3D connecting element is based on trapezoidal shapes. From this one can construct the 2D figure of the unfolded connecting element, see Figure 2.

The first task is to fully describe the geometry of this element with the knowledge of the sizes L and l of the square profiles (with $L > l$) and the

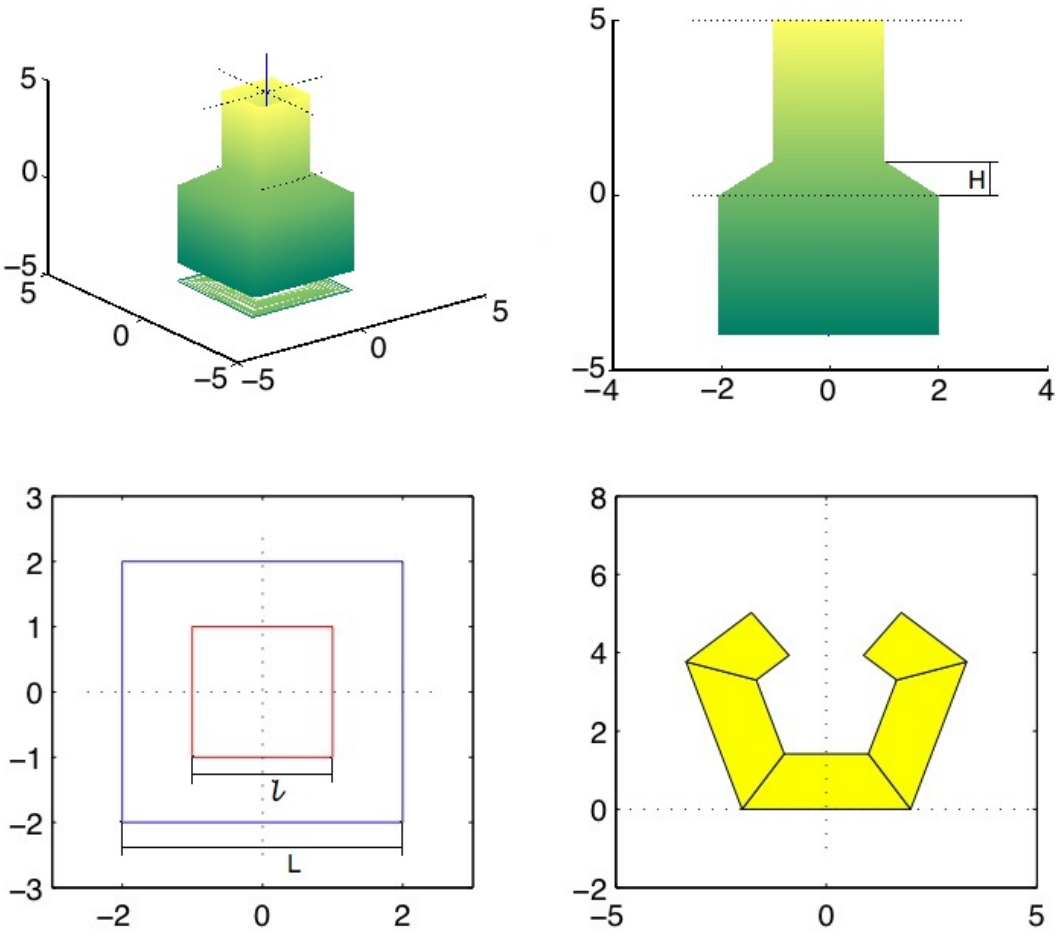

Figure 2: Two square-profile pipes

distance H between the two pipes, see Figure 3 where:
H_1 - height of the trapezoid, that we need to find;
h - length of a side of the trapezoid, which we do not know;
α - angle, which we have to calculate.

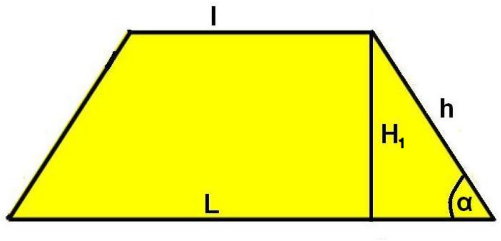

Figure 3: Part of 2D connecting element

For H_1 we have the following expression:

$$H_1 = \sqrt{H^2 + \frac{(L-l)^2}{4}}$$

where H is the distance between the pipes.

From simple high-school geometry we know:

$$\tan(\alpha) = \frac{H_1}{\frac{L-l}{2}}$$

and

$$\sin(\alpha) = \frac{H_1}{h}$$

Now we can easily calculate the angle α and the side h:

$$\alpha = \arctan(\frac{H_1}{\frac{L-l}{2}})$$

$$h = \frac{H_1}{\sin(\alpha)}$$

Using this information, we can describe all 4 parts of the connecting element and, therefore, connect the pipes in the appropriate way.

1.2 Connecting two circular pipes

Now suppose we want to connect two pipes of circular profile with the same symmetry axis. For this problem the connection is based on a cone construction. Again, one can construct the 2-D figure of the unfolded connecting element, see Figure 4.

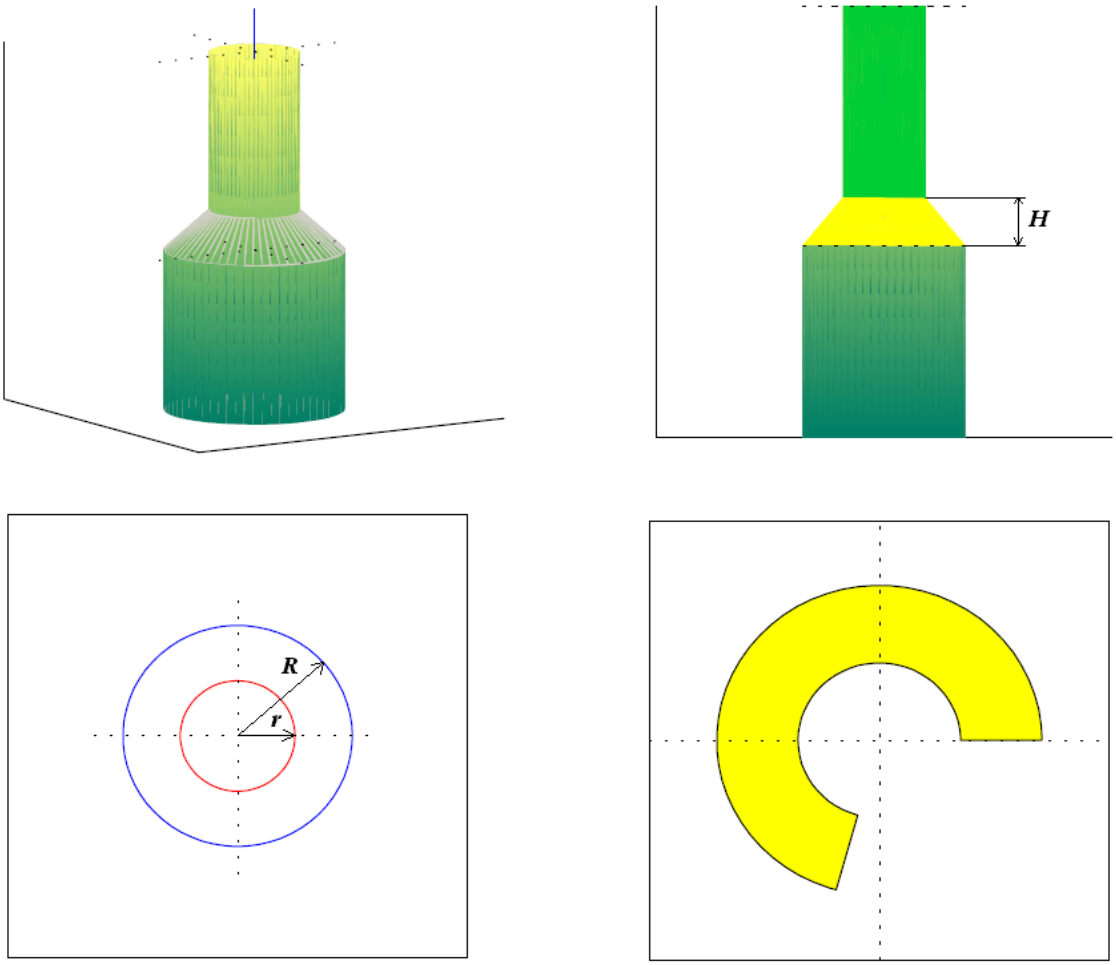

Figure 4: Two circle-profile pipes

Now the second task is to fully describe the geometry of this 2D element with the knowledge of the radiuses R and r of the circle profiles (with $R > r$) and the distance H between the two pipes. For this, consider a regular cone like in Figure 5 where

L - slant height

l - part of the slant height

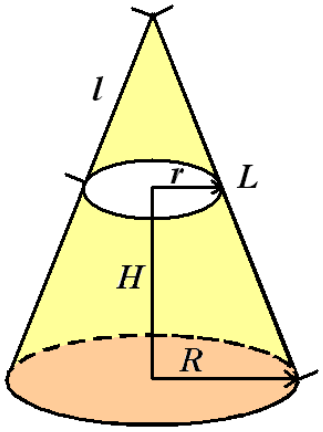

Figure 5: Full cone

Now we have the following relations (similar triangles):

$$\begin{aligned} \frac{L}{R} &= \frac{\sqrt{H^2 + (R-r)^2}}{R-r} \\ \frac{l}{r} &= \frac{\sqrt{H^2 + (R-r)^2}}{R-r} \end{aligned}$$

So we can compute L and l, which we need for constructing 2D shape of the unfolded connecting element, see Figure 6. The only thing left is the angle α. But there exists a simple relation, which we can be obtained from the cone represented at figure 5:

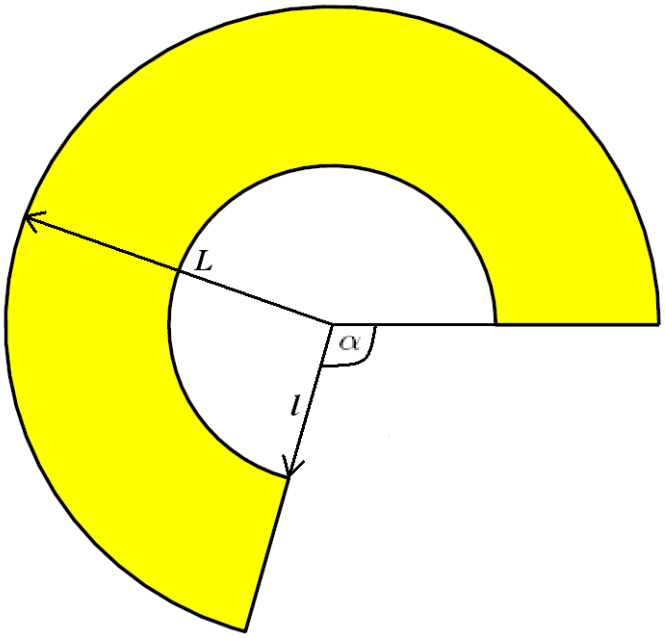

Figure 6: 2D shape of circular connecting element

$$\begin{aligned} \frac{360^\circ - \alpha}{360^\circ} &= \frac{2\pi R}{2\pi L} = \frac{R}{L} \\ \alpha &= 360^\circ(1 - \frac{R}{L}) \end{aligned}$$

So the shape is now fully described, and one can easily build the connecting element using the obtained information.

2 Connecting two pipes of circular and rectangular profiles

2.1 General infromation

Now, let us take into account one rectangular pipe and one circular pipe with different symmetry axes. The main aim of the project is to design a connection element for this problem.

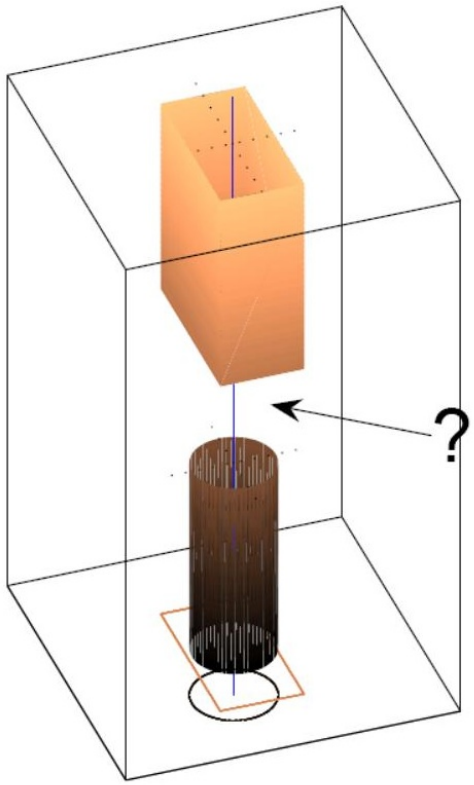

Figure 7: 3D model of pipes

Denote by r, a, b, h and (x, y) the radius of the circular pipe, the length and the width of the rectangular pipe, the distance between the pipes and the coordinates of the centre of the circular pipe, respectively.

To understand the problem better it is useful to observe the 3D model from the top, see Figure 8.

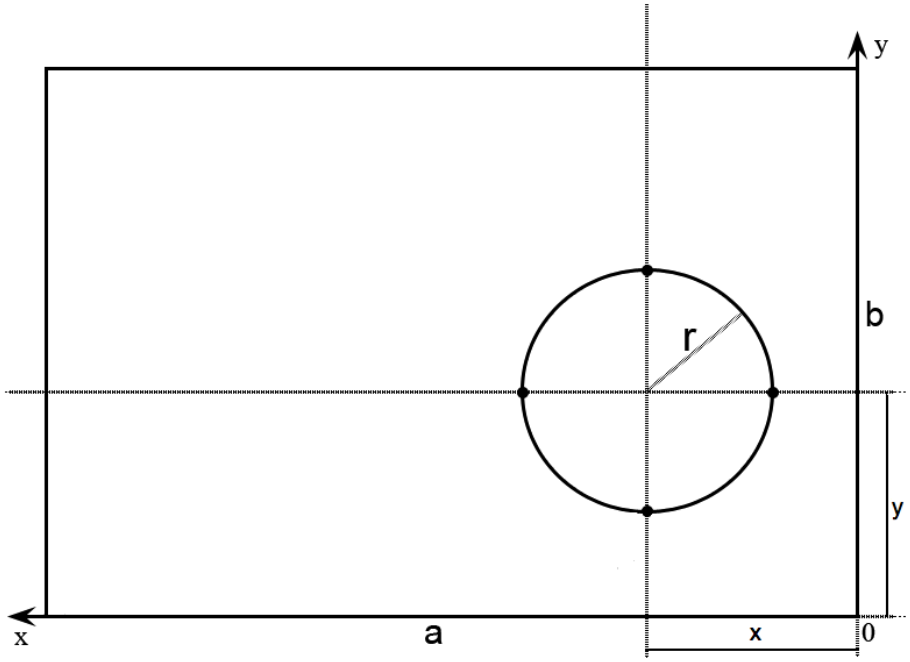

Figure 8: View from the top

From Figure 9 we can see that the 3D connecting model should be made of 4 triangles connected to each other with 4 yet unknown regions. We propose to define these regions as parts of surfaces of the appropriate, asymmetric cones.

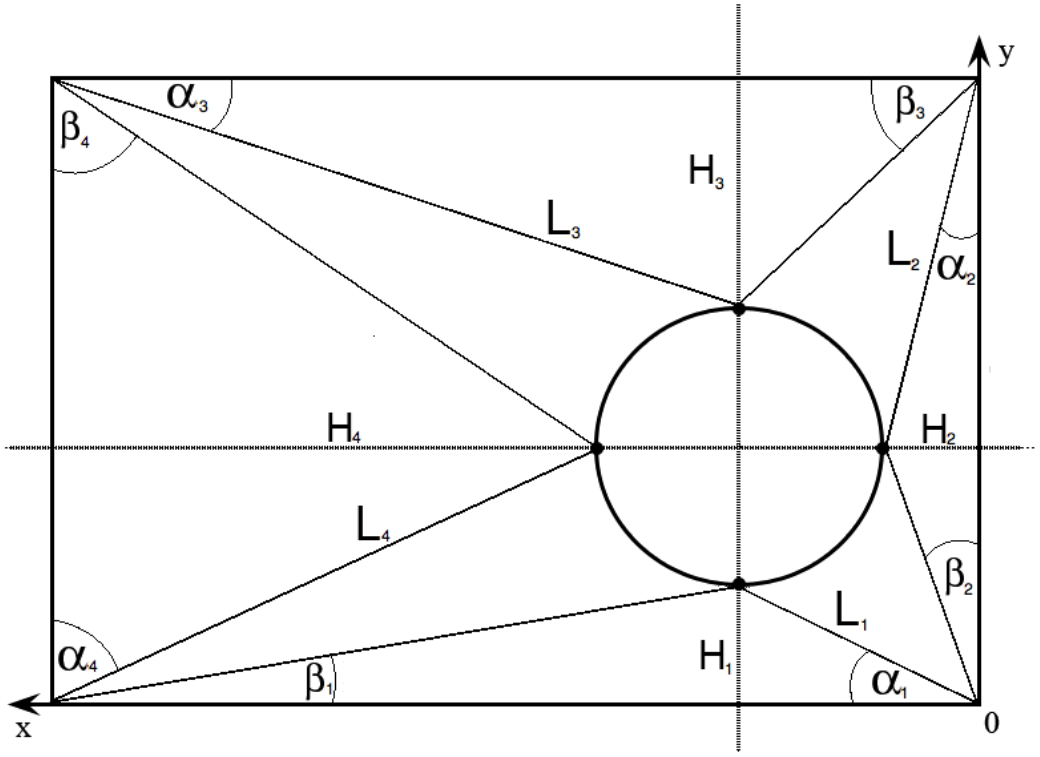

Figure 9: The structure of connecting model

As it can be seen in Figure 10, each of these cones has a base with radius r (the radius of the circle) and a vertex, which coincides with one of the vertices of the rectangular. The line l is the line on the surface of the cone going from the vertex to the base.

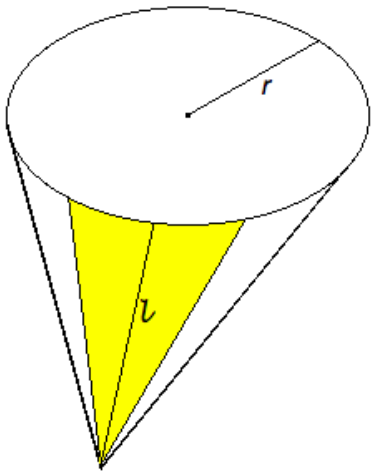

Figure 10: Asymmetric cone

In Figure 11 we can see that the cone regions can be approximately defined by the lines l on the surfaces of each cone. The more lines l we find, the more accurate the approximation will be.

To define the surface, one needs to find the lengths of the lines and the angles between them:

- To find the lengths of all the lines in the unknown region we shall consider cylindrical coordinate system in 3D model with the centre situated in the centre of the circle. Then we can calculate the cylindrical coordinates of the points in the circle and of the vertices of the triangle.
- To find the angles between the lines we need to consider cones with the peaks in the vertices of rectangular with the circle as the base.

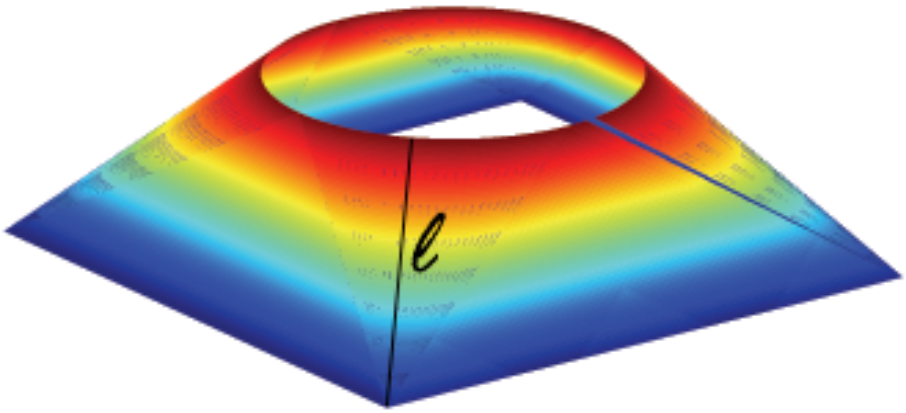

Figure 11: Connecting element

2.2 Calculation of triangles

Consider the rectangular with the length a and the width b. The radius of the circle is r and its coordinates in 3D are (x, y, h).
As it was explained in the previous subsection, the 3D model can be divided in the 4 triangles and 4 parts of unknown shape. To find these triangles we have to calculate their sides and angles between them.

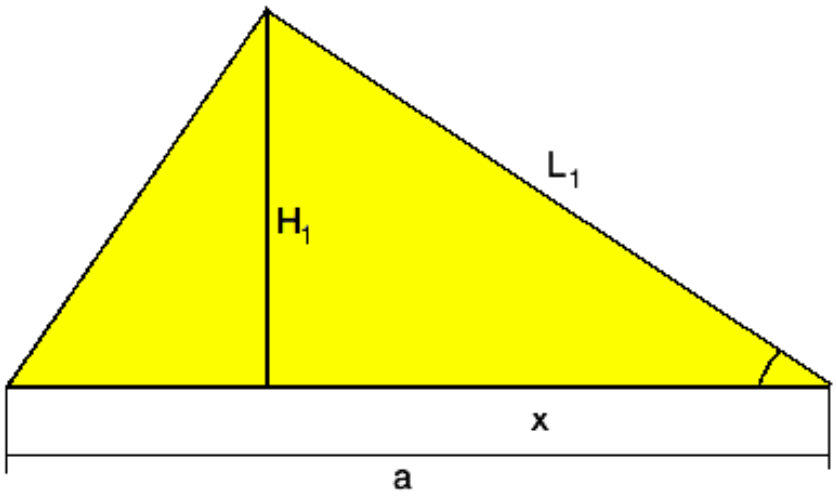

Figure 12: Triangle part of connecting element

To find H_1 we should have a look at this side from the different point.

From Figures 12 and 13 it is obvious that H_1 can be found from the following expression:

$$H_1 = \sqrt{h^2 + (y - r)^2}$$

Then:

$$L_1 = \sqrt{h^2 + (y - r)^2 + (a - x)^2},$$

$$\alpha_1 = \arcsin\left(\frac{H_1}{L_1}\right)$$

and

$$\beta_1 = \arctan\left(\frac{H_1}{a - x}\right)$$

where α_1 is the right angle of the triangle and β_1 is the left angle of the triangle.

Now we can construct the first triangle in 2D plane. All other triangles can be found in the similar way. We will just provide formulas for H_i, L_i, α_i

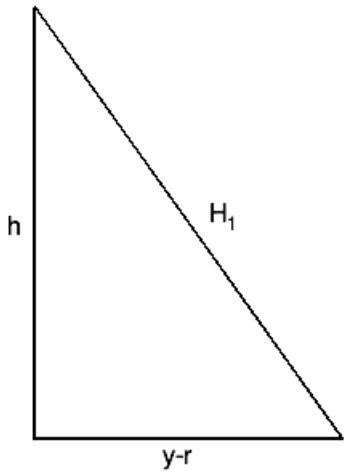

Figure 13: Triangle to find H_1

and β_i.

$$H_2 = \sqrt{h^2 + (a - x - r)^2}$$

$$L_2 = \sqrt{h^2 + (b - y)^2 + (a - x - r)^2}$$

$$\alpha_2 = \arcsin\left(\frac{H_2}{L_2}\right)$$

$$\beta_2 = \arctan\left(\frac{H_2}{y}\right)$$

$$H_3 = \sqrt{h^2 + (b - y - r)^2}$$

$$L_3 = \sqrt{h^2 + x^2 + (b - y - r)^2}$$

$$\alpha_3 = \arcsin\left(\frac{H_3}{L_3}\right)$$

$$\beta_3 = \arctan\left(\frac{H_3}{x}\right)$$

$$H_4 = \sqrt{h^2 + (x - r)^2}$$

$$L_4 = \sqrt{h^2 + y^2 + (x - r)^2}$$

$$\alpha_4 = \arcsin\left(\frac{H_4}{L_4}\right)$$

$$\beta_4 = \arctan\left(\frac{H_4}{b-y}\right)$$

The only unknowns now are the angles between the triangles and the shape of non-triangular parts.

2.3 Calculation of non-triangular parts

For simplicity, we first consider the symmetric case, i.e. when the centre of the circle coincides with the centre of the rectangular.

As it was briefly mentioned above, we use cylindrical coordinates to calculate the shape of non-triangular parts. The idea is the following:

- The centre of the circle is taken as the centre of coordinates.
- All other points of the circle then have the following coordinates: $(r, \phi, 0)$, where r - radius of the circle and $\phi \in [0, \frac{\pi}{2}]$.
- The peak point of the rectangular has the following coordinates:

$$\left(\frac{\sqrt{a^2+b^2}}{2}, \arctan\left(\frac{b}{a}\right), h\right)$$

Then the distance between the peak and the circle point is:

$$l = \sqrt{\left(\frac{\sqrt{a^2+b^2}}{2}\cos\arctan\left(\frac{b}{a}\right) - r\cos\phi\right)^2 + \left(\frac{\sqrt{a^2+b^2}}{2}\sin\arctan\left(\frac{b}{a}\right) - r\sin\phi\right)^2 + h^2}$$

Now, let us return to the more general problem, i.e. the centre of the circle has 3D coordinates (x, y, h).

For this case we use the following approach to simplify the calculations. For each of the 4 sides of the rectangular we construct new virtual rectangulars to calculate lines l. The idea behind that is the following - the centre of the circle appears to be the centre of the new virtual rectangulars, so we can use all the formulas described above. Below, we provide the lengths (a_i) and the widths (b_i) of the new rectangulars:

- $a_1 = 2x$
  $b_1 = 2y$
- $a_2 = 2(b - y)$
  $b_2 = 2x$
- $a_3 = 2(a - x)$
  $b_3 = 2(b - y)$
- $a_4 = 2y$
  $b_4 = 2(a - x)$

The last open question is the angles between the two successive lines. We are going to find the approximate values for these angles using the cones with the peaks in the vertices of the rectangular and the base in circle.

If we use cylindrical coordinates again and choose very small angle ϕ, then we can find the coordinates of the point on the circle $(r, \phi, 0)$ which is very close to the point $(r, 0, 0)$. Obviously, we can find the length of the corresponding segment of the circle, which is $\Delta l = \phi r$. But since this segment is very small, it is also equal to $\Delta l = \psi l$, where l can be found using the formula above, and ψ is the required angle between the lines in non-triangular part.

Then we have

$$\psi_i = \phi_i \frac{r}{l},$$

for every small $\phi_i \in [0, \frac{\pi}{2}]$.

The final step is to find the angle γ_i between the triangles, which is simply found from:

$$\gamma_i = \pi - \alpha_i - \sum_j \psi_{ij} - \beta_{i+1},$$

for $i = 1, 2, 3, 4$.

Using the information above we can construct each of the triangle regions of connecting element, approximately construct non-triangular parts between these triangles and the angles between the neighbouring triangle parts. Obviously, this information is sufficient to construct the whole connecting element, which was done in Matlab (see appendix **pipes.m**).

Now we are going to present some examples of unfolded elements, obtained in Matlab.

1. Circular pipe has radius 1, coordinates $(2, 2)$. Rectangular pipe has length and width equal to 4. The distance between pipes is 2. So the centre of the circular pipe coincides with the centre of rectangular pipe.

   In Matlab this example can be called with the following function:

   ```
   pipes(2, 2, 1, 4, 4, 2, 100)
   ```

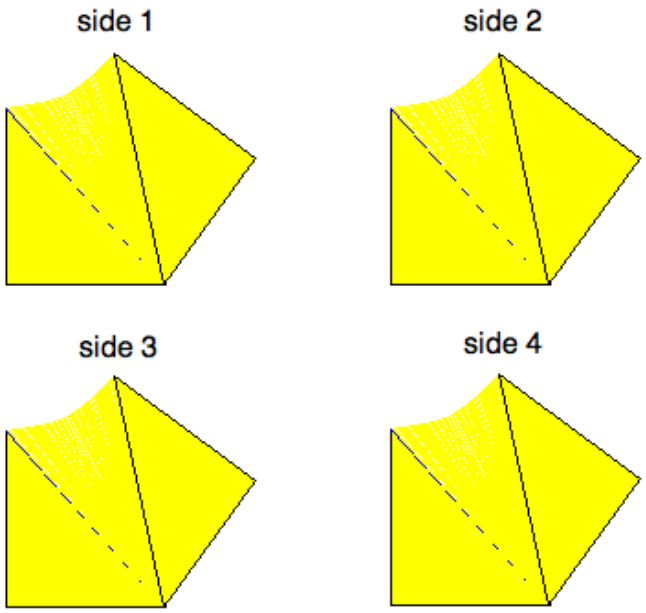

Figure 14: Example 1

2. Circular pipe has radius 1, coordinates $(2, 2)$. Rectangular pipe has length 8 and width 6. The distance between the pipes is 4.

   In Matlab this example can be called with the following function:

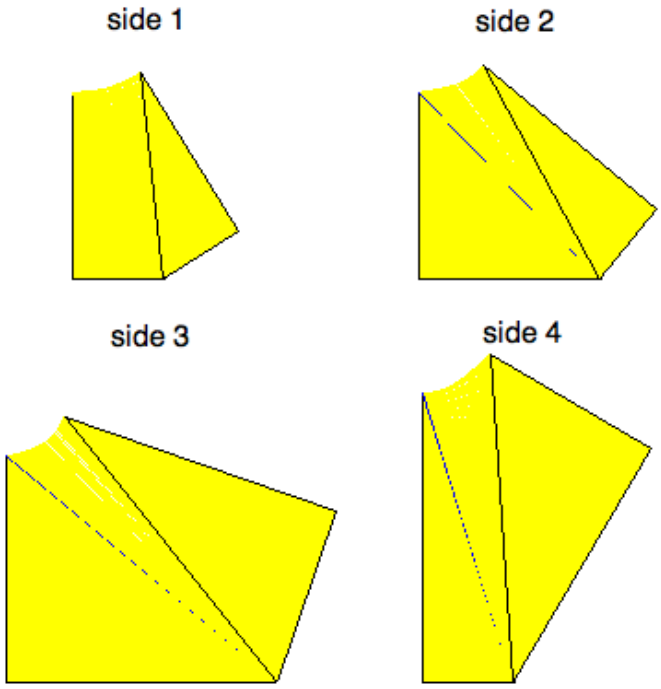

Figure 15: Example 2

```
pipes(2, 2, 1, 8, 6, 4, 100)
```

3. Circular pipe has radius 2, coordinates $(3, 3)$. Rectangular pipe has length 8 and width 6. The distance between the pipes is 4.

   In Matlab this example can be called with the following function:

   ```
   pipes(3, 3, 2, 8, 6, 4, 100)
   ```

4. In this example we will take the radius of the circular pipe bigger than the width and the length of the rectangular pipe. The coordinates of the circular pipe are $(3, 3)$, radius is 6. The length and the width of the rectangular pipe are equal to 4. The distance between the pipes is 3.

   In Matlab this example can be called with the following function:

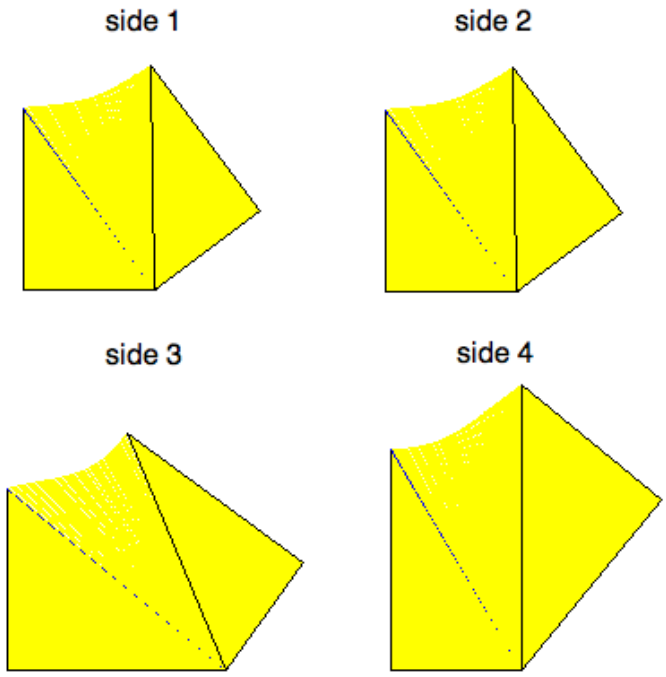

Figure 16: Example 3

```
pipes(3, 3, 6, 4, 4, 3, 100)
```

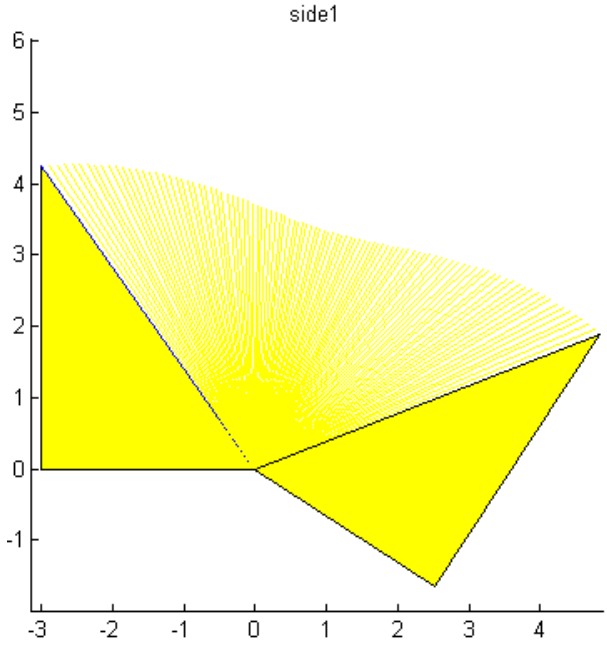

Figure 17: Example 4: side 1

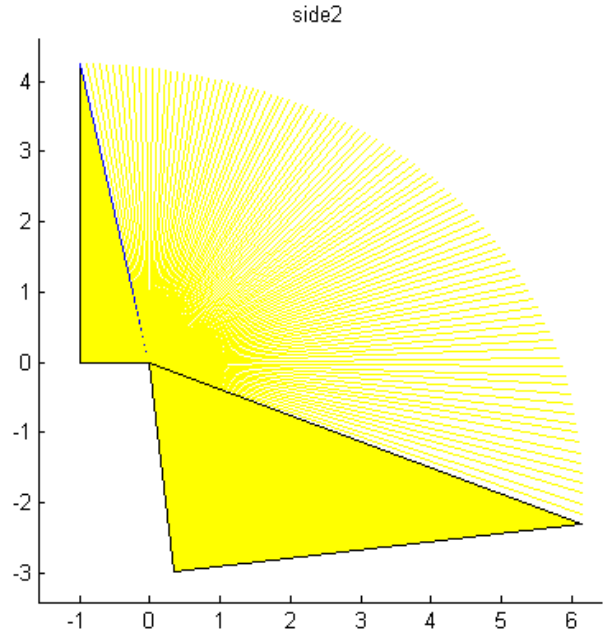

Figure 18: Example 4: side 2

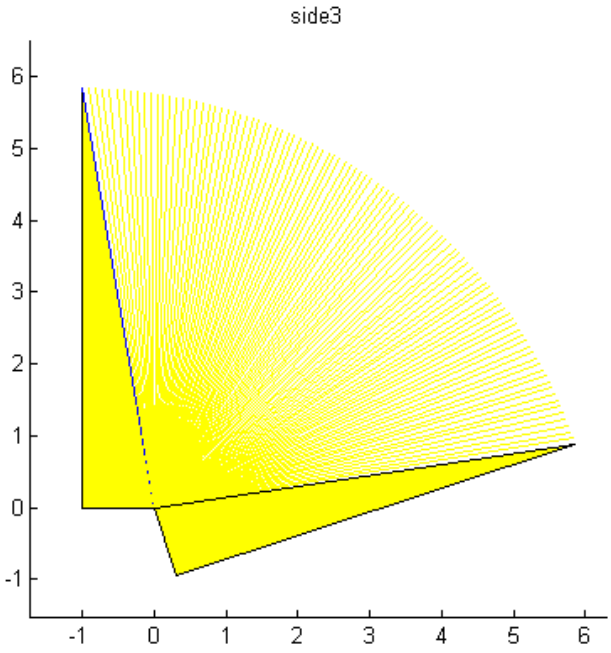

Figure 19: Example 4: side 3

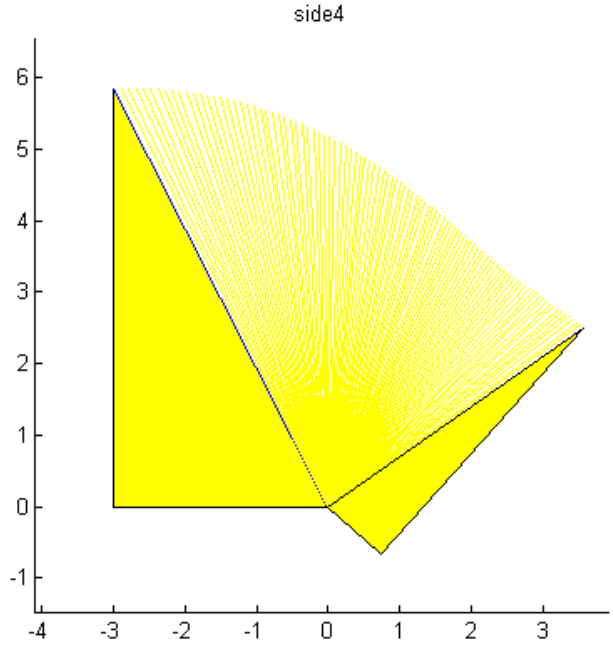

Figure 20: Example 4: side 4

It is difficult to estimate accuracy of this method, as it depends on many factors. All we can say is that, the more lines l we calculate, the more precise the solution will be.

In the next section an alternative solution to the problem will be presented. It allows to solve the problem for more general shapes, not only rectangular and circular.

3 Generalized problem

The generalized problem was not initially the part of the project, but the idea we used to solve circular-rectangular problem could be also used to solve the problem for any convex shapes. That is why we extended the initial problem.

The solution to the generalized problem is divided into three steps.

3.1 Step 1: Find the 3D shape

To find the 3D shape we used the convex hull of the two polygons. We then removed the faces in the polygons making openings for pipes. There exist convex hull algorithms for 3 dimensions with complexity $O(n \log n)$. The convex hull algorithm gives us a set of faces F (that is triangles) that form the 3D shape.
For the simple rectangular-rectangular case this is presented in the Figure 21

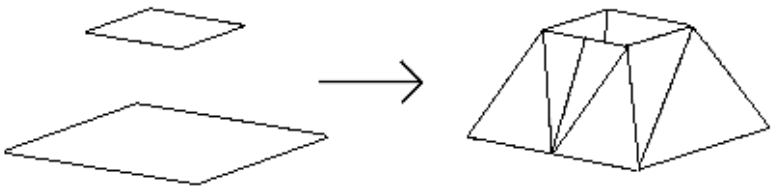

Figure 21: 3D Shape

3.2 Step 2: Make an indexation of the faces

To be able to unfold the 3D shape we need to make an indexation of the faces. Step 1 will give us a set of faces F. We need to find an indexation $I : F \to N$ such that if $|I(f) - I(g)| = 1$ then the faces f and g share an edge.

Right now our implementation of this uses $O(n^2)$ time. But this could probably very easy be improved to someting like $O(n \log n)$.

3.3 Step 3: Unfold the shape

To unfold the shape, start of by placing the first face $I^{-1}(0)$ in a 2D plane in an arbitrary way, but of course isometric.

Then place the next face using the fact that it shares an edge with a face that has already been placed. This gives us that two of the three points in the face has already been placed, the third point can be found since we know the distances inside the face and that the 2D shape must not overlap itself.

The last step can be repeated which gives us the whole 2D shape. The complexity for this step is $O(n)$.

Example for rectangular-rectangular case is presented in Figures 22-24:

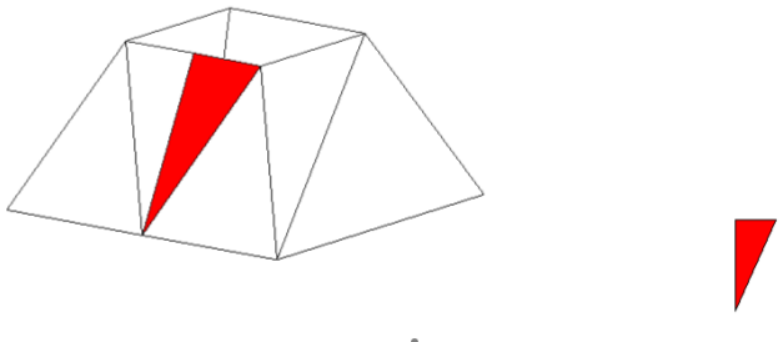

Figure 22: First triangle

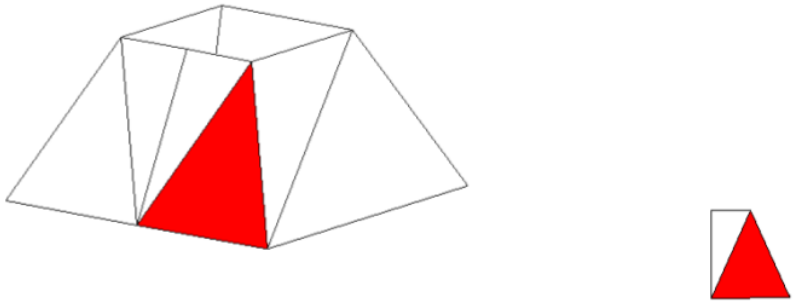

Figure 23: Second triangle

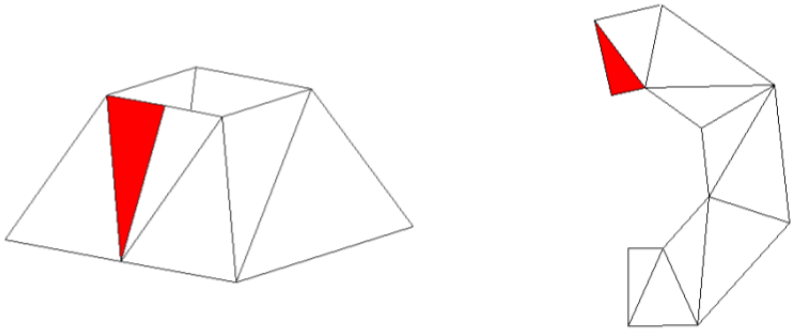

Figure 24: Last triangle

As it was already mentioned, this method allows to solve the problem for the great variety of shapes, even for some extraordinary ones.

4 Conclusions

In this report we discussed the problem of finding an element for connecting two pipes. Our main aim was to find the connection for the pipes of circular and rectangular shape. The solution of the problem should also be real, i.e. the connecting element should be easy to produce. We tried to solve this problem with two different approaches, and both of them produced the solution which looked very similar.

The first solution is for the general, but usual case, i.e. connecting a pipe of a rectangular profile with another one of a circular profile. The problem was successfully solved, and matlab program was created to calculate the connecting element for various initial data.

The second solution is even more general - the shapes of the pipes can be arbitrary (but convex). Obviously, this approach can be used for the problems described before as well. But it also solves some other very interesting problems - the shapes of the pipes can be triangles, trapezoids and even hearts.

So we solved the problem for even more general case than was requested from the industries with 2 completely different approaches, but in the end we had similar results.

However, we still have some open questions:

- One of these is whether the 2D shape can overlap when we unfold the 3D shape. It never happened, even when we tried to connect some very extraordinary convex pipes, but we have no explanation for this.
- The second question is how we can connect concave shapes. This is obviously not the demand of the industries, but the solution could be interesting.
- Another question is, whether our solution is the optimal one. Especially when we look at the flow of liquid in the pipes. Our approach is in this context probably not bad, but maybe another shape of the connection element would cause a better behavior of the flow.

In the end one can say that our solutions can not be so far away from the optimal one because two our different approaches led to very similar solutions.

5 Appendix

5.1 Code for the initial problem

5.1.1 pipes.m

```
function [] = pipes(circx, circy, rad, length, width, dist,
number_of_phi)

% How to use this program:
%
% circx, circy = x- and y-coordinate of the center of the circle from the
% lower left edge of the rectangular
%
% rad = radius of the circle
% length, with = lenght and width of the rectangular
%
% dist = the distance between the two pipes
%
% number_of_phi = the number of small pieces in that the angle between the
%
% triangles will be devided to aproximate it.

n = number_of_phi;
h1 = sqrt(dist^2 + (circy - rad)^2);     %height 1st triangle
h2 = sqrt(dist^2 + (circx - rad)^2);     %height 2nd triangle
h3 = sqrt(dist^2 + (width - circy - rad)^2);     %height 3rd triangle
h4 = sqrt(dist^2 + (length - circx - rad)^2);     %height 4th triangle
% the side of the 1st triangle:
a1 = sqrt(dist^2 + (circx)^2 + (circy - rad)^2);
% the side of the 2nd triangle:
a2 = sqrt(dist^2 + (width - circy)^2 + (circx - rad)^2);
% the side of the 3rd triangle:
a3 = sqrt(dist^2 + (length - circx)^2 + (width - circy - rad)^2);
% the side of the 4th triangle:
a4 = sqrt(dist^2 + circy^2 + (length - circx - rad)^2);
```

```
length_save = length;
width_save = width;

%side1
length = abs(2*circx);  %choice of the new, ...
width = 2*circy;        %imaginary rectangle
alpha1 = atan((h1*2)/length); %lower right angle of the 1st triangle

step = pi/(2*(n-1));
phi = 0;
r = rad;

figure;
title('side1');
axis ([-length_save width_save 0 length_save+width_save]);
axis square;
x =[-length/2, -length/2,0];
y = [0,h1,0];
hold on;
fill(x,y,'b');  %drawing the 1st triangle

l1 = sqrt(((sqrt(length^2+width^2)/2)*(cos(atan(length/width))) - ...
    r*cos(phi))^2 + ((sqrt(length^2+width^2)/2)*(sin(atan(length/width)))...
    - r*sin(phi))^2 + dist^2);

%draw the n lines which describe the angle alpha:
for p=1:n
    dp(p) = phi;

    l = sqrt(((sqrt(length^2+width^2)/2)*(cos(atan(length/width))) - ...
        r*cos(phi))^2 + ((sqrt(length^2+width^2)/2)*...
        (sin(atan(length/width))) - r*sin(phi))^2 + dist^2);
    da(p) = (phi*r)/l1;
    dh = sin(alpha1 + da(p))*l;
    dl = -cos(alpha1 + da(p))*l;

    x = [dl,0];
    y = [dh,0];
```

```
    plot(x,y,'r');
    phi = phi + step;
end

%draw the second triangle:
beta1 = atan(h2/circy);
gamma = pi - alpha1 - beta1 - da(n);

c = sqrt(dh^2+dl^2-h2^2);
xc = cos(gamma)*c;
yc = sin(gamma)*c;
x = [0, dl, xc];
y = [0, dh, yc];

fill(x,y,'b');


%For the sides 2,3,4 we do the same as for the 1st:
%side2
length = 2*(width_save - circy);
width = abs(2*circx);
alpha2 = atan((h2*2)/length);

step = pi/(2*(n-1));
phi = 0;
r = rad;

figure;
title('side2');
axis ([-width_save length_save 0 length_save+width_save]);
axis square;
hold on;
x =[-length/2, -length/2, 0];
y = [0,h2,0];


fill(x,y,'b');
l1 = sqrt(((sqrt(length^2+width^2)/2)*(cos(atan(length/width))) - ...
```

```
    r*cos(phi))^2 + ((sqrt(length^2+width^2)/2)*(sin(atan(length/width)))...
    - r*sin(phi))^2 + dist^2);
for p=1:n
    dp(p) = phi;

    l = sqrt(((sqrt(length^2+width^2)/2)*(cos(atan(length/width))) -...
        r*cos(phi))^2 + ((sqrt(length^2+width^2)/2)*(sin(atan(length/width)))...
        - r*sin(phi))^2 + dist^2);
    da(p) = (phi*r)/l1;
    dh = sin(alpha2 + da(p))*l;
    dl = -cos(alpha2 + da(p))*l;

    x = [dl,0];
    y = [dh,0];
    plot(x,y,'r');
    phi = phi + step;
end

beta2 = atan(h3/circx);
gamma = pi - alpha2 - beta2 - da(n);

c = sqrt(dh^2+dl^2-h3^2);
xc = cos(gamma)*c;
yc = sin(gamma)*c;
x = [0, dl, xc];
y = [0, dh, yc];

fill(x,y,'b');


%side3
length = 2*(length_save-circx);
width = abs(2*(width_save - circy));
alpha3 = atan((h3*2)/length);

step = pi/(2*(n-1));
phi = 0;
r = rad;
```

```
figure;
title('side3');
axis ([-length_save width_save 0 length_save+width_save]);
axis square;
x =[-(length_save-circx), -(length_save-circx),0];
y = [0,h3,0];

hold on;
fill(x,y,'b');
l1 = sqrt(((sqrt(length^2+width^2)/2)*(cos(atan(length/width))) - ...
    r*cos(phi))^2 + ((sqrt(length^2+width^2)/2)*(sin(atan(length/width))) ...
    - r*sin(phi))^2 + dist^2);
for p=1:n
    dp(p) = phi;

    l = sqrt(((sqrt(length^2+width^2)/2)*(cos(atan(length/width))) -...
        r*cos(phi))^2 + ((sqrt(length^2+width^2)/2)*(sin(atan(length/width)))...
        - r*sin(phi))^2 + dist^2);
    da(p) = (phi*r)/l1;
    dh = sin(alpha3 + da(p))*l;
    dl = -cos(alpha3 + da(p))*l;

    x = [dl,0];
    y = [dh,0];
    plot(x,y,'r');
    phi = phi + step;
end

beta3 = atan(h4/(width_save-circy));
gamma = pi - alpha3 - beta3 - da(n);

c = sqrt(dh^2+dl^2-h4^2);
xc = cos(gamma)*c;
yc = sin(gamma)*c;
x = [0, dl, xc];
y = [0, dh, yc];
```

```
fill(x,y,'b');


%side4
length = 2*circy;
width = 2*(length_save-circx);
alpha4 = atan((h4*2)/length);

step = pi/(2*(n-1));
phi = 0;
r = rad;

figure;
title('side4');
axis ([-width_save length_save 0 length_save+width_save]);
axis square;
hold on;
x =[-circy, -circy,0];
y = [0,h4,0];

fill(x,y,'b');
l1 = sqrt(((sqrt(length^2+width^2)/2)*(cos(atan(length/width))) - ...
    r*cos(phi))^2 + ((sqrt(length^2+width^2)/2)*(sin(atan(length/width)))...
    - r*sin(phi))^2 + dist^2);
for p=1:n
    dp(p) = phi;

    l = sqrt(((sqrt(length^2+width^2)/2)*(cos(atan(length/width))) - ...
        r*cos(phi))^2 + ((sqrt(length^2+width^2)/2)*(sin(atan(length/width)))...
        - r*sin(phi))^2 + dist^2);
    da(p) = (phi*r)/l1;
    dh = sin(alpha4 + da(p))*l;
    dl = -cos(alpha4 + da(p))*l;

    x = [dl,0];
    y = [dh,0];
    plot(x,y,'r');
    phi = phi + step;
```

```
end

beta4 = atan(h1/(length_save - circx));
gamma = pi - alpha4 - beta4 - da(n);

c = sqrt(dh^2+dl^2-h1^2);
xc = cos(gamma)*c;
yc = sin(gamma)*c;
x = [0, dl, xc];
y = [0, dh, yc];

fill(x,y,'b');
hold off;
```

5.2 Code for genaralized problem

5.2.1 gen_paint.m

```
function [ p ] = gen_paint( ax )
figure('Name','Paint a shape','NumberTitle','off')
axis(ax);
title('Paint a shape');
h = impoly;
pos = wait(h);
p = pos';
end
```

5.2.2 gen_square.m

```
function [ p ] = gen_rect( mx, my, w, h )
p = [ [0 h/2+h/1000]' [w/2 h/2]' [w/2 -h/2]' [-w/2 -h/2]' [-w/2 h/2]'];
% + h/1000 is just to make the polygon convex
p = [p(1, :)+mx;p(2, :)+my];
end
```

5.2.3 gen_circle.m

```
function [ p ] = gen_circle( r, mx, my, pres)
    phi = [0:2*pi/pres:2*pi-2*pi/pres];
    p = [ sin(phi).*r+mx; cos(phi).*r+my];
end
```

5.2.4 shape2p3dv2.m

```
function [ p3d, p, faces ] = shape2p3dv2( shape1, shape2, dist )

p = [ [shape1;shape1(1, :).*0] [shape2;shape2(1, :).*0+dist] ];

faces_raw = convhulln(p')';
faces = [];
for i=1:size(faces_raw, 2)
    if all(p(3, faces_raw(:,i))==0)
        continue
```

```
    end
    if all(p(3, faces_raw(:,i))==dist)
        continue
    end
    faces = [faces faces_raw(:,i)];
end
used_faces = [];
curr1 = 1;
curr2 = size(shape1, 2)+1;
end1 = 1;
end2 = size(shape1, 2)+1;
p3d = [ [p(:, curr1);0;0] [p(:, curr2);0;0] ];
p3di1=1;
p3di2=2;
curr_p3di=3;
while 1
    % find next point
    next_p = -1;
    for i=1:size(faces, 2)
        if any(used_faces==i)
            continue
        end
        if all([any(faces(:, i)==curr1) any(faces(:, i)==curr2)])
            f = find(faces(:, i)~=curr1 & faces(:, i)~=curr2);
            next_p = faces(f, i);
            p3d = [p3d [p(:, next_p); p3di1; p3di2]];
            if p(3, next_p) == 0
                % shape1
                p3di1=curr_p3di;
                curr1 = next_p;
            else
                % shape2
                p3di2=curr_p3di;
                curr2 = next_p;
            end
            used_faces = [used_faces i];
            curr_p3di=curr_p3di+1;
            break;
```

```
        end
    end
    if next_p == -1
        error('Couldnt build p3d data')
    end

    if all([curr1==end1 curr2==end2])
        break
    end
end

end
```

5.2.5 p3d2ddate.m

```
function [ p d ] = p3d2ddata( p3d )


% generate ddata
v_first = p3d([1,2,3], 1)-p3d([1,2,3], 2);
d_first = sqrt(v_first'*v_first);
p = [ [0 0]' [0 d_first ]' ];
d = [];
for i=3:size(p3d, 2)
    vect1 = p3d([1,2,3], i)-p3d([1,2,3], p3d(4, i));
    dist1 = sqrt(vect1'*vect1);
    vect2 = p3d([1,2,3], i)-p3d([1,2,3], p3d(5, i));
    dist2 = sqrt(vect2'*vect2);
    d = [d [p3d(4,i) dist1 p3d(5, i) dist2]'];
end

end
```

5.2.6 ddata2screen.m

```
function ddata2screen( p, d )
%p = [ [0 0]' [0 3]' ];
%d = [ [1 4 2 5]' [1 5 3 3]' ];
```

```
figure('Name','2d plot','NumberTitle','off')
hold on;
for i=1:size(d, 2)
    p1 = p(:, d(1,i));
    p2 = p(:, d(3,i));

    qq = p2-p1;
    a = sqrt(qq'*qq);
    b = d(2, i);
    c = d(4, i);

    alpha = acos( (a^2+b^2-c^2)/(2*a*b) );
    beta = acos ( qq(2)/a );
    if qq(1)<0
        beta=-beta;
    end
    new_v = [sin(alpha+beta);cos(alpha+beta)]*b;
    p3 = p1+new_v;

    if sqrt((p1-p3)'*(p1-p3))-b > 0.01
        fel = i
    end
    if sqrt((p2-p3)'*(p2-p3))-c > 0.01
        fel2 = i
    end

    p = [p p3];
    points = [p1 p2 p3];
    fill(points(1, :), points(2, :), 'r');
end
hold off;
axis equal;
axis off;
end
```

5.2.7 pipe.m

```
function [ ] = pipe( shape1, shape2, dist )
```

```
if size(convhulln(shape1'), 1) ~= size(shape1, 2)
    error('shape1 isnt convex');
end
if size(convhulln(shape2'), 1) ~= size(shape2, 2)
    error('shape2 isnt convex');
end

[p3d, points, faces] = shape2p3dv2(shape1, shape2, dist);
[p d] = p3d2ddata(p3d);

figure('Name','Shape1','NumberTitle','off')
patch(shape1(1, :), shape1(2, :), 'w');
axis equal;
axis off;

figure('Name','Shape2','NumberTitle','off')
patch(shape2(1, :), shape2(2, :), 'w');
axis equal;
axis off;

figure('Name','3d plot','NumberTitle','off')
trisurf(faces',points(1,:),points(2,:),points(3,:))
axis equal;
axis off;

ddata2screen(p, d);

end
```