SIS Scuola Interdipartimentale delle **Scienze**, dell'**Ingegneria** e della **Salute**

DiST Dipartimento di Scienze e Tecnologie

L. Magistrale in IA (ML&BD)

Scientific Computing (part 2 – 6 credits)

prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Bldg. C4
room: n. 423 – North Side, 4th floor
phone: 081 547 6545
email: mariarosaria.rizzardi@uniparthenope.it

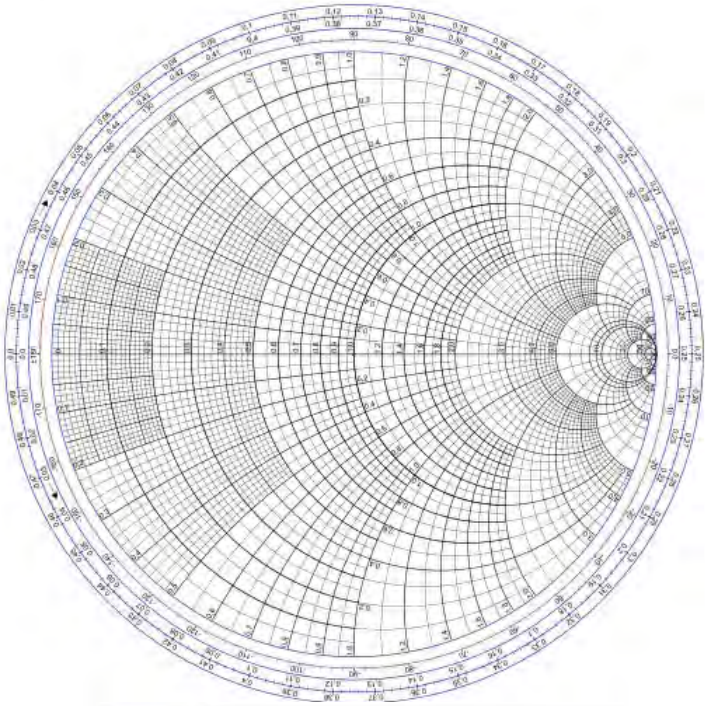
Contents

- **Applications of Moebius mappings.**
- **Joukowski mappings.**

Some applications of Moebius Mappings

Smith chart

Application to radio antennas
(electrical/electronic engineering)

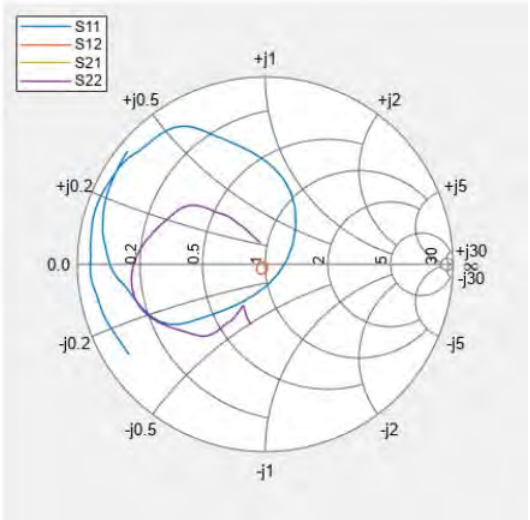


The function

$$f(z) = \frac{z-1}{z+1}$$

maps reflection coefficients to normalized impedance. The Smith chart is commonly used to display the relationship between a reflection coefficient, typically given as S11 or S22, and a normalized impedance.

https://en.wikipedia.org/wiki/Smith_chart



The MATLAB **smithplot()** function, in RF Toolbox, plots measurement data on a Smith Chart.

Some applications of Moebius Mappings

Connection with Einstein's Theory of Special Relativity

In Einstein's Theory of Special Relativity the time T and the 3D Cartesian coordinates of an event are combined into a single vector (T,X,Y,Z) of the 4D space-time.

Einstein's Theory tells us that if two (momentarily coincident) observers are in relative motion, they will disagree about the times at which events occur, but Einstein discovered that both observers will agree on the value of

$$T^2 - (X^2 + Y^2 + Z^2)$$

A Lorentz transformation is a linear transformation of space-time (a 4×4 matrix) that maps one observer's description of an event to another observer's description of the same event, and it preserves the above quantity.

If a point P in space emits a flash of light, then each of the light rays could be represented by a complex number. By choosing units of space and time so that the speed of light is 1, after one unit of time, the expanding sphere of light emitted by P (made up of particles of light called *photons*) form a unit sphere.Thus each photon may be identified with a point on the Riemann sphere, and, via stereographic projection, with a complex number.

The complex mappings that correspond to the Lorentz transformations are the Moebius transformations, and, conversely, every Moebius transformation of $\mathbb{C}$ yeld a unique Lorents transformation of space-time.

Application of T_M: example 1 (touching circles)

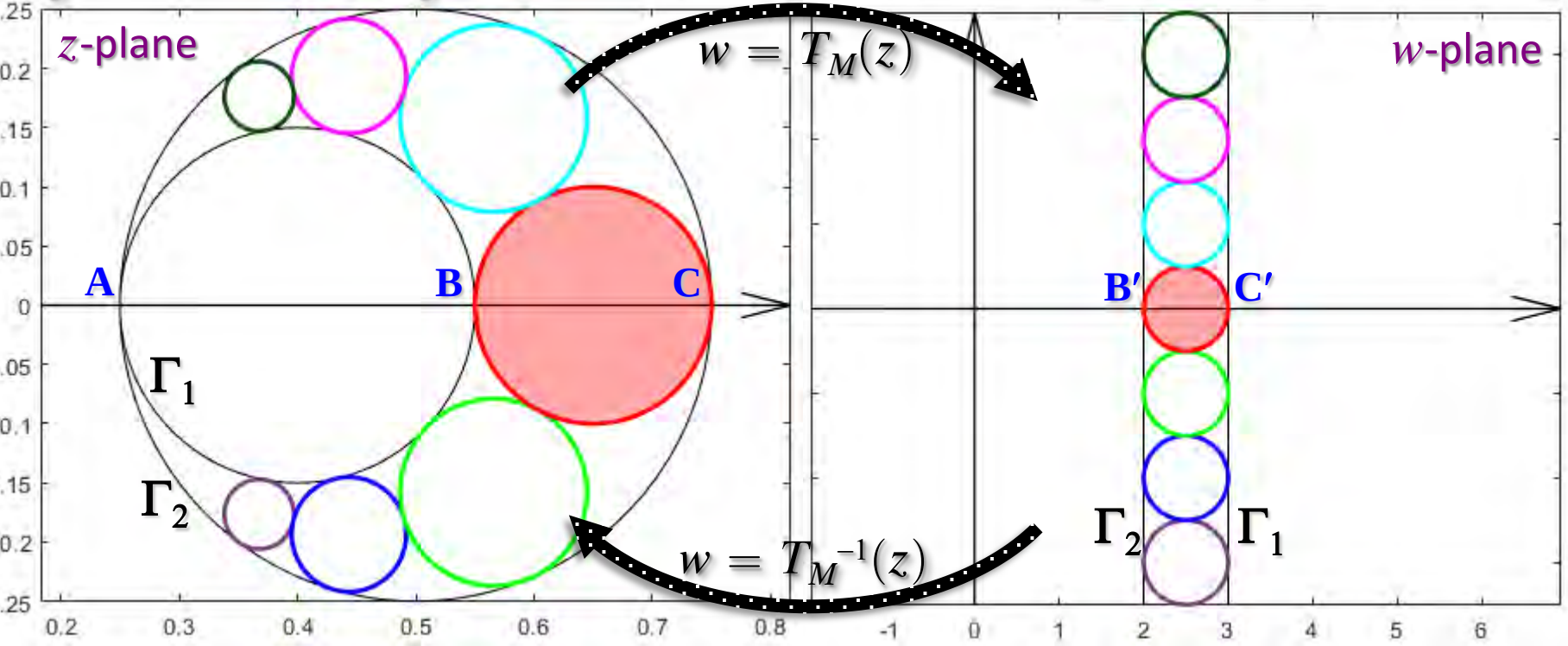
Problem: Given two circles Γ_1 and Γ_2, internally tangent at a point $A(q,0)$, find all the circles, which are externally tangent each other and also tangent to Γ_1 and Γ_2, starting from that one centered on the real axis.

We have to find a Moebius map T_M such that:

$$z_1\ \mathbf{A} \leftrightarrow \infty\ w_1$$
$$z_2\ \mathbf{B} \leftrightarrow \mathbf{B}'\ w_2$$
$$z_3\ \mathbf{C} \leftrightarrow \mathbf{C}'\ w_3$$

$$\frac{w-w_1}{w_2-w_1}\cdot\frac{w_2-w_3}{w-w_3} = \frac{z-z_1}{z-z_3}\cdot\frac{z_2-z_3}{z_2-z_1}$$

by solving w.r.t. w, we find T_M, by solving w.r.t. z, we find T_M^{-1}



Exercise

By means of **TM()** and **TM1()** functions (in exercise at pag. 24 of SC2_08e.pdf) implement the algorithm to solve the problem described in the previous example, starting from centers and radii of the circles Γ_1 and Γ_2 in the z-plane:

Γ_1: center at **c1=0.4**, radius **r1=0.15**

Γ_2: center at **c2=0.5**, radius **r2=0.25**

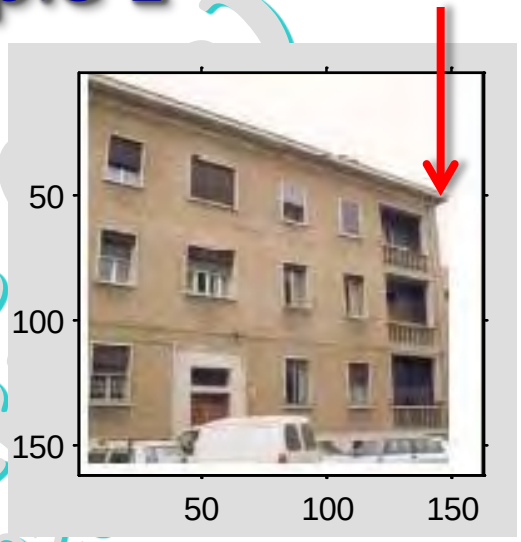
Compute also centers and radii of the output circles (in the z-plane).

Complex parametric eq. of a circle centered at $c_0 \in \mathbb{C}$ and of radius $r_0 \in \mathbb{R}^+$:

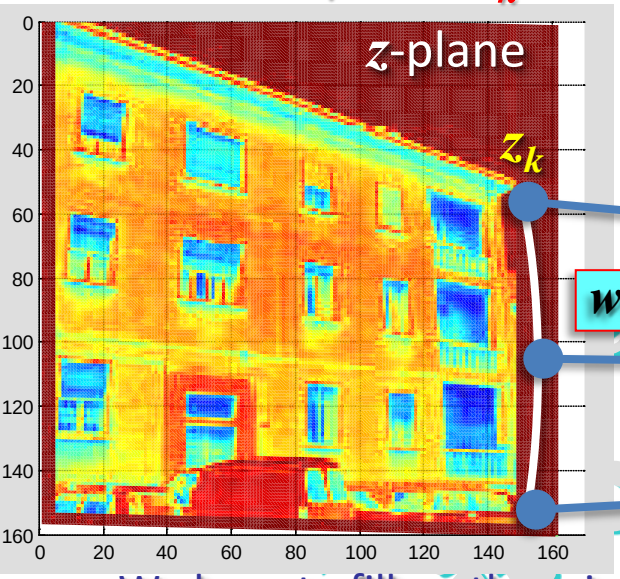
```
Gamma=@(c0,r0) c0 + r0*exp(1i*t);
```

Application of T_M: example 2

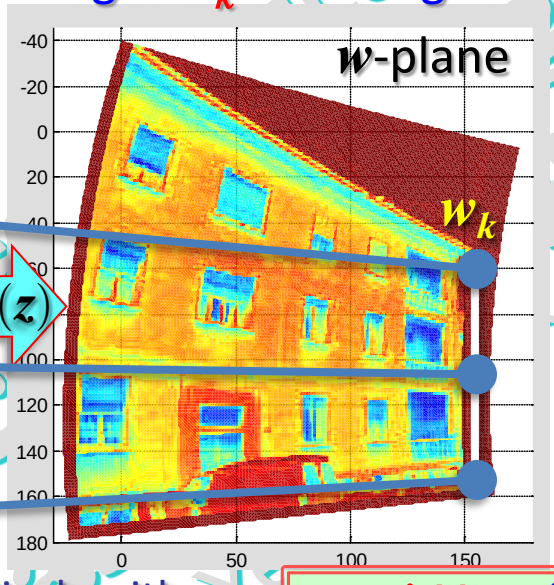
We want to correct the following image (straightening a crooked image):



We choose 3 points z_k and their images w_k to be aligned



$w = T_M(z)$

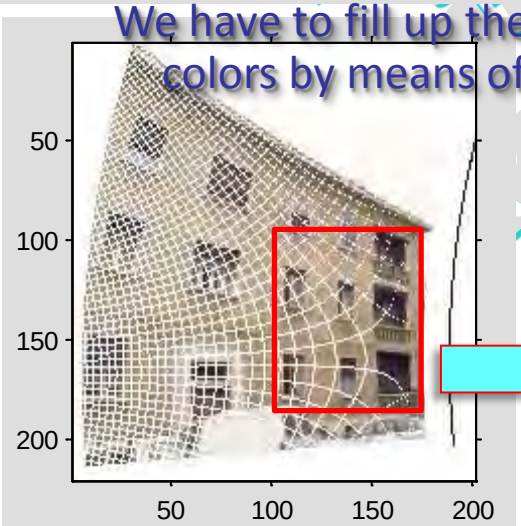


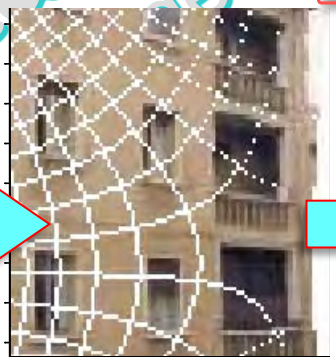
MATLAB

```
Z=griddata(xi,yi,zi,X,Y,method);
```

```
F=scatteredInterpolant(xi,yi,zi,method);
```

We have to fill up the missing pixels with colors by means of 2D interpolation

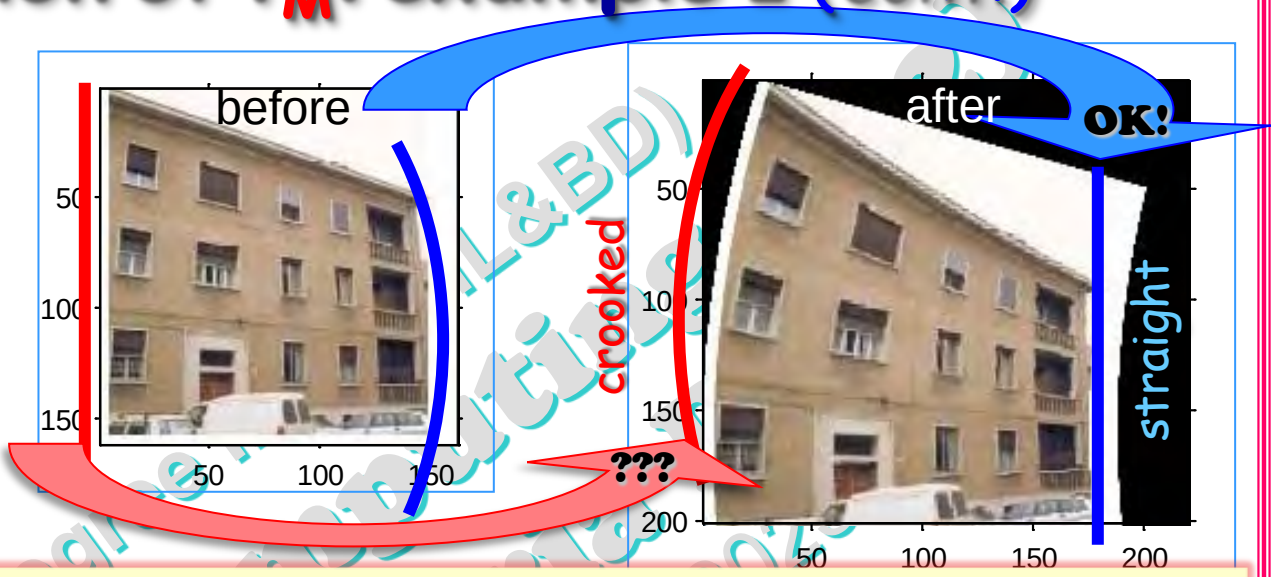




method:
'linear'
'nearest'
...

Application of T_M: example 2 (cont.)

Problem:



Solution: We may apply **two local maps**, such that we pass from one to another with continuity.

identity $w = f_1(z) = z$

Moebius $w = f_2(z) = T_M(z)$

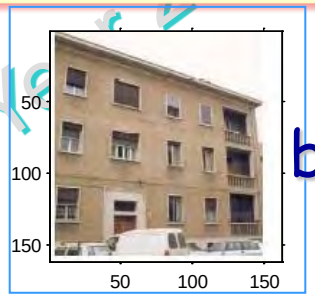
$$w = f_1(z) \longrightarrow w = f_2(z)$$

$$\tau = \frac{\text{real}(z)}{\max[\text{real}(z)]}$$

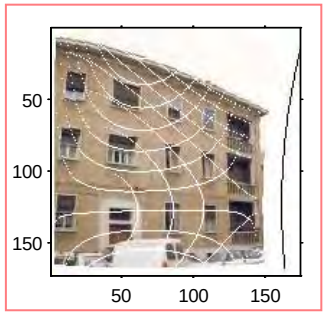
$$0 \xrightarrow{\ \tau\ } 1$$

$$w = (1-\tau) f_1(z) + \tau f_2(z), \quad \tau \in [0,1]$$

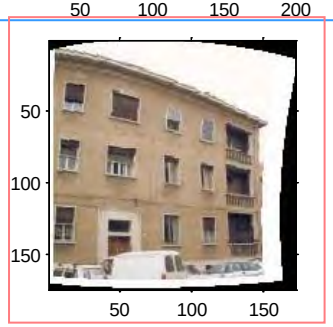
convex combination of $f_1(z)$ and $f_2(z)$



before

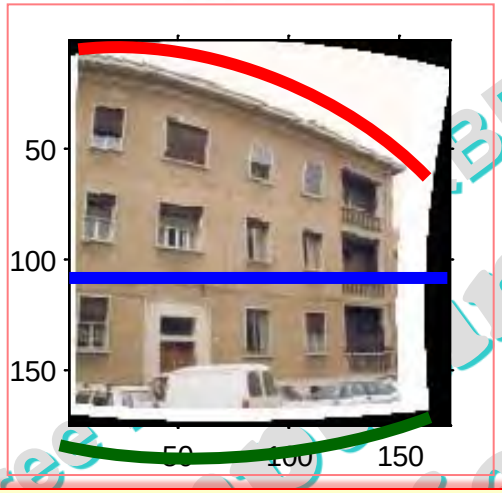


now



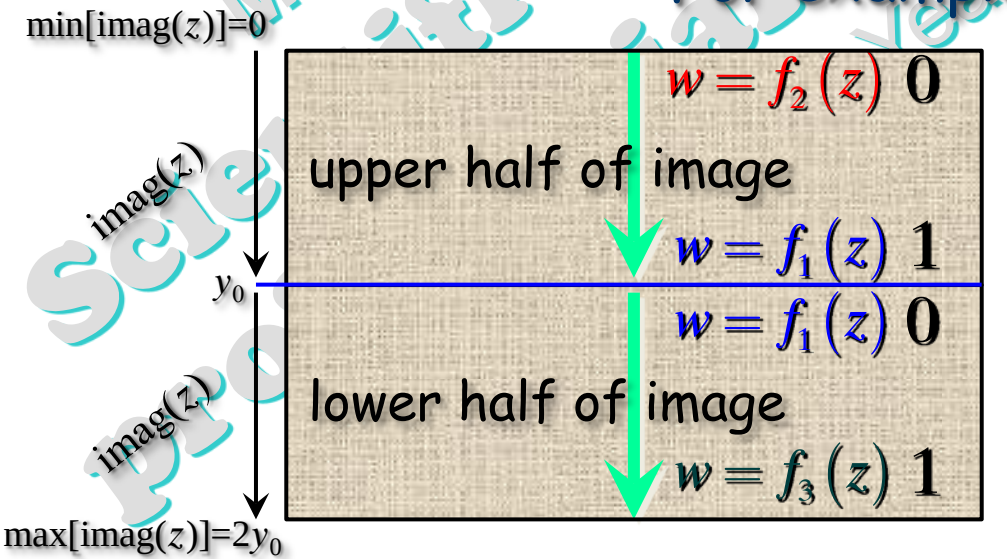
Application of T_M: example 2 (cont.)

Problem:



Solution: Divide the image vertically into two parts (or more), and apply local maps to each portion of the image.

For example:



$w = f_2(z) = T_M^{[1]}(z)$ ← Moebius

$w = (1-\tau) f_2(z) + \tau f_1(z)$

$w = f_1(z) = z$ ← identity

$w = (1-\tau) f_1(z) + \tau f_3(z)$

$w = f_3(z) = T_M^{[2]}(z)$ ← Moebius

Application of T_M: example 2 (cont.)

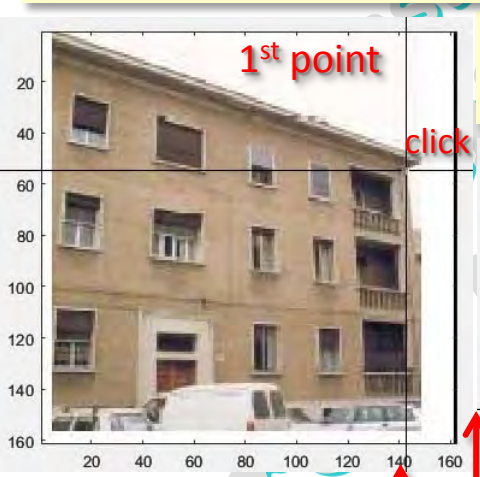
How to read an image from a graphic file and display it in MATLAB

```
f=imread('photo.png');
figure(1); imshow(f); axis on
```

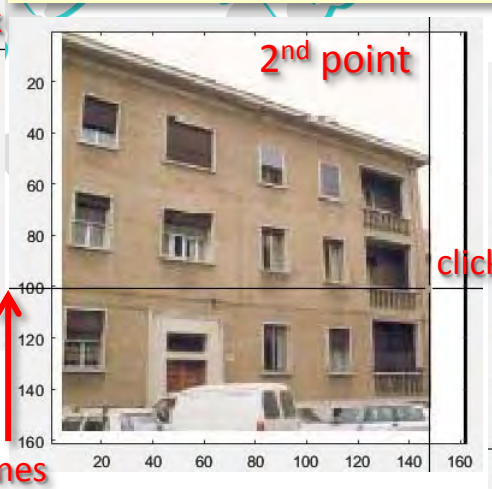
How to find, in the image, the coordinates (pixels) of 3 points to be straightened: **ginput()**

```
f=imread('photo.jpg');
figure; imagesc(f)
axis equal; axis tight
[x,y]=ginput(3);
```

[x,y]=ginput(n) allows you to identify the coordinates of **n** points within Cartesian, polar, or geographic axes. To choose a point, move your cursor to the desired location and press either a mouse button or a key on the keyboard. Press the Return key to stop before all **n** points are selected. MATLAB returns the coordinates of your selected points.



When you start the command execution, by moving the mouse over the image, two guidelines appear to facilitate the location of a point.



At end:

```
[xk,yk]=ginput(3)
xk =
         144
         148
         147
yk =
          55
         102
         144
```

coordinates (pixels)

(prof. M. Rizzardi)

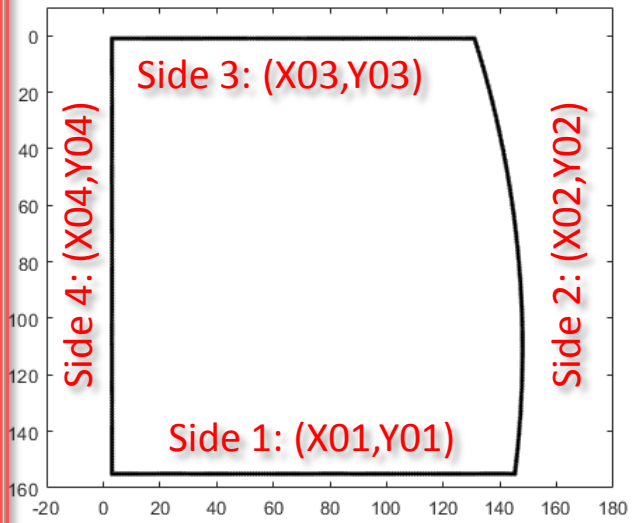
MATLAB Lab. (level 1 - easy)

How to create a rectangle with a curved side

```
%% input point zk
%[x,y]=ginput(3); % from image photo.jpg
xk=[144;148;147]; yk=[ 55;102;144]; zk=xk+1i*yk;
Y04=(1:155)'; X04=3*ones(size(Y04));
                                    Find the circle passing through (xk,yk)
%% Eq. of the circle (side 2): x^2+y^2+c(1)*x+c(2)*y+c(3) = 0
A=[xk yk ones(3,1)]; b=-(xk.^2+yk.^2); c=A\b
syms x y real; Eqn=x^2+y^2+dot(c,[x;y;1]) == 0;
sol=solve(Eqn,x,'ReturnConditions',true)
% sol.x: two solutions ==> sol.x(1) < 0,  sol.x(2) > 0
% figure; fplot(sol.x(1),[0 160]); hold on; fplot(sol.x(2),[0 160])

Y02=flipud(Y04); % side 2: curved
X02=double(subs(sol.x(2),{y},{Y02}));
X01=(3:X02(1))'; Y01=155*ones(size(X01));
X03=flipud((3:X02(end))'); Y03=1*ones(size(X03));
Z01=X01+1i*Y01; Z02=X02+1i*Y02;
Z03=X03+1i*Y03; Z04=X04+1i*Y04;

%% draw the curved rectangle counterclockwise
figure(1); clf
plot([X01;X02;X03;X04],[Y01;Y02;Y03;Y04],'.k-')
axis equal; axis ij; axis([-20 180 -10 160])
```



axis xy: for math plots
The origin is at the bottom left

axis ij: to draw matrices or images
The origin is at the top left

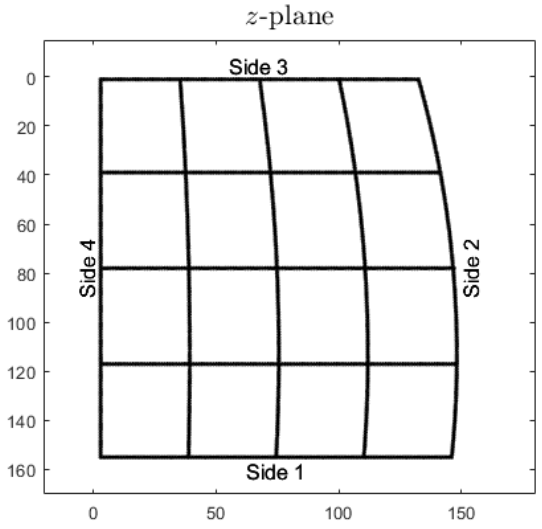
MATLAB Lab. (level 1 - easy) (cont.)

Add a grid to the curved rectangle

```
text(mean(X01([1 end])),Y01(1),'Side 1','VerticalAlignment','top', ...
                               'HorizontalAlignment','center','FontSize',12)
text(mean(X03([1 end])),Y03(1),'Side 3','VerticalAlignment','bottom', ...
                               'HorizontalAlignment','center','FontSize',12)
text(X04(1),mean(Y04([1 end])),'Side 4','VerticalAlignment','bottom', ...
                    'HorizontalAlignment','center','Rotation',90,'FontSize',12)
text(max(X02),mean(Y02([1 end])),'Side 2','VerticalAlignment','top', ...
                    'HorizontalAlignment','center','Rotation',90,'FontSize',12)

%% internal grids
X05=(flipud(X02)+X04)/2;      Y05=Y04;
X06=(X05+X04)/2;              Y06=Y04;
X07=(flipud(X02)+X05)/2;      Y07=Y04;
j=(Y04(1)+Y04(end))/2;
X08=(X04(j):round(X02(j)))'; Y08=j*ones(size(X08));
j=(Y04(1)+Y04(end))/4;
X09=(X04(j):round(X02(numel(X02)+1-j)))';
Y09=j*ones(size(X09));
j=round((Y08(1)+Y01(1))/2);
X10=(X04(j):round(X02(numel(X02)+1-j)))';
Y10=j*ones(size(X10));
Z05=X05+1i*Y05; Z06=X06+1i*Y06; Z07=X07+1i*Y07;
Z08=X08+1i*Y08; Z09=X09+1i*Y09; Z10=X10+1i*Y10;

%% draw the internal grid
figure(1); hold on
plot(X05,Y05,'.k-'); plot(X06,Y06,'.k-')
plot(X07,Y07,'.k-'); plot(X08,Y08,'.k-')
plot(X09,Y09,'.k-'); plot(X10,Y10,'.k-')
title('$z$-plane','FontSize',16,'Interpreter','LaTeX')
```



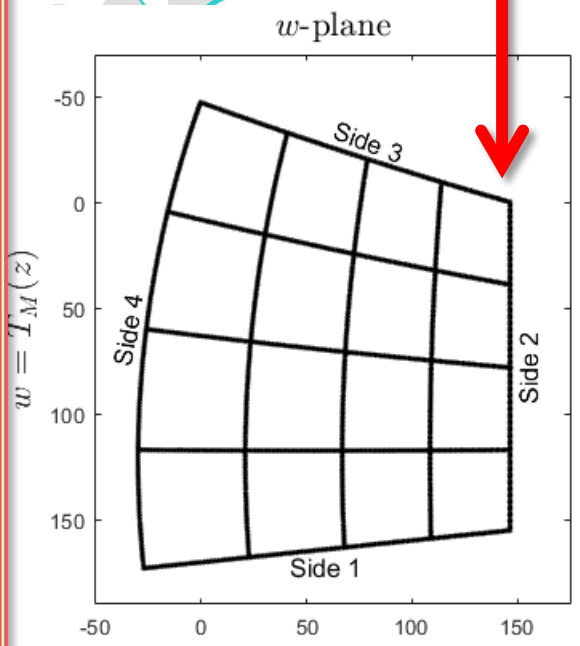
MATLAB Lab. (level 1 - easy) (cont.)

Draw the image of the curved rectangle in the w-plane

```
W01=TM(Z01); W02=TM(Z02); ... W10=TM(Z10);

figure(2); clf
plot(real([W01;W02;W03;W04]), ...
                              imag([W01;W02;W03;W04]),'.k-')
axis equal; axis ij; axis([-50 175 -70 190])
text(mean(real(W01([1 end]))),mean(imag(W01([1 end]))),'Side 1', …
   'VerticalAlignment','top','HorizontalAlignment','center','FontSize',12)
text(mean(real(W03)),mean(imag(W03)),'Side 3', …
          'VerticalAlignment','bottom','HorizontalAlignment','center', …
                                          'Rotation',-20,'FontSize',12)
text(mean(real(W04))-5,mean(imag(W04)), 'Side 4', …
          'VerticalAlignment','bottom','HorizontalAlignment','center', …
                                          'Rotation',80,'FontSize',12)
text(mean(real(W02)),mean(imag(W02)),'Side 2', …
            'VerticalAlignment','top','HorizontalAlignment','center', …
                                          'Rotation',90,'FontSize',12)
hold on
plot(real(W05),imag(W05),'.k-')
plot(real(W06),imag(W06),'.k-')
plot(real(W07),imag(W07),'.k-')
plot(real(W08),imag(W08),'.k-')
plot(real(W09),imag(W09),'.k-')
plot(real(W10),imag(W10),'.k-')
ylabel('$w=T_M(z)$','FontSize',16,'Interpreter','LaTeX')
title('$w$-plane','FontSize',16,'Interpreter','LaTeX')
```

straightened



Exercise: How to avoid distorting Side 4?

MATLAB Lab. (level 2)

Create a grid (of integer values=pixels) with a curved side

```
%% input points zk, wk
%[x,y]=ginput(3); % from image photo.jpg
xk=[144;148;147]; yk=[ 55;102;144]; zk=xk+1i*yk;
uk=mean(xk);      vk=yk;            wk=uk+1i*vk;
Y04=(1:155)'; X04=3*ones(size(Y04));

%% TM: Moebius mapping
syms W Z
Eqn=(W-wk(1))*(wk(2)-wk(3))/((W-wk(3))*(wk(2)-wk(1))) == ...
               (Z-zk(1))*(zk(2)-zk(3))/((Z-zk(3))*(zk(2)-zk(1)));
S=solve(Eqn,W,'ReturnConditions',true)
fprintf('\nTM: trasformazione di Moebius\n')
TM=matlabFunction(simplify(S.W,50))

%% Eq. of the circle (side 2) x^2+y^2+c(1)*x+c(2)*y+c(3) = 0
A=[xk yk ones(3,1)]; b=-(xk.^2+yk.^2); c=A\b;
syms x y real; Eqn=x^2+y^2+dot(c,[x;y;1]) == 0;
sol=solve(Eqn,x,'ReturnConditions',true)

Y02=flipud(Y04); % curved side 2
X02=floor(double(subs(sol.x(2),{y},{Y02})));
X01=(3:X02(1))'; Y01=155*ones(size(X01));
X03=flipud((3:X02(end))'); Y03=1*ones(size(X03));
Z01=X01+1i*Y01; Z02=X02+1i*Y02;
Z03=X03+1i*Y03; Z04=X04+1i*Y04;
```

```
%% grid
Xmin=min([X01;X02;X03;X04]);
Xmax=ceil(max([X01;X02;X03;X04]));
Ymin=min([Y01;Y02;Y03;Y04]);
Ymax=max([Y01;Y02;Y03;Y04]);
[x,y]=meshgrid(Xmin:Xmax,Ymin:Ymax);

%% NaN
for k=1:numel(Y04)
    H=find(Y02 == k);
    J=find(x(k,:) > X02(H));
    x(k,J)=NaN;
end
z=x + 1i*y;

%%
W01=TM(Z01); W01=floor(W01);
W02=TM(Z02); W02=floor(W02);
W03=TM(Z03); W03=floor(W03);
W04=TM(Z04); W04=floor(W04);
w=TM(z); w=floor(w);
```

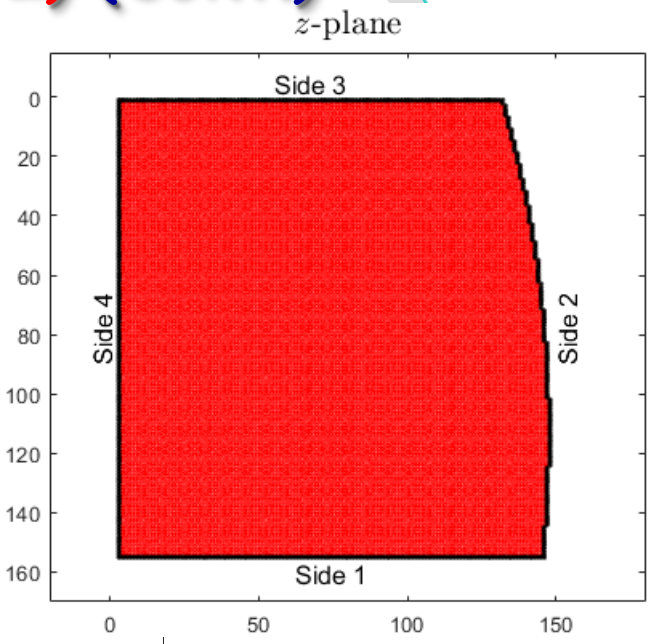
floor(x)=$\lfloor x \rfloor$ rounds each element of **x** to the nearest integer less than or equal to that element.

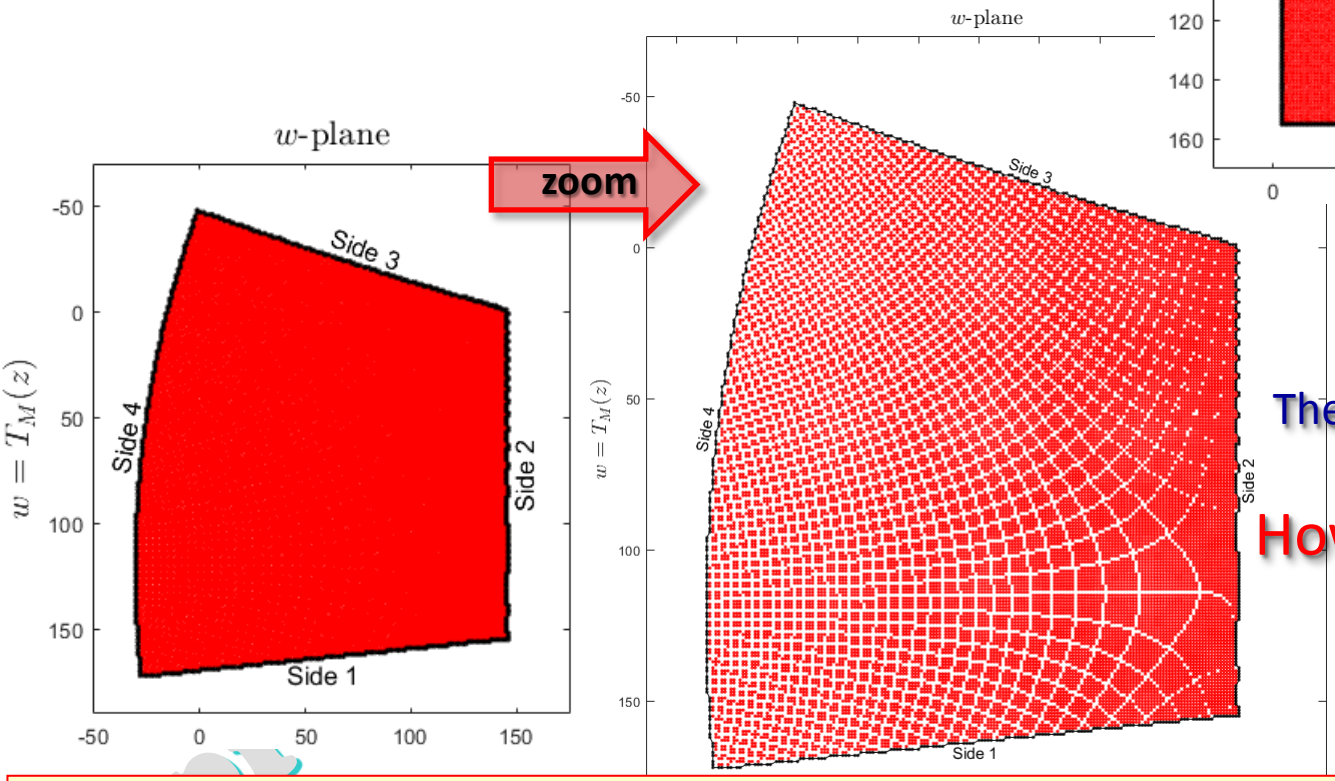
ceil(x)=$\lceil x \rceil$ rounds each element of **x** to the nearest integer greater than or equal to that element.

MATLAB Lab. (level 2) (cont.)

```
%% draw
figure(1); plot(x,y,'.r','MarkerSize',4)
axis equal; axis ij; axis([-20 180 -15 170]); hold on
plot([X01;X02;X03;X04],[Y01;Y02;Y03;Y04],'.k-')
title('$z$-plane','FontSize',16,'Interpreter','LaTeX')

figure(2); plot(real(w),imag(w),'.r'); hold on
plot(real([W01;W02;W03;W04]),imag([W01;W02;W03;W04]),'.k-')
axis equal; axis ij; axis([-50 175 -70 190])
```





There are some white pixels (undefined pixels)

How to remove them?

Exercise: How can we avoid distorting sides n. 4, 1 and 3?

Exercise (level 3)

Download photo.jpg, or photo2.png, or photo4.jpg, or choose another crooked image and …

Implement the previous algorithm to straighten a crooked image.

photo.jpg

photo2.png

photo4.jpg

Brief recap about RGB images in MATLAB

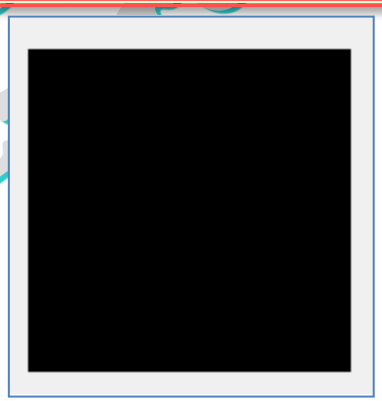
```
I=imread('photo.png');
imshow(I)
size(I)
ans =
   161   161     3
```

image dimension in pixels

Why 3?

```
[m,n,p]=size(I);
B=ones(m,n);
W=cat(3,B,B,B);
figure; imshow(W)
```

```
N=zeros(m,n);
W=cat(3,N,N,N);
figure; imshow(W)
```



Pixels with value 1 give white color;
pixels with value 0 give black color

3D image matrix:
each face contains red, green and blue values for each pixel

RGB triplet:
Red **G**reen **B**lue

```
Ired=cat(3,I(:,:,1),B,B);
figure; imshow(Ired)
```

```
Igreen=cat(3,B,I(:,:,2),B);
figure; imshow(Igreen)
```

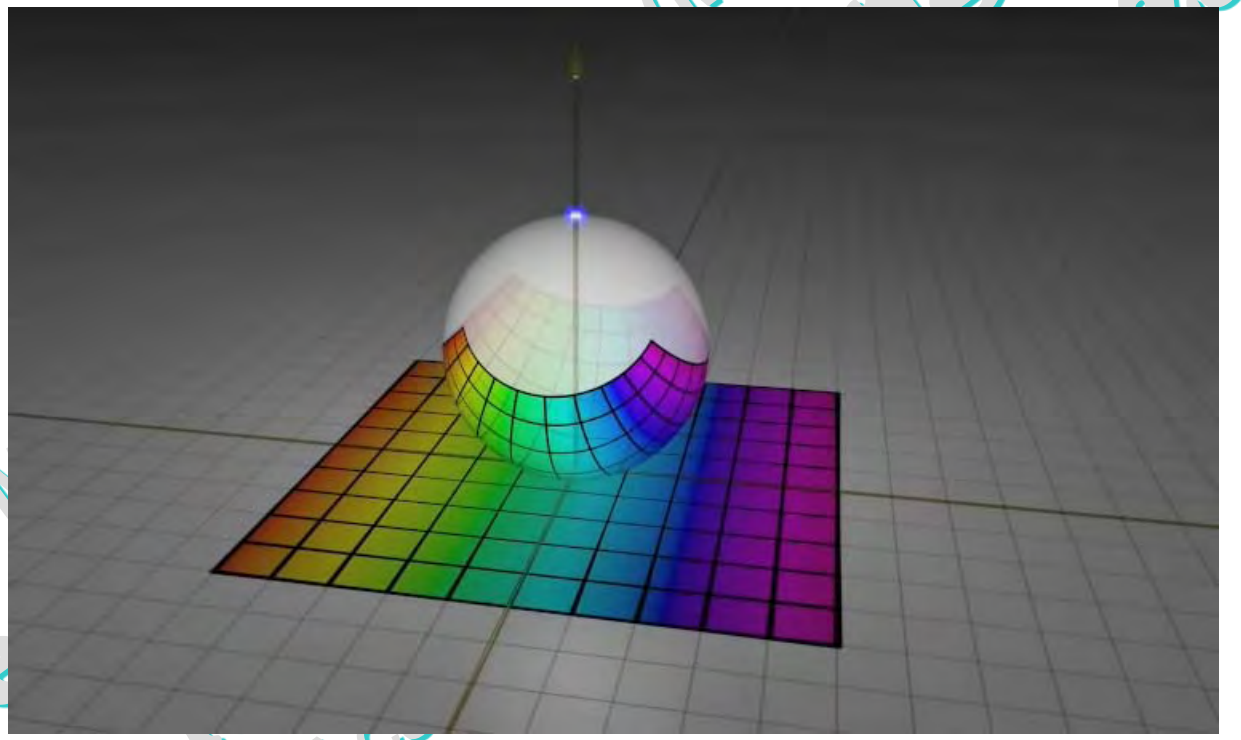
```
Iblue=cat(3,B,B,I(:,:,3));
figure; imshow(Iblue)
```

```
[max(max(abs(I(:,:,1)-I(:,:,2)))) max(max(abs(I(:,:,1)-I(:,:,3)))) max(max(abs(I(:,:,2)-I(:,:,3))))]
ans =
  1×3 uint8 row vector
   46   74   44
```

The 3 matrices, with **RGB** colors, differ

Moebius Transformations Revealed

A wonderful video, which illustrates the behavior of a Moebius map, and explains its connection with the Stereographic Projection.



The video received an Honorable Mention in the "2007 Science and Engineering Visualization Challenge", cosponsored by the National Science Foundation and *Science* Magazine.

Joukowski mappings

$$T_J : z \in \mathbb{C}^* \longrightarrow w \in \mathbb{C}^*$$

symmetric $$w = T_J(z) = \frac{a}{2}\left(z + \frac{1}{z}\right)$$

non-symmetric $$w = T_J(z) = z + \frac{L^2}{z}$$

Symmetric Joukowski mapping

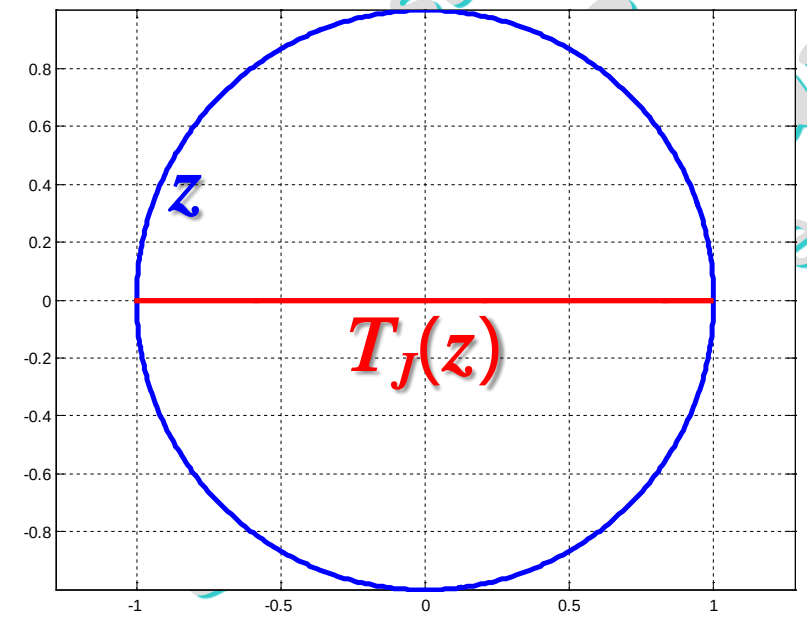
$$w = f(z) = \frac{1}{2}\left(z + \frac{1}{z}\right)$$

$$f(z) = f\left(\frac{1}{z}\right)$$ not one-to-one

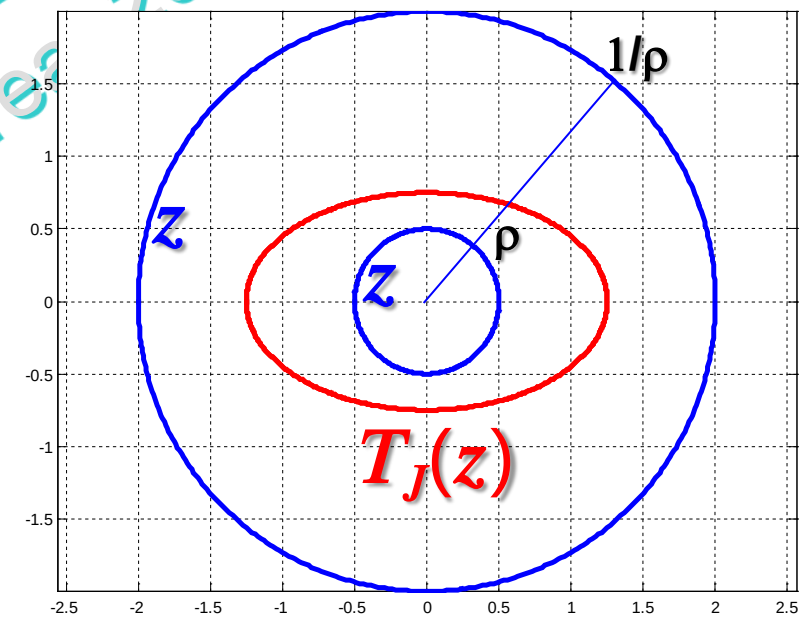
$$f'(z) = \frac{1}{2}\left(\frac{z^2 - 1}{z^2}\right)$$

critical points

conformal for $z \neq \pm 1$



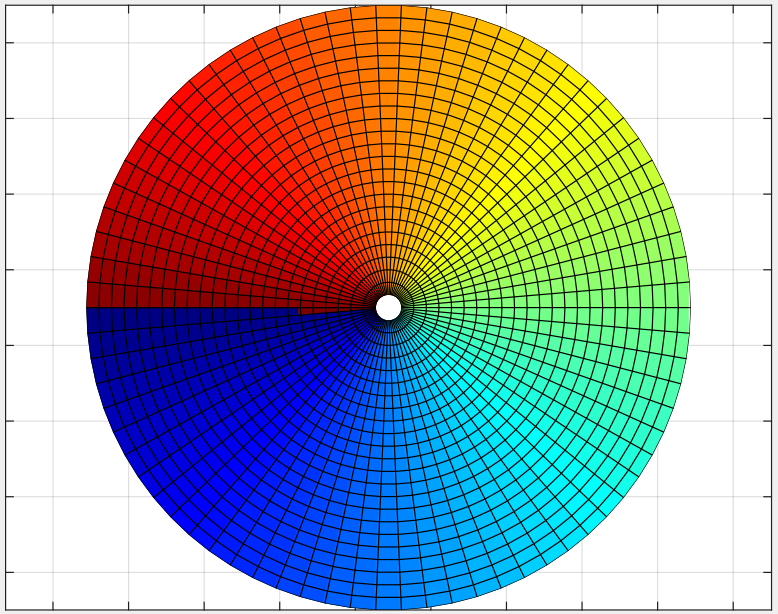
origin curve
image curve



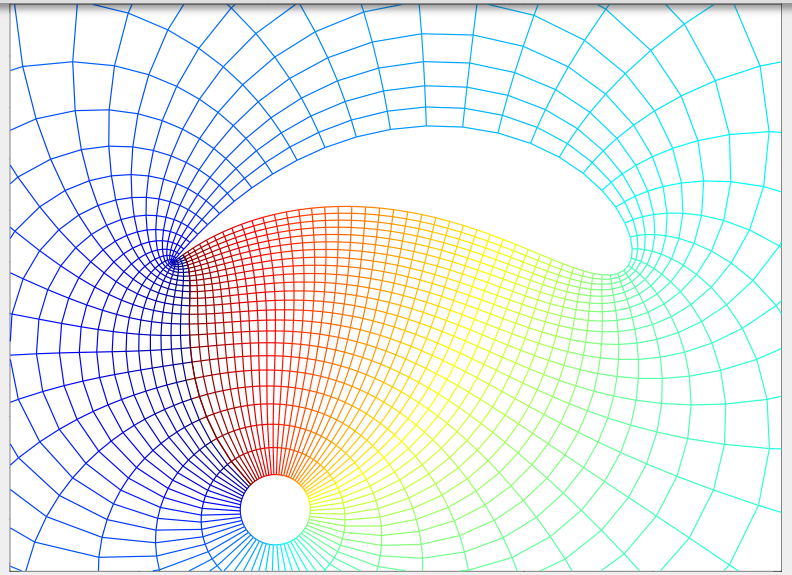
Non-symmetric Joukowski map

$$w = T_J(z) = z + \frac{0.188}{z}$$

$\Gamma(z_0, \rho) : z_0 = -0.056 + 0.15i, \quad \rho = 0.4$

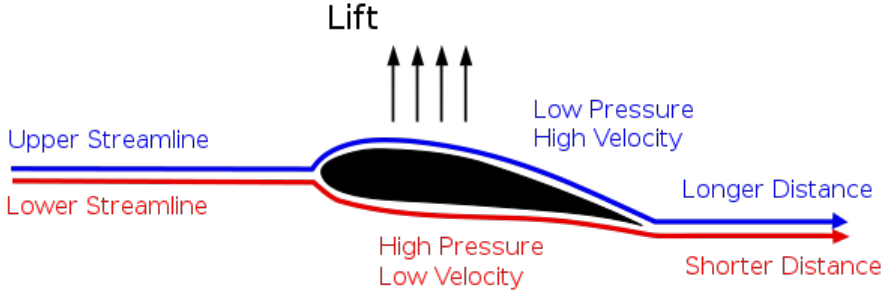


It is able to transform a circle into an airfoil



```
TJ=@(Z) Z+0.188./Z; z0=-0.056+0.15i; rhoMAX=0.4;
rho=linspace(0,rhoMAX,25)';
th=linspace(-pi,pi,75);
x=rho*cos(th); y=rho*sin(th); z=z0+x+1i*y;
[I,J]=find(abs(z-z0)<0.01); z(I,J)=nan; w=TJ(z);
figure(1); clf
surf(real(z),imag(z),zeros(size(z)),angle(z-z0))
view(2); colormap('jet'); axis equal; box on
figure(2); clf
mesh(real(w),imag(w),zeros(size(w)),angle(z-z0))
view(2); colormap('jet'); axis equal; box on
AX=[-1.5 1.5 -1.2 1]; axis(AX)
```

The Kutta-Joukowski theorem is a fundamental theorem in aerodynamics used for the calculation of lift of an airfoil

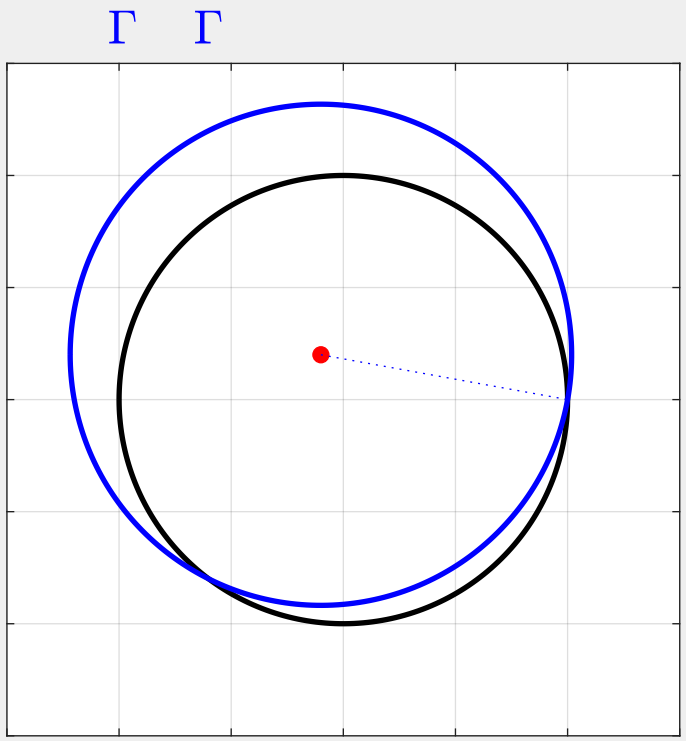


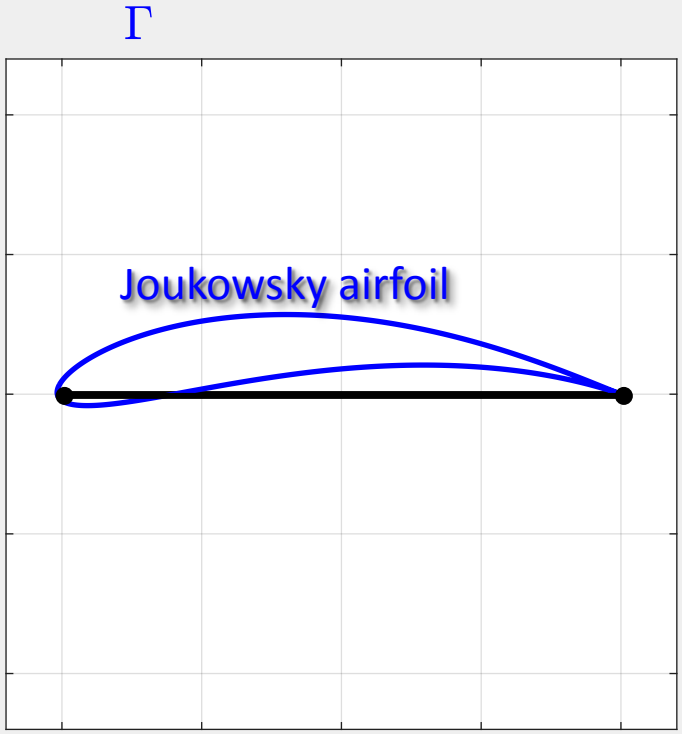
symmetric Joukowski mapping

$$z = \frac{1}{2}\left(z + \frac{1}{z}\right)$$

Nikolai Yegorovich Zhukovsky, 1910

Download: **airfoil.m**





The blue circle has been mapped into the airfoil.

The black unit circle has been mapped to the real segment [-1,+1], traversed twice

Joukowski mapping

MathWorks® Prodotti Soluzioni Università Assistenza Community Eventi

File Exchange

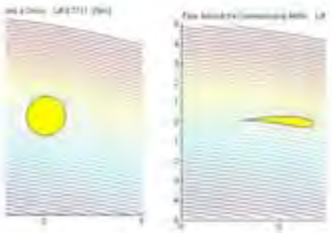
Search File Exchange

MATLAB Central | Files | Authors | My File Exchange | Publish | About

ANNOUNCEMENT

MATLAB EXPO 2023: Sign up now for free

MATLAB EXPO is coming up soon and it is time to register....

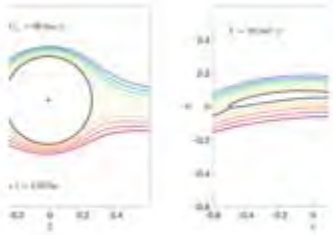


Joukowski Airfoil Transformation

Version 1.0.0.0 (1.96 KB) by Dario Isola

Script that plots streamlines around a circle and around the correspondig Joukowski airfoil.

https://it.mathworks.com/matlabcentral/fileexchange/8870-joukowski-airfoil-transformation



Joukowski Airfoil Demonstration

Version 1.0.0 (2.62 KB) by Wen Wu

Perform the Joukowski transformation to an off-center cylinder.

https://it.mathworks.com/matlabcentral/fileexchange/86902-joukowski-airfoil-demonstration