
MUSKHELISHVILI'S METHOD APPLIED TO ELLIPTICAL AND LUBRICATED CIRCULAR INCLUSIONS IN GENERAL SHEAR: SOLUTION IMPLEMENTATION IN MATLAB

Dani Schmid & Yuri Podladchikov

Geologisches Institut, ETH Zentrum, 8092 Zürich, Switzerland

Keywords: Muskhelishvili Method, Circular and Elliptical Inclusions, Lubricated Inclusions, Mantled Inclusions, Pure and Simple Shear, Viscous Rheology, Elastic Rheology, Incompressibility

INTRODUCTION

Two-dimensional problems in elasticity can be solved analytically by means of Muskhelishvili's method¹. This method is based on the fact that the result of the bi-harmonic equation can be represented by two complex potentials, called $\phi(z)$ and $\psi(z)$. The classical application of the method are circular and elliptical holes. At the beginning of the 20th century engineers had to face the problem that openings for the cannons on battle ships or for windows in airplanes could weaken the bulk structure so much that small forces/collisions could cause complete failure. This prompted the school of Kolosov and later Muskhelishvili to develop a method capable of solving such problems analytically.

In elastic geo-engineering problems such as tunnel building, Muskhelishvili's method is frequently applied, but rarely in the related field of slow viscous flow. Yet, the two problems are mathematically identical and consequently Muskhelishvili's method

¹ Muskhelishvili, N.I., 1953. Some basic problems of the mathematical theory of elasticity. Noordhoff, Groningen, 704S.

Through Kluwer's PoD program the book is available again and can be printed on demand, www.wkap.nl.

equally well applicable. We have recently shown ² the applicability of the method to slow viscous flow and we have derived all possible closed form solutions for mantled inclusions. “Mantled inclusions” means that we are dealing with a three phase system (inclusion, mantle and matrix), of incompressible materials with three, possibly different, viscosities. The inclusion is elliptical, including all shapes from infinite aspect ratio to circular inclusion. The inclusion is subjected to far-field general shear flows, i.e., arbitrary combinations of inclined pure and simple shear. Since, the solution for the mantled elliptical inclusion is an infinite series, we only present here the closed form solutions, which are: circular inclusion perfectly bonded to the matrix, circular inclusion perfectly bonded to an intermediate layer (mantle) which is perfectly bonded to the matrix, and the elliptical inclusion perfectly bonded to the matrix.

In order to stimulate a more widespread use of Muskhelishvili’s method which may be hampered by the fact that it makes use of “obscure” complex potentials we give here the complete set of MATLAB^{®3} scripts which we used to implement the solutions and visualize the results. The eight scripts provided have the following tasks:

CYL_P_INTERF.M

Pressure in the matrix at the circular clast-matrix interface.

CYL_P_INTERF_MAX.M

Maximum pressure in the matrix at the circular clast-matrix interface as a function of the viscosity contrast between clast and matrix.

CYL_P_MATRIX.M

Pressure around circular inclusion.

CYL_W_RIM

Complete two-dimensional field of pressure, maximum shear stress and stream function, when there is a third material between cylindrical inclusion and matrix.

ELL_DYNAMIX.M

2D pressure, stress and maximum shear stress caused by an elliptical inclusion

² ETH PhD Thesis by Dani Schmid, 2002.

³ MATLAB is a registered trademark of The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, USA, info@mathworks.com, <http://www.mathworks.com>

ELL_P_INTERF.M

Pressure in the matrix at the elliptical elast-matrix interface.

ELL_ROT_RATE.M

Analytical formula for the rotation rate of an ellipse in combined, inclined simple and pure shear.

ZHOUK_DEMO.M

Demonstration of the Joukowski transform.

These scripts also allow extraction of numerous other parameters with the help of the three basic formulae of Muskhelishvili's method:

$$p = -2\Re(\phi'(z)) \quad (1)$$

$$\frac{\sigma_{yy} - \sigma_{xx}}{2} + i\sigma_{xy} = \bar{z}\phi''(z) + \psi'(z) \quad (2)$$

$$v_x + iv_y = \frac{\phi(z) - z\overline{\phi'(z)} - \overline{\psi(z)}}{2\mu} \quad (3)$$

Where p is the pressure perturbation, σ is the total stress tensor, μ the viscosity, z the complex coordinate, v the velocity vector, x and y the usual Cartesian coordinates, ' means the first derivative versus z and '' the second, i is equal $\sqrt{-1}$, the over bar denotes conjugation, and \Re means the real part.

These expressions are only valid if no conformal mapping is applied. For cases where the Joukowski transformation is used, i.e., elliptical inclusions, care must be taken concerning the spatial derivatives. Since the problem is solved in a geometrically simpler image plane ζ , but the spatial derivatives must still be taken versus z , i.e., the complex coordinate of the physical plane, the relevant expressions are more complicated and the following set of equations must be employed:

$$p = -2\Re\left(\frac{1}{1-\zeta^{-2}} \frac{\partial\phi}{\partial\zeta}\right) \quad (4)$$

$$\frac{\sigma_{yy} - \sigma_{xx}}{2} + i\sigma_{xy} = \overline{\left(\zeta + \frac{1}{\zeta}\right)} \left(\frac{1}{(1-\zeta^{-2})^2} \frac{\partial^2\phi}{\partial\zeta^2} + \frac{-2}{(1-\zeta^{-2})^3 \zeta^3} \frac{\partial\phi}{\partial\zeta} \right) + \frac{\partial\psi}{\partial\zeta} \frac{1}{1-\zeta^{-2}} \quad (5)$$

$$v_x + iv_y = \frac{\phi - \left(\zeta + \frac{1}{\zeta}\right) \overline{\left(\frac{1}{1-\zeta^{-2}} \frac{\partial\phi}{\partial\zeta}\right)} - \bar{\psi}}{2\mu} \quad (6)$$

MATLAB SCRIPTS

Please note, that since these scripts are provided free of cost we put a big disclaimer on all of them and do not take any liability for correctness, errors, loss or whatever may happen; you use it at your own risk. Therefore the following applies to all of them:

%=====

DISCLAIMER OF WARRANTY:

Since the Software is provided free of charge, it is provided on an as is basis, without warranty of any kind, including without limitation the warranties of merchantability, fitness for a particular purpose and non-infringement. The entire risk as to the quality and performance of the Software is borne by the user. Should the Software prove defective, the user will assume the entire cost of any service and repair.

LIMITATION OF LIABILITY:

Under no circumstances and under no legal theory, tort, contract, or otherwise, shall the authors be liable to the user or any other person for any indirect, special, incidental, or consequential damages of any character including, without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses.

%=====

CYL_P_INTERF.M

```
=====
CYL_P_INTERF.M
%
% Pressure in the matrix at the circular clast-matrix interface.
%
% 2002, Dani Schmid
%
=====
%SETUP ANGLES
theta    = 0:2*pi/359:2*pi;

%VISCOSITIES
mm       = 1;
mcs      = [1,2,10,1e6];

%FAR FIELD FLOW
er       = 0;
gr       = 1;

%CALCULATE AND PLOT
Styles   = {':k', '-.k', '--k', '-k'};
figure(1);
clf
counter=1;
for mc=mcs;
    Pressure    = 2.*mm.*(-2.*mc.*cos(2.*theta).*er- ...
                  mc.*gr.*sin(2.*theta)+2.*mm.*cos(2.*theta).* ...
                  er+mm.*gr.*sin(2.*theta))./(mc+mm);
    plot(theta/pi*180, Pressure, Styles{counter});
    hold on;
    counter     = counter+1;
end

axis([0 360 -2 2]);
set(gca, 'XTick', [0:45:360]);
xlabel('\theta');
ylabel('p/(\mu_m\gamma)', 'Rotation', 0);
title('Pressure Around Cylindrical Inclusion')
legend('\mu_c/\mu_m=1', '\mu_c/\mu_m=2', '\mu_c/\mu_m=10', ...
       '\mu_c/\mu_m=\infty', -1);
```

CYL_P_INTERF_MAX.M

```
=====
CYL_P_INTERF_MAX.M
%
% Maximum pressure in the matrix at the circular clast-matrix
% interface as a function of the viscosity contrast between clast
% and matrix.
%
% 2002, Dani Schmid
%
=====

%LOGARITHMIC RANGE OF VISCOSITIES
mm      = 1;
mc      = logspace(-3,3);

%FAR FIELD FLOW
gr      = 1;

%MAXIMUM PRESSURE EXPRESSION
Pressure = 2.*mm.*(mc-mm)./(mc+mm).*gr;

%PLOT PRESSURE vs. THETA
figure(1);
clf
plot(mc, Pressure, '-k');
set(gca, 'XScale', 'log');
grid on;
set(gca, 'XMinorgrid', 'off');

xlabel('\mu_c/\mu_m');
ylabel('p/(\mu_m\gamma)', 'Rotation', 0);
title('Max. Pressure as f(\mu_c/\mu_m)')
```

CYL_P_MATRIX.M

```
%=====
CYL_P_MATRIX.M
%
% Pressure around circular inclusion
%
% 2002, Dani Schmid
%
%=====
%FAR FIELD FLOW - VISCOSITIES - GEOMETRY
gr    = 1;
er    = 0;
mm    = 1;
mc    = 1e6;
rc    = 1;

%PRESSURE CALCULATION IN THE Z-PLANE
[X,Y] = meshgrid(-2:.01:2);
Z      = X+i*Y;
P      = -2.*mm.*(mc-mm)./(mc+mm).*real(rc^2./Z.^2.*(i*gr+2*er));

%PRESSURE IS ONLY FOR THE OUTSIDE OF THE CLAST
P(abs(Z)<rc) = NaN;

%PLOTTING
pcolor(X,Y,P)
axis image;
shading interp;
hold on;
contour(X,Y,P, [-1.5,-1,-.5,0,.5,1,1.5], 'k');
colorbar('horiz');
```

CYL_W_RIM

```

=====
% CYL_W_RIM
%
% Complete two-dimensional field of pressure, maximum shear stress
% and stream function
%
% 2002, Dani Schmid
%
=====
%DEFINE i
i   = sqrt(-1);
I   = sqrt(-1);

%INPUT PARAMETERS
er  = -0;    %Negative values indicate horizontal compression
gr  = 1;     %Positive value indicate top to the left shear
rl  = 1.2;
ml  = 1e+3;
mc  = 1e+3;

%DESIRED BOX SIZE
bs  = 3; %Times rl

%K's
K1  = ml*mc;
K2  = (ml-mc)*(ml-1);
K3  = (mc+ml)*(ml+1);
K4  = ml/(ml-1);
K5  = ml/(ml+1);
K6  = (ml-mc)*(ml+1);
K7  = (mc+ml)*(ml-1);
K8  = (ml-mc)*(mc+ml)*(1-ml+ml^2);
K9  = (ml-mc)*(mc+ml)*(3-8*ml+3*ml^2);

%Q's
Q0  = K2^2+K2*K3*(-4*rl^2+6*rl^4-4*rl^6)+K3^2*rl^8;
Q1  = 4*K1*K2*(rl^2-rl^4)/Q0;
Q2  = (-16*K1*K2*rl^2+12*K1*K2*rl^4+4*K1*K3*rl^8)/Q0;
Q3  = K2*K4*(-2*K2*rl^2+2*K3*rl^8)/Q0;
Q4  = K2*(ml^2+K1)*(2*rl^2-2*rl^4)/Q0;
Q5  = K2*K4*(-2*K2*rl^4+2*K3*rl^8)/Q0;
Q6  = K3*(K2*K5*(-8*rl^2+6*rl^4)+2*(ml^2+K1)*rl^8)/Q0;
Q7  = (-K2*K6*rl^2+4*K2*K7*rl^4-6*K2*K7*rl^6+4*K8*rl^8 ...
      -K3*K7*rl^10)/Q0;
Q8  = (-K2*K6*rl^4+4*K2*K7*rl^6-2*K9*rl^8+4*K2*K7*rl^10 ...
      -K3*K7*rl^12)/Q0;

%RESOLUTION
nr   = 100;
nt   = 200;
Theta = 0:2*pi/nt:2*pi;

```

```

%CLAST
[R, THETA] = meshgrid(0:1/nr:1, Theta);
z          = R.*exp(i*THETA);
x          = real(z);
y          = imag(z);
Z_CLAST   = z; %Save z's for later contour plots

%Pressure
figure(1);
clf
PRES_CLAST = -6*Q1*real(z.^2*(-2*er+i*gr));
pcolor(real(z), imag(z), PRES_CLAST);
hold on;
%Streamfun
figure(2);
clf
STREAM_FUN_CLAST = er.*(-2./mc.*Q1.*y.^3.*x-1./mc.*Q2.*y.*x ...
    -2./mc.*Q1.*x.^3.*y)+(-1./4.*y.^2-1./4.*1./mc.*Q2.*y.^2 ...
    -1./2.*1./mc.*y.^4.*Q1).*gr;
pcolor(real(z), imag(z), STREAM_FUN_CLAST);
hold on;
%TAU
figure(3);
clf
TAU_CLAST = sqrt((-2.*Q2.*er+6.*Q1.*real(conj(z).*z.*(I.*gr ...
    -2.*er))).^2+(Q2.*gr+6.*Q1.*imag(conj(z).*z.*(I.*gr-2.*er))).^2);
pcolor(real(z), imag(z), TAU_CLAST);
hold on;

%LUBR
[R, THETA] = meshgrid(1:(rl-1)/nr:rl, Theta);
z          = R.*exp(i*THETA);
x          = real(z);
y          = imag(z);
Z_LUBR    = z;
%Pressure
figure(1);
PRES_LUBR = 2*real((i*gr+2*er)*Q3./z.^2-3*(-2*er+i*gr)*Q4*z.^2);
pcolor(real(z), imag(z), PRES_LUBR);
%Streamfun
figure(2);
STREAM_FUN_LUBR = ...
    -1./2./ml.*er.*(4.*x.*y.^3.*Q4+2.*y.*Q3./(x.^2+y.^2).*x ...
    -2.*Q5.*y.*x./(x.^2+y.^2).^2-x.*(-2.*Q3./(x.^2+y.^2).*y ...
    -6.*Q4.*(x.^2.*y-1./3.*y.^3))-2.*Q4.*(x.^3.*y ...
    -y.^3.*x)+2.*Q6.*y.*x)-1./2.*1./ml.*gr.*(1./2.*y.^2.*ml ...
    -x.*(Q3./(x.^2+y.^2).*x-3.*Q4.*x.*y.^2) ...
    -3./2.*Q4.*x.^2.*y.^2+3./4.*Q4.*y.^4-x.^2.*Q3./(x.^2+y.^2) ...
    -Q4.*(3./2.*x.^2.*y.^2-1./4.*y.^4)-Q5.*( ...
    -x.^2./(x.^2+y.^2).^2+1./2./(x.^2+y.^2))+1./2.*Q6.*y.^2);

pcolor(real(z), imag(z), STREAM_FUN_LUBR);
%TAU
figure(3);
TAU_LUBR = sqrt(real( ...
    -conj(z).* (2.*(I.*gr+2.*er).*Q3./z.^3+6.*(I.*gr ...
    -2.*er).*Q4.*z)+3.*(I.*gr+2.*er).*Q5./z.^4-(I.*gr ...
    -2.*er).*Q6).^2+imag(-conj(z).* (2.*(I.*gr+2.*er).*Q3./z.^3 ...

```

```

+6.*(I.*gr-2.*er).*Q4.*z)+3.*(I.*gr+2.*er).*Q5./z.^4-(I.*gr ...
-2.*er).*Q6).^2);
pcolor(real(z), imag(z), TAU_LUBR);

%MAT
[R, THETA] = ...
meshgrid(rl:(sqrt(2*bs^2*rl^2)-rl)/nr:sqrt(2*bs^2*rl^2), Theta);
z          = R.*exp(i*THETA);
x          = real(z);
y          = imag(z);
Z_MAT     = z;
%Pressure
figure(1)
PRES_MAT  = 2*real((i*gr+2*er)*Q7./z.^2);
pcolor(real(z), imag(z), PRES_MAT);
%Streamfun
figure(2)
STREAM_FUN_MAT = ...
er.*(-2.*Q7.*y.*x./(x.^2+y.^2)+Q8.*y.*x./ ...
(x.^2+y.^2).^2-x.*y)+(-1./2.*y.^2+Q7.*x.^2./ ...
(x.^2+y.^2)+1./2.*Q8.*(-x.^2./(x.^2+y.^2).^2+1./2./ ...
(x.^2+y.^2))).*gr;
pcolor(real(z), imag(z), STREAM_FUN_MAT);
%TAU
figure(3);
TAU_MAT   = ...
sqrt((-2.*er+real(2.*conj(z).* (I.*gr+2.*er).*Q7./z.^3 ...
-3.*(I.*gr+2.*er).*Q8./z.^4)).^2+(gr+imag(2.*conj(z).* ...
(I.*gr+2.*er).*Q7./z.^3-3.*(I.*gr+2.*er).*Q8./z.^4)).^2);
pcolor(real(z), imag(z), TAU_MAT);

%FINALIZE PLOTS=====
%Complex coordinates of clast and layer
Clast  = exp(i*Theta);
Layer  = rl*exp(i*Theta);

figure(1)
axis off
axis equal
axis([-bs*rl bs*rl -bs*rl bs*rl]);
shading interp;
plot(real(Clast), imag(Clast), '--k');
plot(real(Layer), imag(Layer), '--k');
title(['P \epsilon:', num2str(er), ' \gamma:', num2str(gr), ...
' \mu_c: ', num2str(mc), ' \mu_l: ', num2str(ml), ...
' r_l: ', num2str(rl)]);
cb_h    = colorbar('horiz. ');
pos_cb  = get(cb_h, 'pos');
set(cb_h, 'pos', [.26666, pos_cb(2), .504, pos_cb(4)])
set(get(cb_h, 'Title'), 'String', 'P');

figure(2)
axis equal
axis([-bs*rl bs*rl -bs*rl bs*rl]);
shading interp;
plot(real(Clast), imag(Clast), '--k');
plot(real(Layer), imag(Layer), '--k');
colorbar('horiz. ');

```

```

Z          = [ Z_MAT          ];
STREAM_FUN = [ STREAM_FUN_MAT ];
contour(real(Z), imag(Z), STREAM_FUN, 10, 'k')
axis off;
axis equal
axis([-bs*rl bs*rl -bs*rl bs*rl]);
shading interp;
plot(real(Clast), imag(Clast), '--k');
plot(real(Layer), imag(Layer), '--k');
title(['\Theta \epsilon:', num2str(er), ' \gamma:', num2str(gr), ...
' \mu_c: ', num2str(mc), ' \mu_l: ', num2str(ml), ' r_l: ', ...
num2str(rl)]);
cb_h      = colorbar('horiz. ');
pos_cb    = get(cb_h, 'pos');
set(cb_h, 'pos', [.26666, pos_cb(2), .504, pos_cb(4)])
set(get(cb_h, 'Title'), 'String', '\Theta');

figure(3)
axis off
axis equal
axis([-bs*rl bs*rl -bs*rl bs*rl]);
shading interp;
plot(real(Clast), imag(Clast), '--k');
plot(real(Layer), imag(Layer), '--k');
title(['\tau \epsilon:', num2str(er), ' \gamma:', num2str(gr), ...
' \mu_c: ', num2str(mc), ' \mu_l: ', num2str(ml), ' r_l: ', ...
num2str(rl)]);
cb_h      = colorbar('horiz. ');
pos_cb    = get(cb_h, 'pos');
set(cb_h, 'pos', [.26666, pos_cb(2), .504, pos_cb(4)])
set(get(cb_h, 'Title'), 'String', '\tau');

```

ELL_DYNAMIX.M

```
%=====
% ELL_DYNAMIX.M
%
% 2D pressure, stress and maximum shear stress caused by an
% elliptical inclusion
%
% 2002, Dani Schmid
%=====

%COMPLEX NUMBER DEFINITION
I      = sqrt(-1);
i      = sqrt(-1);

%VISCOSITY CONTRAST BETWEEN CLAST AND MATRIX
mc     = 1000;

%FAR FIELD FLOW
er     = -.5;
gr     = 0;
alpha  = -30/180*pi;

%ASPECT RATIO, t CANNOT BE 1 or SMALLER, USE t=1.001 FOR CIRCULAR
INCLUSION APPROXIMATION
t      = 4;
rc     = sqrt((t-1)*(t+1))/(t-1);

%SOLUTION CONSTANTS
BC     = (2.*er-I.*gr).*exp(+2.*I.*alpha);
B1     = rc.^4.*mc+rc.^4-1+mc;
B2     = rc.^4.*mc+rc.^4-mc+1;
B3     = rc.^4.*mc-mc-rc.^4+1;
B4     = -rc.^4.*mc-mc-rc.^4+1;
B5     = rc.^8.*mc-mc-rc.^8+1;

%RESOLUTION
rs     = 100;
ts     = 200;

%CLAST GRID IS FROM 1..rc
[rho, theta] = meshgrid(1:(rc-1)/rs:rc, 0:2*pi/ts:2*pi);
zeta_clast   = rho.*exp(i*theta);
p_clast      = real(-I.*mc.*B4./B1.*gr+2.*rc.^2.* ...
    (mc-1).*(I.*mc.*imag(BC)./B1-real(BC)./B2));
tau_clast    = -2.*mc.*rc.^4.*(I.*imag(BC)./B1+real(BC)./B2);
tau_clast    = sqrt((real(tau_clast)).^2 ...
    + (imag(tau_clast)).^2);

%Correct size of arrays
p_clast      = ones(size(rho))*p_clast;
tau_clast    = ones(size(rho))*tau_clast;

%MATRIX
[rho, theta] = meshgrid(rc:2*rc/rs:3*rc, 0:2*pi/ts:2*pi);
zeta_mat     = rho.*exp(i*theta);
```

```

p_mat          = -2.*rc.^2.*real(B3.*(-I.*imag(BC).*B2+ ...
                    real(BC).*B1)./(zeta_mat.^2-1)./B1./B2);
str_mat        = conj(zeta_mat+1./zeta_mat).* ...
                ((-I.*gr./zeta_mat.^3+2.*B3.*rc.^2.*(I.*imag(BC)./B1- ...
                    real(BC)./B2)./zeta_mat.^3)./(1-1./zeta_mat.^2)).^2- ...
                2.*(-1./2.*I.*gr.*(1-1./zeta_mat.^2))-B3.*rc.^2.* ...
                (I.*imag(BC)./B1-real(BC)./B2)./zeta_mat.^2)./(1- ...
                1./zeta_mat.^2)).^3./zeta_mat.^3)+(-real(BC)+I.*imag(BC)) ...
                .* (1-1./zeta_mat.^2))-B5.*(I.*imag(BC)./B1-real(BC) ...
                ./B2)./(zeta_mat.^3-zeta_mat).^2.*(3.*zeta_mat.^2-1))./ ...
                (1-1./zeta_mat.^2));

tau_mat        = sqrt((real(str_mat)).^2 + (imag(str_mat)).^2);

%TRANSLATE zeta -> z
z_clast        = zeta_clast+1./zeta_clast;
z_mat          = zeta_mat+1./zeta_mat;

%PLOT IN ZETA (IMAGE) DOMAIN
figure(1);
clf
subplot(211)
pcolor(real(zeta_clast), imag(zeta_clast), p_clast);
hold on;
pcolor(real(zeta_mat),    imag(zeta_mat),    p_mat);
shading interp;
axis image; axis off;
title('Pressure')
colorbar('vert')

subplot(212)
pcolor(real(zeta_clast), imag(zeta_clast), tau_clast);
hold on;
pcolor(real(zeta_mat),    imag(zeta_mat),    tau_mat);
shading interp;
axis image; axis off;
title('\tau')
colorbar('vert')

%PLOT IN Z (PHYSICAL) DOMAIN
figure(2);
clf
subplot(211)
pcolor(real(z_clast), imag(z_clast), p_clast);
hold on;
pcolor(real(z_mat),    imag(z_mat),    p_mat);
shading interp;
axis image; axis off;
title('Pressure')
colorbar('vert')

subplot(212)
pcolor(real(z_clast), imag(z_clast), tau_clast);
hold on;
pcolor(real(z_mat),    imag(z_mat),    tau_mat);
shading interp;
axis image; axis off;
%title('\tau')
colorbar('vert')

```

ELL_P_INTERF.M

```
=====
% ELL_P_INTERF.M
%
% Pressure in the matrix at the elliptical clast-matrix interface.
%
% 2002, Dani Schmid
=====

%CLEAR FIGURE
figure(1); clf

%COMPLEX NUMBER DEFINITION
I      = sqrt(-1);
i      = sqrt(-1);

%VISCOSITY CONTRAST BETWEEN CLAST AND MATRIX
mc     = 1e+3;

%FAR FIELD FLOW
er     = -1;
gr     = 0;
alpha  = 0/180*pi;

%ASPECT RATIO, t CANNOT BE <=1, USE t=1.001 FOR CIRCULAR INCLUSION
t      = [1.0001, 2, 10, 20];
Styles = {':k', '-.k', '--k', '-k'};

%CIRCUMFERENCE
theta  = 0:2*pi/360:2*pi;

for m=1:length(t)
    %TRANSLATE ASPECT RATIO INTO RADIUS
    rc  = sqrt((t(m)-1)*(t(m)+1))/(t(m)-1);
    %PRESSURE ON RADIUS
    press=(2.*mc.*rc.^4-2.*mc-2.*rc.^4+2).*rc.^2./...
    (mc.*rc.^4+mc-1+rc.^4)...
    ./((1+rc.^4+mc.*rc.^4-mc).*((rc.^2.*(cos(theta).^2-sin(theta) ...
    .^2)-1)./((rc.^2.*(cos(theta).^2-sin(theta).^2)-1).^2+4.*rc.^4 ...
    .*sin(theta).^2.*cos(theta).^2).*(-2.*er.*cos(2.*alpha)-gr ...
    .*sin(2.*alpha)).*(mc.*rc.^4+mc-1+rc.^4)+2.*rc.^2.*sin(theta) ...
    .*cos(theta)./((rc.^2.*(cos(theta).^2-sin(theta).^2)-1).^2+4.* ...
    rc.^4.*sin(theta).^2.*cos(theta).^2).*(-gr.*cos(2.*alpha)+ ...
    2.*er.*sin(2.*alpha)).*(1+rc.^4+mc.*rc.^4-mc));

    %PLOTTING
    plot(theta/pi*180, press, Styles{m});
    hold on;
end
achsen = axis;
axis([0 360 axsen(3:4)]);
set(gca, 'Xtick', [0:45:360])
grid on;

title('Pressure around elliptical inclusion');
xlabel('\theta'); ylabel('Pressure');
legend('t=1','t=2','t=10','t=20', -1);
```

ELL_ROT_RATE.M

```
%=====
% ELL_ROT_RATE.M
%
% Analytical formula for the rotation rate of an ellipse in
% combined, inclined simple & pure shear
%
% 2002, Dani Schmid
%=====
%CLEAR FIGURE
figure(1);
clf;

%FAR FIELD FLOW
er      = 0;
gr      = 1;
alpha   = -pi/2:(pi/2)/100:pi/2;

%ELLIPSE ASPECT RATIO
t       = 6;

%VISCOSITIES
mc      = [1e6,      1,  1/10, 1/100];
Styles  = {':k', '-.k', '--k', '-k'};

for m=1:length(mc)
    %ROTATION RATE
    rot_rate = (-1./2.*(t.^2-mc(m).*t.^2+mc(m)-1)./ ...
                (mc(m).*t.^2+mc(m)+2.*t).*cos(2.*alpha)-1./2).*gr ...
                -1./2.*(2.*mc(m).*t.^2-2.*t.^2-2.*mc(m)+2)./ ...
                (mc(m).*t.^2+mc(m)+2.*t).*sin(2.*alpha).*er;

    %PLOT
    plot(rot_rate, alpha/pi*180, Styles{m});
    hold on;
end
grid on;
xlabel('Rotation Rate/Shear Rate');
ylabel('\alpha');
legend('\mu_c/\mu_m=\infty', '\mu_c/\mu_m=1', ...
       '\mu_c/\mu_m=1/10', '\mu_c/\mu_m=1/100', -1);
```

ZHOUK_DEMO.M

```
%=====
% ZHOUK_DEMO.M
%
% Demonstration of the Joukowski transform
%
% 2002, Dani Schmid
%
%=====
%RESOLUTION
nt          = 200;
Theta       = 0:2*pi/nt:2*pi;

%SETUP THREE DIFFERENT CIRCLES
SLIT        = exp(i*Theta);
ELLE        = 2*exp(i*Theta);
JOUK        = 2*exp(i*Theta)-0.9696+i*0.3473;

%PLOT IN ZETA
figure(1)
clf
subplot(121)
hold on;
plot(real(SLIT), imag(SLIT), '-k');
plot(real(ELLE), imag(ELLE), '--k');
plot(real(JOUK), imag(JOUK), ':k');
axis equal
grid on
title('\zeta-Plane');

%TRANSFORM
SLIT        = SLIT + 1./SLIT;
ELLE        = ELLE + 1./ELLE;
JOUK        = JOUK + 1./JOUK;

%PLOT IN Z
subplot(122)
hold on;
plot(real(SLIT), imag(SLIT), '-k');
plot(real(ELLE), imag(ELLE), '--k');
plot(real(JOUK), imag(JOUK), ':k');
axis equal;
grid on
title('z-Plane');
```