MATLAB EXERCISES to “Fundamentals of rock physics”

MATLAB Exercise 1. Plotting the grain size distribution function

MATLAB code below uses `xlsread` operator to read data vectors from A, B, C columns in `'exercise_9.xlsx'` file. A column contain the minimum values for bins, B column contain the maximum values of bins , and C column contains the mass factions in grams. In order to copare the measured PDF with the normal PDF the function `normpdf` is used.

```
%%% Matlab example of plotting grain size distribution %function.
The path to data file should be open in Matlab directory. The grain
size data are in columns A (min of size bin) and in B the maximum of
a size bin, and in C %the mass fraction in gramm.
filename = 'exercise_9.xlsx';% specify file name
sheet = 1;% specify a sheet number in an excell-file
xlRange1 = 'a2:a16'; % specify a cell range for d min in %bins
xlRange2 = 'b2:b16'; % specify a cell range for d max in %bins
xlRange3 = 'c2:c16'; % specify cell range for mass in bins
dmin = xlsread(filename,sheet,xlRange1); %read data set1
dmax = xlsread(filename,sheet,xlRange2); %read data set2
mass = xlsread(filename,sheet,xlRange3); %read data set3
smass=sum(mass); mass100=mass/smass; % normalize mass %fraction in %
each bin
dmean=(dmax+dmin)/2;% calculate a mean grain size point in a bin
fi=-log2(dmean); %conversion into fi-units
figure;
bar(fi, 100*mass100,1,'r')% plot a histogram
xlabel(' \phi = log_2(d, mm)'); ylabel('mass, %') % axis %labelling
xticks('auto'); yticks('auto'); hold on % plotting ticks
 m=dot(mass100,fi);
 dm=2^(-m)% mean average size in mm
   v=dot(mass100,(fi-m).^2) %calculating variance
 s=sqrt(v)% calculating standard deviation
  Sk=dot(mass100,(fi-m).^3)/s^3 %skewness
   Kurtosis=dot(mass100,(fi-m).^4)/v^2 % kurtosis
 X = linspace(-log2(0.01),-log2(20));% define x-axe for %plot
 norm = 100*normpdf(X,m,s);%calculate a normal PDF%
plot(X,norm,'-.rd')% plot a normal PDF
 legend('grain size hystogramm','normal distribution')% %define
legend labels
hold on
pdfnrm = @(x,b) 1./(b(2)*sqrt(2*pi)) .* exp(-((x-
b(1)).^2./(2*b(2).^2))); % definition of a target function
% b(1)= m; b(2)=s; %Initialisation of fitting %parameters
SSECF = @(b) sum((mass100-pdfnrm(fi,b)).^2);% Sum-Squared-%Error
Cost Function
[B,SSE] = fminsearch(SSECF, [m, s]);% minimization of Sum-%Squared
Error Function, B is the fitting parameter vector
plot(X,100*pdfnrm(X,B),'bl') B % Plot fitted PDF
hold off
grid
```

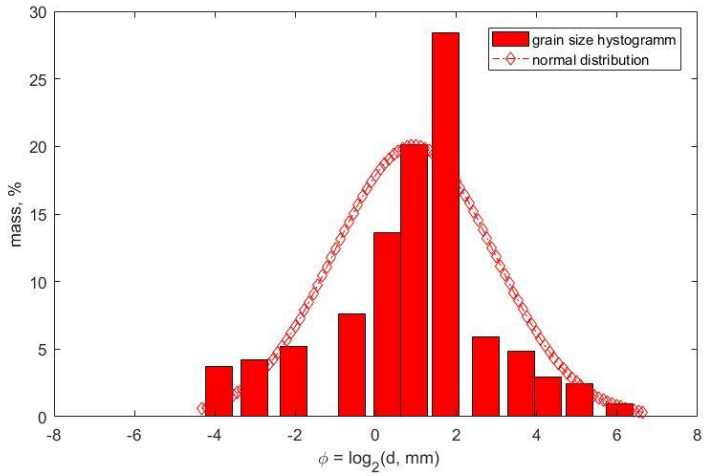


Fig. ME1.1 The histogram of the grain size distribution function. Alternatively, the histogram can be fitted directly to the shape of the normal PDF using the least square procedure as follows:

The results of fitting including empty bins could be slightly different, the mean size 0.46 mm, the standard deviation of the mean ϕ is 1.87.

MATLAB Exercise 2.1 Monte-Carlo simulation of rock density

Monte-Carlo simulations are computer experimentation methods in which a random choice of input parameters is used in order to calculate statistics of output parameters. It is a very useful statistical tool and widely used both in non-engineering and engineering fields. In the base of computational experiments lies random sampling and large number of computer runs. Then, the mean value and standard deviation or PDF of model outputs are estimated. MATLAB provides random number generators for commonly used PDFs as follows:
R = normrnd(MU,SIGMA) is the normal or Gaussian distribution
R = lognrnd (MU,SIGMA) is the lognormal distribution,
R = unifrnd (A,B) is the uniform probability distribution between A and B values with zero outside probability (A,B),
`pd = makedist('Triangular','a',A,'b',B,'c',C);`
`R = random(pd,Number of points),` is the triangular probability distribution within the interval (A,C) having the maximum probability at B, and zero probability outside (A,C). (see Fig. ME2.1)

MATLAB code below presents an example of Monte-Carlo simulation for rock density using volumetric % of modal mineral composition.

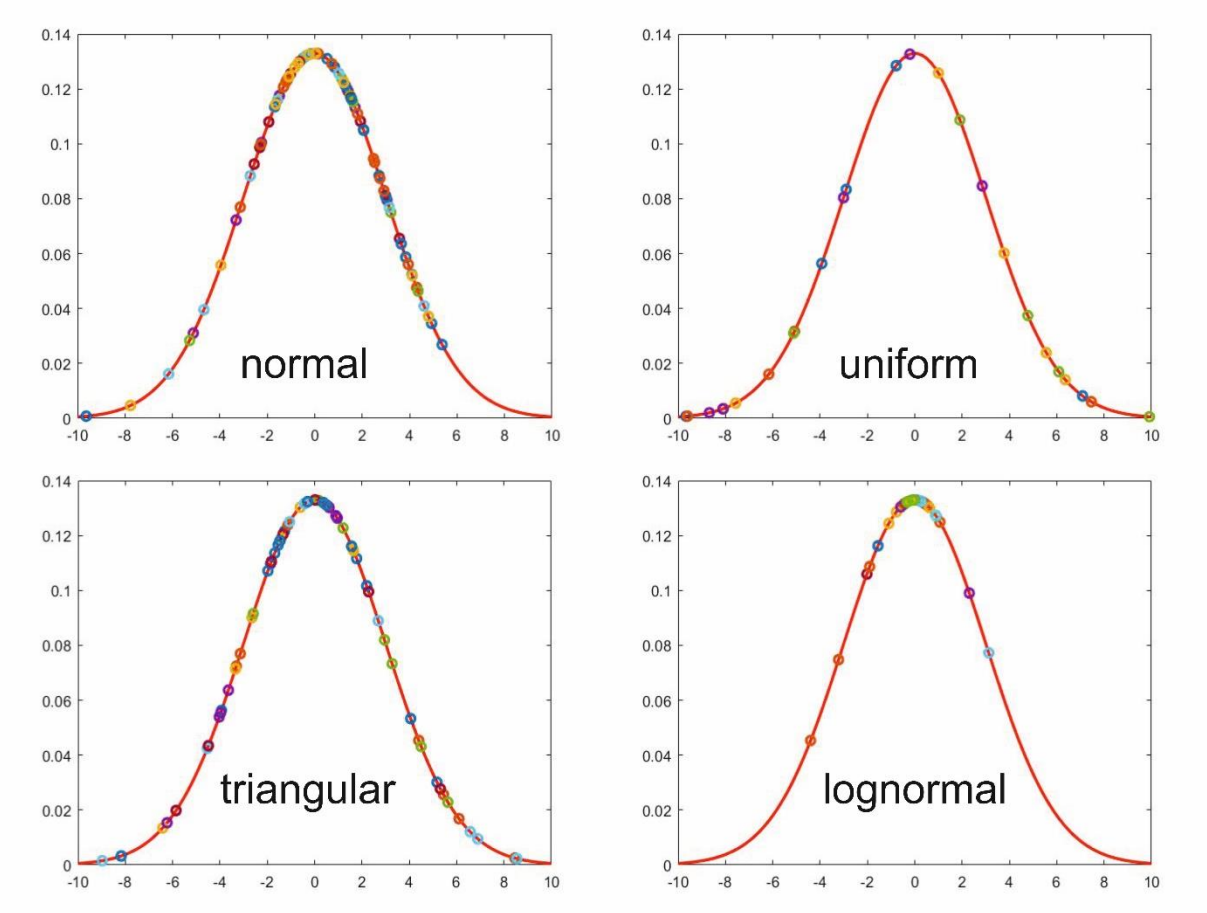


Fig. ME2.1 Illustration of different random PDF. The arguments for plotting normal PDF have been chosen by using differing random number generators. When the average and standard deviation of some mineral composition are known, the normal PDF random generator may be used. When only max and min vol. fractions of modal minerals have been estimated, then the uniform PDF random choice is applicable. Points on curves indicate how dense data are located to the mean value for these four PDFs.

```
% Matlab Example 2 Monte-Carlo simulation
%Density of crustal rocks
clear all
close all
N_MC=10000 % Number of Monte-Carlo simulations
% read the data from excell-file *.xlsx;
[xlsfile,path2xls] = uigetfile('*.xlsx', 'Please, choose the data
file'); %open the path to the file in PC
[data,text] = xlsread(fullfile(path2xls,xlsfile)); % read the
%excell file
N_lines=length(data(:,1));%defines number of lines in the file
for n=1:4:N_lines
 n_end=length(data(n,:));% define the length of a line
vol_fr = data(n,1:n_end);% read max/min vol% of minerals
%in the line n from the 1-st to n_end-th column
dens_miner=data(n+1,1:n_end); % read max/min density of %minerals %
in the line n+1 from the column 1 to n_end-th %column
for j=1:N_MC
summe_comp=0;% initialization of summation
```

```
r_vol=unifrnd(0,1:1,n_end/2,1); %random n_end/2 numbers in the
%interval 0,1 for vor_fr
r_dens=unifrnd(0,1:1,n_end/2,1); %random n_end/2 numbers in %the
interval 0,1 for dens_mine
for i=1:n_end/2
Rand_comp(i) = vol_fr(2*i)+(vol_fr(2*i-1)-vol_fr(2*i))*r_vol(i);
%random chose of vol% in the range from %max vol% and min vol%
Rand_dens(i)=dens_miner(2*i)+(dens_miner(2*i-1)-
dens_miner(2*i))*r_dens(i);%random chose of%vol% in the range %from
max vol% and min vol%
summe_comp=summe_comp+Rand_comp(i);%summation of random vol% %
end
Rand_comp=Rand_comp/summe_comp; %normalization of vol% in %
%fraction von 0 bis 1
dens(j)=dot(Rand_comp,Rand_dens);% calculate the average % density
end
rock_number=n
dens_mean=mean(dens)
dens_sigma=std(dens)
end
```

MATLAB Exercise 2.2 Expansion of functions in Taylor series

Let *f(x)* be the function of argument *x* which is differentiated *n + 1* times, i.e. the derivative $f(x)^{(n+1)}$ exists at point *x* which belongs to interval $(x_0-\alpha, x_0 + \alpha)$, where $\alpha>0$ is a small positive number. Then, for all *x* belonging to this interval, there is a unique Taylor expansion of *f(x)* in accordance with the formula:

$$f(x) \quad = \sum_{k=0}^{n} \frac{f(x_0)^{(k)}}{k!} \cdot (x - x_0)^k + R_n(x) \qquad \text{(ME 2.2.1)},$$

where $R_n(x)$ is the remaining *n-th* term of the series, which can be estimated trough the *(n+1)-th* derivative of $f(x_0+(x-x_0)\cdot\theta)^{(n+1)}$ $(0<\theta<1)$ at any point belonging to the interval (x_0, x):

$$R_n(x) = \frac{f(x_0+(x-x_0)\cdot\theta)^{(n+1)}}{(n+1)!} \cdot (x - x_0)^{n+1} \qquad \text{(ME 2.2.2)},$$

the so called remaining term in the form of Lagrange. When $\lim_{n\to\infty} R_n(x) \quad = 0$, for any *x* belonging to the interval $(x_0-\alpha, x_0 + \alpha)$, then, the function *f(x)* can be approximated by a polynomial (ME 2.2.1) of the *n-th* degree.

The graphical illustration of the Taylor expansion is rather simple. Using only the first term of the Taylor series one approximates the function *f(x)* with the tangential straight line to the function at $x=x_0$. Using two terms in the series one approximates the function with a parabola tangential curve to *f(x)* at x_0, and so on (see Fig. FB 2.2.1). When $x_0=0$, the expansion series are called the *Maclaurin series*.

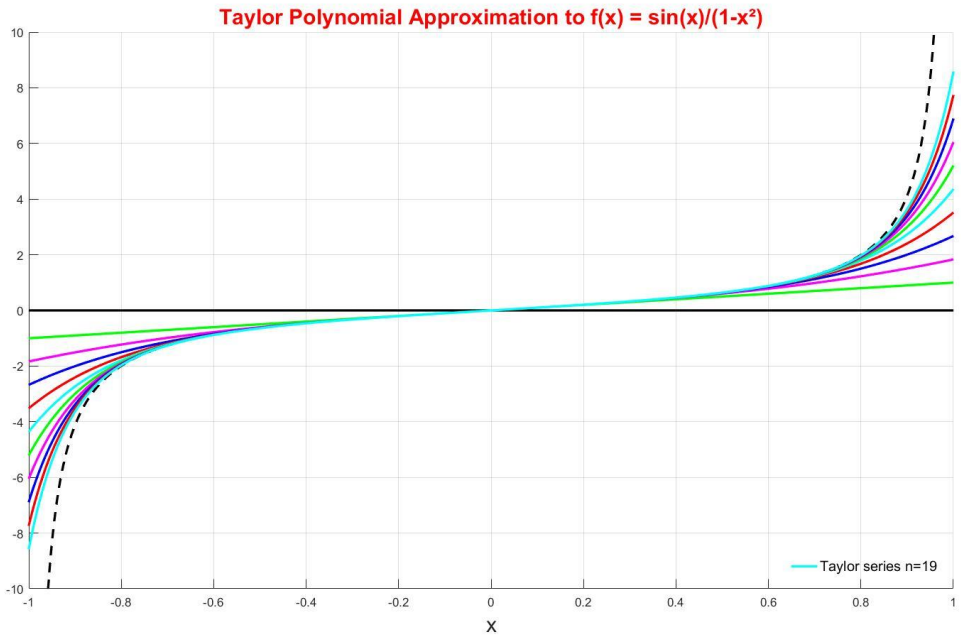


Fig. ME2.2 Approximation of $f(x) = \frac{\sin(x)}{1-x^2}$ with the Taylor series.

```
function sinx_1_x2_movie
%          SIN(x)/(1-x²)
%MOVIE  Taylor polynomial approximations to y = sin x/(1-x²) %   are
presented in a graphical form. Successive graphs
%   of Taylor polynomials of degrees 1, 3, 5, 7, 9,
%   11, 13, 15, 17, 19 are superimposed.
%   As the degree increases the Taylor polynomials wrap
%   themselves onto the sin(x)/(1-x²) curve over longer intervals.
%   modified from David R. Hill, Math Dept, Temple
%  University Philadelphia
s0='    ';
s1='Taylor Polynomial Approximation to f(x) = sin(x)/(1-x²)';
s2='Press enter to continue.';
%set colors for graphs
colorch=['g' 'm' 'b' 'r' 'c'];colorch=[colorch colorch];
%The function definition:
f='sin(x)./(1-x.^2)';
%The terms of the Tayor polynomial
maxdeg=9; %max degree of Taylor poly is 2*maxdeg + 1
timdelay=5; %time delay for pause in sec
x=[-pi/2:pi/1000:pi/2]';fx=eval(f);
clc
disp([blanks(15) 'SIN(x)/(1-x²) MOVIE - Taylor Polynomial
Approximation']);
disp(s0),help sinx_1_x2_movie, disp(s1),disp(s2),pause
for k = 0:maxdeg
    d=0;
    for i=0:k
    d=d+(-1)^i/prod([1:(2*i+1)]);% Calculate Taylor series
%coefficients
    end
```

```
    g = 'd*x.^(2*k+1)'; z = eval(g); Calculate k-th term in %Taylor
series
    if k==0,y=z;else,[m,n]=size(y);y=[y y(:,n)+z];end
end
%Begin graphing
hfig=figure('units','normal','position',[0 0 1 1],'color','white');
axis([-1 1 -10 10]),axis(axis), grid on, hold on
hh=plot(x,fx,'--k',x,zeros(size(x)),'-k','linewidth',2)
title(s1,'fontsize',18,'color','r')
xlabel('x', 'FontSize',18)
legend(hh,{'f(x)=sin(x)/(1-x²)', 'f(x)=0'},'Location',
'northeast','FontSize',12)
hold on, pause(timdelay)
for j=0:maxdeg
    k=2*j+1;cl=['-' colorch(j+1)];
    h=plot(x,y(:,j+1),cl,'linewidth',2)
legend(h,{['Taylor series n='
int2str(k)]},'Location','southeast','Box','off','FontSize',12)
    grid on, hold on, pause(timdelay)
end
pause
end
```

The example presented above dealing with the function *f(x)=sin(x)/(1-x²)*. The function possesses two singularities at *x*=±*1*. The coefficients of the Taylor series at x_0=0 may be calculated as follows:

$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} = \sum_{i=0}^{\infty}\left(\frac{(-1)^i}{(2i+1)!}\cdot x^{2i+1}\right)$ (ME2.2.3),

and for *f(x)* at *x* close 0 there are two infinite series which should satisfy the equation:

$[\sum_{n=0}^{\infty}(a_n \cdot x^{kn}))]\cdot(1-x^2) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$ (ME2.2.3).

By equating the cofficients at the same power of two series in the left and right sides one gets the reccurent formula:

$$a_0 = 0, a_{2i} = 0;\ a_{2i+1} - a_{2i-1} = \frac{(-1)^i}{(2i+1)!}, \qquad a_{2i+1} = \sum_{i=0}^{\infty}\left(\frac{(-1)^i}{(2i+1)!}\right)$$

Thus, the Taylor series of *sin(x)/(1-x²)* at x_0=*0* is expanded as follows:

$$\frac{sin(x)}{1-x^2} = \sum_{n=0}^{\infty}\{x^{2n+1}\cdot\sum_{i=0}^{n}\left(\frac{(-1)^i}{(2i+1)!}\right)\}.$$

With the increase of the Taylor series polynomial degree *n*, one can approximate *f(x)* in broader interval *x*, closer and closer to singularity points (Fig. ME2.2).

MATLAB Exercise 3. Stress-strain curves: strength and yield stress

The example data containing deformation of a rectangular bar specimen in mm, load F in N, and geometric parameters in mm, length L, width W and thickness h are inputs of the code and saved in the text-file 'strength1.txt'. In order to read the file the function `textscan` is used. The empty cells in columns are filled with zeros. To find the maximum value on the function plot [*max, indexmax*]=`max( )` is applied, which defines also the index of the function argument corresponding to the maximum value. The span of arguments between 0 and *indexmax* is fitted to the linear regression having the intercept constant 0 by using the function `dlm=fitlm(..,Intercept, 'False')`. The parameter `dlm.Rsquared.Ordinary` is the extracted from linear fit regression coefficient R and

`coeffs=dlm.Coefficients.Estimate` is the slope coefficient. The span of fitting arguments consequently decreases until the regression coefficient R is close to 1 within a given error (`err=0.02` and desired `R=0.98` in the code below) using `while` loop. The stress point corresponding to the right hand side limit value of the linear deformation span is the yield stress (Fig. ME3.1).

```
%The MATLAB file with the data of deformation in mm, load in N, length,
%wide and thickness of a bar sample is processed to obtain:
%Proportional Limit of Deformation; Ultimate Strength; Yield Stress;
clc; clear all
fid=fopen('strength1.txt');%Load and open text file
s=textscan(fid,'%f %f %f %f %f','emptyValue', 0);
%read 5 columns in txt-file
fclose(fid);
L=s{3}; %read L, input length in mm from the third column
L=dot(ones(1,length(L)),L); %making a scalar value of L
W=s{4}; %read W, input width in mm from the fourth column
h=s{5}; %read h, input thickness in mm from the fifth column
A=dot(W,h) %calculating cross-section area in mm^2
F = s{2};% input load in N from the second column
delta = s{1}; % input deformation in mm from the first column
%Calculation of deformation in % and stress in MPa
stress=F./A; % in MPa
eps=100*delta./L; % deformation in %
%Plotting Load-Deformation Curve
subplot(2,1,1)
hold on; grid on; plot(delta,F,'--b'); title('Load versus Deformation')
xlabel('\Delta, mm'); ylabel('Load F, N')
%Plotting Stress-Deformation Curve
subplot(2,1,2)
hold on; grid on; plot(eps,stress,'-.r')
title('Stress Versus Deformation in %'); xlabel('\epsilon, %');
ylabel('\sigma, MPa')
hold on
%calculating strength and yield stress from the curve
[maxstrength, imaxstrength] = max(stress);%estimation of the index of max
maxeps = eps(imaxstrength); %estimation of eps corresponding to strength
plot(maxeps,maxstrength,'o-
','MarkerFaceColor','red','MarkerEdgeColor','black')
text(maxeps,maxstrength,'  \leftarrow strength') %put the text on the graph
hold on
R=0;err=0.02;imaxstrength0=imaxstrength;
%err is a deviation error from linearity
while R<(1-err)
   imaxstrength=imaxstrength-1;% one step down in eps array index
dlm= fitlm(eps(1:imaxstrength),stress(1:imaxstrength),'Intercept',false);
R=dlm.Rsquared.Ordinary;%extracting correlation coefficient R
end
yieldstress=stress(imaxstrength;
% yield stress corresponds to deviation
%of linear fit from 1 with a given error
yieldeps=eps(imaxstrength)% estimation of eps corresponding to yield stress
coeffs=dlm.Coefficients.Estimate;  % the coefficient of the linear fit
plot(yieldeps,yieldstress,'o-
','MarkerFaceColor','green','MarkerEdgeColor','black')
text(yieldeps,yieldstress,'  \leftarrow yield stress'); hold on
plot(eps(1:imaxstrength0),coeffs(1).*eps(1:imaxstrength0),'g'); hold on
span=find(eps>=maxstrength/coeffs(1)& eps<=maxeps);
plot(eps(span),coeffs(1).*eps(span)-maxstrength,'b');
plot(eps(span(1)),0,'o-
..','MarkerFaceColor','blue','MarkerEdgeColor','black')
```

```
text(eps(span(1)),0,'  \leftarrow plastic deformation');% place the text
hold on
```

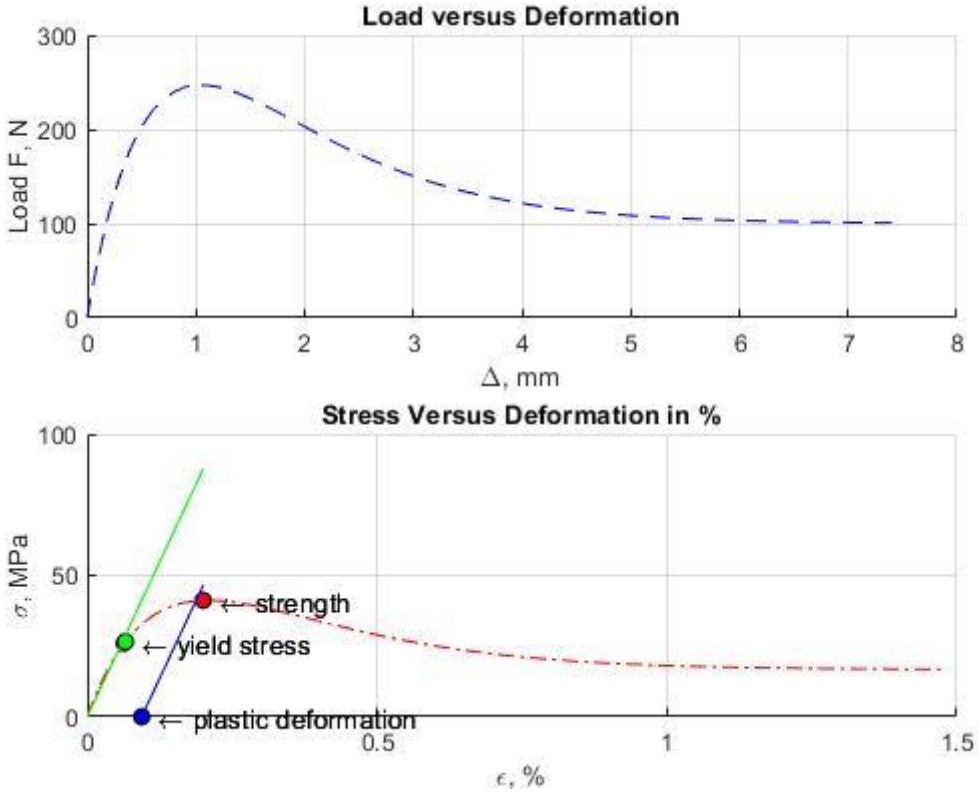


Fig. ME 3.1 MATLAB generated curves of a specimen deformation plotted together with yield stress and strength estimations: green line is the linear fit having the intercept =0 with e ordinate axis.

MATLAB Exercise 4. Differential effective medium approach: Eshelby tensor

The differential effective medium (DEM) approach is the evolution scheme to calculate effective elastic parameters of a medium by introducing incrementally of spherical pore volume (i. e. through the incremental increase of porosity $\delta\varphi$), and computing the corresponding incremental change in effective elastic moduli. The procedure is repeated until the target porosity is reached (Brantut&David, 2019). In this MATLAB exercise this approach will be illustrated by calculations of the Poisson's ratio as a function of porosity. The solid matrix material and the resulted effective medium are assumed isotropic and spheroidal inclusions (with half axis $\breve{\alpha} = [a_1, a_2, a_3]$) are uniformly distributed and oriented parallel to their largest half axis in matrix. For the effective bulk ($\bar{K}$) and shear ($\bar{\mu}$) moduli there are two coupled ordinary differential equations (e.g. David, 2012):

$\frac{d\bar{K}}{\bar{K}} = -\frac{d\varphi}{1-\varphi} \cdot P(\breve{\alpha}.\nu)$ and $\frac{d\bar{\mu}}{\bar{\mu}} = -\frac{d\varphi}{1-\varphi} \cdot Q(\breve{\alpha}.\nu)$ (ME 4.1),

where $P(\alpha.\nu)$ and $Q(\alpha.\nu)$ are the bulk and shear compliances of spheroidal voids, indicated in Chapter (4.40a&b), respectively. They depend on the Poisson's ratio ν of solid matrix, and the aspect ratio α of spheroids. The Poisson's ratio ν is given by the relationship:

$2 \cdot \nu = \frac{3\bar{K}-2\bar{\mu}}{3\bar{K}+\bar{\mu}}$ or $2 \cdot d\nu = d\left(\frac{3\bar{K}-2\bar{\mu}}{3\bar{K}+\bar{\mu}}\right) = \frac{9\bar{K}\cdot\bar{\mu}\cdot}{(3\bar{K}+\bar{\mu})^2} \cdot \left[\frac{d\bar{K}}{\bar{K}} - \frac{d\bar{\mu}}{\bar{\mu}}\right]$, and

$d\nu = -\frac{(1+\nu)\cdot(1-2\nu)}{3} \cdot \frac{d\varphi}{1-\varphi} \cdot [P(\breve{\alpha}.\nu) - Q(\breve{\alpha}.\nu)]$ (ME 4.2).

In the case when inclusions are filled with compressible fluid $\mu_2 = 0$ and $\frac{K_2}{K_1} = \zeta$

$$P(\breve{\alpha}.\nu,\zeta) = \frac{K_1-\zeta\cdot K_1}{K_1}\cdot\frac{1}{1-\frac{K_1-\zeta\cdot K_1}{K_1}\cdot\sum_{i=1}^{3}s_{1i}} = \frac{1-\zeta}{1-(1-\zeta)\sum_{i=1}^{3}s_{1i}.}$$

$$Q(\breve{\alpha}.\nu) = \frac{1}{1-2\cdot s_{44}} \qquad \text{(ME 4.3).}$$

where $\sum_{i=1}^{3} s_{1i}$ is the bulk compliance and $2\cdot s_{44}$ is the shear compliance of inclusions, and s_{ij} are the elements of the compliance Eshelby tensor matrix (*i.e.* Meng et al., 2012):

$$\begin{aligned} s_{11} &= \frac{a_1\cdot a_2\cdot a_3}{4\cdot(1-\nu)}\cdot[3a_1^2\cdot J_{11} + (1-2\nu)\cdot J_1] \\ s_{12} &= \frac{a_1\cdot a_2\cdot a_3}{4\cdot(1-\nu)}\cdot[a_2^2\cdot J_{12} - (1-2\nu)\cdot J_1] \\ s_{13} &= \frac{a_1\cdot a_2\cdot a_3}{4\cdot(1-\nu)}\cdot[a_3^2\cdot J_{13} - (1-2\nu)\cdot J_1] \end{aligned} \qquad \text{(ME 4.4).}$$

$$2\cdot s_{44} = \frac{a_1\cdot a_2\cdot a_3}{4\cdot(1-\nu)}\cdot[(a_1^2+a_2^2)\cdot J_{12} + (1-2\nu)\cdot(J_1+J_2)]$$

More sophisticated than (ME4.3) dependence of $P(\breve{\alpha}.\nu,\zeta)$ on ζ has been considered in (Brantut & David, 2019). The constants J_j and J_{ij} depend only on inclusion geometry and for ellipsoidal cavities are given by the integrals.

$$J_j = \int_0^\infty \frac{d\xi}{\left(a_j^2+\xi\right)\cdot\prod_{j=1}^{3}\left(a_j^2+\xi\right)^{1/2}},$$

$$J_{1j} = \int_0^\infty \frac{d\xi}{\left(a_1^2+\xi\right)\cdot\left(a_j^2+\xi\right)\cdot\prod_{j=1}^{3}\left(a_j^2+\xi\right)^{1/2}}, \qquad \text{(ME 4.5).}$$

In the case of spherical inclusions $\breve{\alpha} = [1,1,1]$, the results of integration are $J_j = -\frac{2}{3\cdot(1+\xi)^{\frac{3}{2}}}\Big]_0^\infty = \frac{2}{3}$ and $J_{1j} = -\frac{2}{5\cdot(1+\xi)^{\frac{5}{2}}}\Big]_0^\infty = \frac{2}{5}$. Finally for the spherical case one obtains $s_{11} = \frac{7-5\nu}{15\cdot(1-\nu)}$, $s_{12} = s_{13} = \frac{5\nu-1}{15\cdot(1-\nu)}$, $2s_{44} = \frac{2\cdot(4-5\nu)}{15\cdot(1-\nu)}$, $\sum_{i=1}^{3} s_{1i} = \frac{1+\nu}{3\cdot(1-\nu)}$, and for the dry compressibility and shear compliances are: $P(\breve{1}.\nu,0) = \frac{3\cdot(1-\nu)}{2\cdot(1-2\nu)}$, $Q(\breve{1}.\nu) = \frac{15\cdot(1-\nu)}{7-5\nu}$.

Here below is MATLAB code to calculate J-integrals (ME 4.5) and use them for calculation of *P* and *Q* for arbitrary α and ν. The evolution scheme starts with the initial value of the Poisson's ratio ν_0 at zero porosity $\varphi = 0$, and for each small step of $\delta\varphi$ the increment of n is given by (ME 4.2).

The loop *while end*, marked by grey colour, defines the evolution scheme of calculations, from the previously calculated point a new point is determined.

```
%Calculations of Poisson's ratio after Brantut%David, 2019 using Eshelby
%tensor components
nu0=0.2;%input initial Poisson's ratio at zero porosity
zeta=0.001; %input zeta=k2/k1
alpha=[10,5,1];%input half axis of ellipsoid, first element is the maximum
an=alpha(1); alpha=alpha./an;% normalization to the maximum axis
fi0=0; dfi=0.001;% define initial porosity and porosity step
fi=fi0; nu=nu0;%initialization of the evolution scheme
while fi<0.55 %maximum porosity 55%
    delta=(Q(alpha,nu)-P(alpha,nu,zeta))*(1+nu)*(1-2*nu)/3;
    dnu=delta/(1-fi)*dfi;%calculate increament of Poisson's ratio
        plot(fi,nu,'r.')%put the data point on a graph
        grid on
        hold on
        xlabel('\phi') %put label on horizontaö axis
ylabel('\nu')%put label on vertical axis
set(gca,'fontsize', 18);%define the font size
nu= nu+dnu;fi=fi+dfi; % Poisson's ratio and porosity at the next step
end
hold on
```

```
grid on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function p=P(a,nu,z)% function to calculate bulk compliance
S11=3*a(1)^2*Jot(a, 2,0,0)+(1-2*nu)*Jot(a, 1,0,0);
S12=a(2)^2*Jot(a, 1,1,0)-(1-2*nu)*Jot(a, 1,0,0);
S13=a(3)^2*Jot(a, 1,0,1)-(1-2*nu)*Jot(a, 1,0,0);
S1j=a(1)*a(2)*a(3)/4/(1-nu)*(S11+S12+S13);
p=(1-z)/(1-(1-z)*S1j);
end
function q=Q(a,nu)% function to calculate bulk compliance
S44=(a(1)^2+a(2)^2)*Jot(a,1,1,0)+(1-2*nu)*(Jot(a,1,0,0)+Jot(a,0,1,0));
S44=a(1)*a(2)*a(3)/4/(1-nu)*S44;
q=1/(1-S44);
end
function Jij=Jot(alpha,j,k,l)
a=alpha(1); b=alpha(2); c=alpha(3);
fun = @(x,a,b,c,j,k,l)
……   1./(a.^2+x).^(j+1/2)./(b.^2+x).^(k+1/2)./(c.^2+x).^(l+1/2);
Jij=integral(@(x)fun(x,a,b,c,j,k,l),0,Inf, 'RelTol',1e-8,'AbsTol',1e-13);
end
```

MATLAB Exercise 5. Plotting velocity profile of fluid flow in elliptic and triangular channels

The MATLAB code below used formulas for viscous flow in an elliptic, triangular and four sided cusped pipes explained in Focus Box 5. These 3 situation are arranged in the "case" menu structure. For the input of parameter *X* the operator *X=input('test appears on screen'*) is used. The velocity field *v(x,y)* is calculated for *[–a<x<+a, –b<y<+b*] in the elliptic pipe case, but velocity *v*-values outside the ellipse are declared as *non-a-number* (*NaN*). Two vectors *x* and *y* consisting of *N* points are used to form the matrix *meshgrid*. The results of calculations are presented in 3 figures shown below. The *stem3* plot presents the data in the discrete form, *contourf* plots are 2D maps of velocity field with isoclines of velocity *v*, and *meshc* graph presents the velocity field in 3D (Fig. ME 5.1).

```
%***************************************************************************
*****************
%-----------------------------------------------------------------------
-----
           %Calculation of velocity Vmean,Vmax and volumetric flow
rate Q
       %adapted from Hatem Ali (2020). Fluid Mechanics: Flow in
Straight %  %           Pipes and its Extensions
              %MATLAB Central File Exchange.
              disp('Flow in a non-circular pipe ')
column={'Elliptic-cross section';'Equilateral triangular pipe';'4-
sided cusped pipe'};
Symbols=[1;2;3];
inputs={'pressure between two points, viscosity and two half axis of
the pipe';...
    'pressure between two points, viscosity, and the size of the
triangle pipe';...
    'pressure between two point, viscosity, size  of the cusped
pipe'};
Outputs={'velocity profile';'Q flow rate';'flow mean velocity'};
Information={'You can calculate elliptic geometry by pressing 1';...
```

```
    'You can calculate triangular geometry by pressing 2';...
    'You can calculate 4-sided cusped geometry by pressing 3'};
Start=table(Symbols,'RowNames',column)
inputs_Outputs=table(inputs,Outputs,'RowNames',column)
Informations=table(Information,'RowNames',column)
    s=input('what do you want to calculate (1,2,3)?  ');
    N=201;
    switch s
        case 1              % Elliptic geometry
   %---------------------Input of paramets of elleptic flow:
        Pa=input('Pressure at point a in Pa    ');
        Pb=input('Pressure at point b in Pb   ');
        mu=input('Viscosity of liquid in Pa.sec   ');
        a=input('Half axis x of ellptic pipe in m   ');
        b=input('Half axis y of ellptic pipe in m   ');
        l=input('Length of pipe in m   ');
        %-----------------------Print Table
        Value=[Pa;Pb;mu;a; b;l];
        Units=char({'Pa';'Pa';'Pa.sec';;'m';'m';'m'});
Information=table(Value,Units,'RowNames',{'Pa';'Pb';'Viscosity';'a';
'b';…
…'Length'})
        %------------Define vectors x and y, calculate velocity
                if  Pa==Pb % exclude zero pressure gradient
            disp('There is No Pressure Drop in the Pipe')
        else
            x=linspace(-a,a,N);y=linspace(-b,b,N);S= pi*a*b;
            [X,Y]=meshgrid(x,y);
            Vmax=(abs(Pa-Pb))*(a^2)*(b^2)/(a^2+b^2)/(4*mu*l);
            V=Vmax*(1-(X.^2)/a^2-(Y.^2)/b^2);V(V<0)=NaN;
            % outside of ellipse velocity is not a number
            Q=Vmax/2*S; Vmean=Vmax/2;
                end
            %-----------------------------------------------------
-----
        case 2     %triangular geometry
         Pa=input('Pressure at point a in Pa    ');
        Pb=input('Pressure at point b in Pb   ');
        mu=input('Viscosity of liquid in Pa.sec   ');
        a=input('Side length of triangular pipe in m   ');
                l=input('Length of pipe in m   ');
                if  Pa==Pb
            disp('There is No Pressure Drop in the Pipe')
                else
        x=linspace(-a/2,a/2,N);y=linspace(0,sqrt(3)/2*a,N);
        S=sqrt(3)/4*a^2; [X,Y]=meshgrid(x,y);
        Vmax=abs(Pa-Pb)/mu/l*a^2/36;
        V=Vmax*18*sqrt(3)/a^3*Y.*((a/2-Y./sqrt(3)).^2-
X.^2);V(V<0)=NaN;
        Q=Vmax*9/80*sqrt(3)*a^2; Vmean=Q/S;
                end
                %-----------------------------------------------------
------
        case 3 %four-sided cusped geometry after Lenker, 2007
            Pa=input('Pressure at point a in Pa    ');
        Pb=input('Pressure at point b in Pb   ');
```

```
        mu=input('Viscosity of liquid in Pa.sec   ');
        a=input('Size of cusped pipe in m  ');
        l=input('Length of pipe in m   ');
        if  Pa==Pb
            disp('There is No Pressure Drop in the Pipe!')
        else
                x=linspace(-a,a,N); y=x;[X,Y]=meshgrid(x,y);
                Vmax=abs(Pa-Pb)/l/mu*a^2/8;
        V=Vmax*((X.^2+Y.^2-a^2).^2-
8*(X.^2).*(Y.^2))./a^4;V(V<0)=NaN;
                 R=@(theta,r) r;
Vpolar = @(theta,r) Vmax*(r.^4.*cos(4*theta)-
2.*a^2.*r.^2+a^4)./a^4.*r;
   rmax = @(theta) a./sqrt(1+sqrt(2)*sin(2*theta));
   S= 4*quad2d(R,0,pi/2,0,rmax,'AbsTol',1e-6,'MaxFunEvals', 5000)
   Q = 4*quad2d(Vpolar,0,pi/2,0,rmax,'AbsTol',1e-6,'MaxFunEvals',
5000);
   Vmean=Q/S;
            end
    end
            %---------Presents results as Table
            Value=[Vmax;Vmean;Q];
            Units=char({'m/sec';'m/sec';'m^3/sec'});

Sumarry=table(Value,Units,'RowNames',{'Vmax';'Vmean';'Q'})
            v=Vmax*linspace(0,1,5);v=round(v,3);%define isoline
vector
            %-----------Plot using stem3 (digital presentation of
velocity
            figure(1)
            stem3(X,Y,V,':or','MarkerSize',1);hold on;grid on;
xlabel('x-distance (m)'), ylabel(' y-distance (m)'),
zlabel(' Velocity (m/s)');
title(' Velocity
profile');legend({'V(x,y)'},'Location','southeast');
            %-------Presents results as a contour map
            figure(2)
            [C,h]=contourf(X,Y,V,v,'-y'); colormap copper;
            clabel(C,h,v,'FontSize',8,'Color','white');
            c=colorbar; c.Label.String = 'Velocity in m/sec';
            grid on; xlabel('D-X(m)');
            ylabel('D-Y(m)');
            title(' Velocity cross-sectional profile (m/s)');
            %--------Presents results in 3D
            figure(3)
            meshc(V,X,Y)
            title('3-D of flow');zlabel('X-Diameter in m');
            ylabel('Y-Diameter in m'); xlabel('Velocity in m/s')
            %-----------------------------------------------------
```

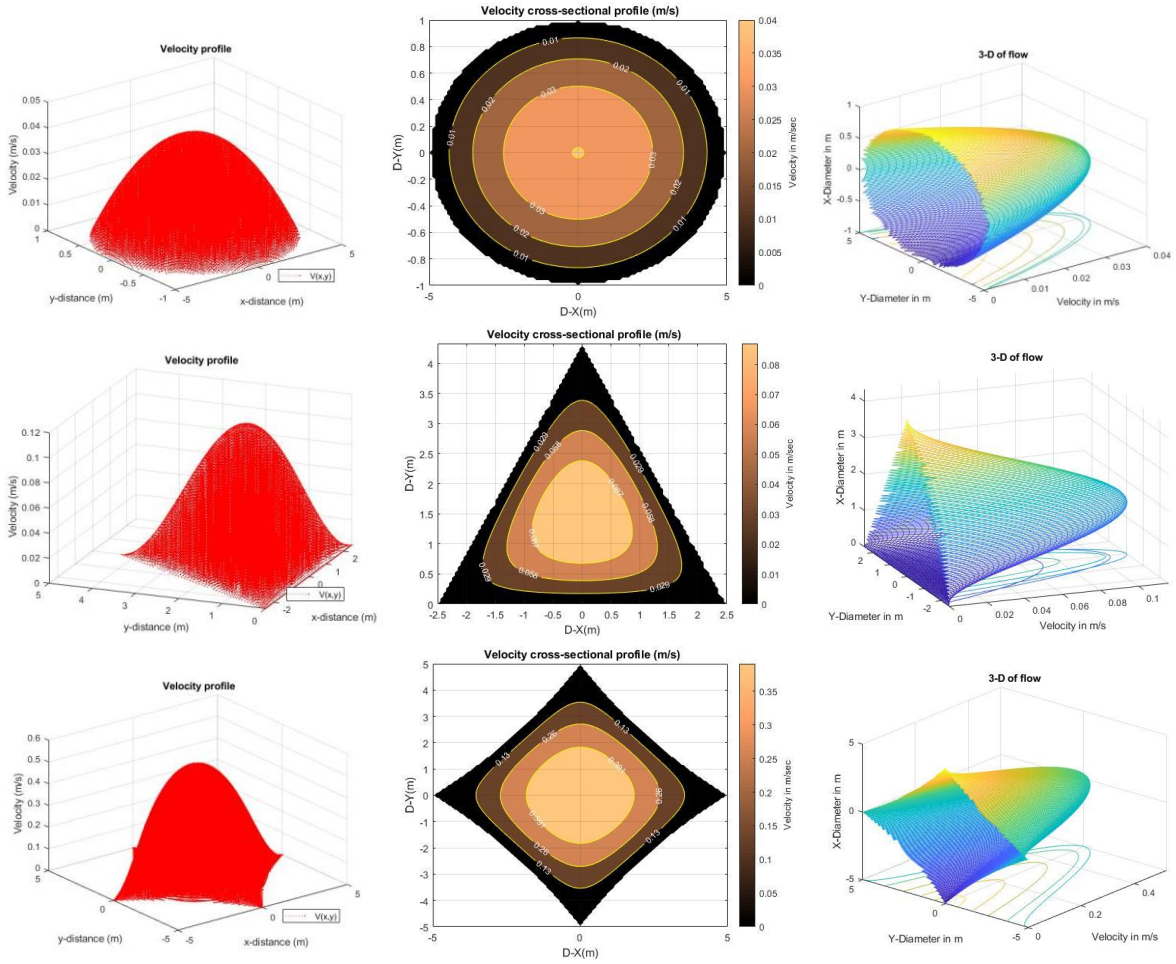


Fig. ME 5.1 Example of viscous flow by MATLAB calculations for pressure gradient =0.17 P/m , viscosity =1 Pa s. (upper panels) Elliptic pipe a=5 m, b=1 m;. (middle panels) Equilateral triangular tube with side length a=5 m; (lower panels) Four sided cusped duct with side length a=10 m.

MATLAB Exercise 6.1 Subsidence and consolidation of soils and rocks

Subsidence of topographic surfaces may occur in differing ways. When it happens in a relative slow way, then it is known as settlement. Subsidence is distinctly observed in former mined areas, where significant volumes of underground material has been extracted. Removal of porous fluid phase and reduction of pore pressure are also responsible for consolidation of soils and rocks and the consequent collapse of their intergranular space. The processes like sediment deposition or building of thick ice covers due to glaciation as well as man-made constructions are typical causes of soils and rocks consolidation. Subsidence and consolidation processes are very often connected with the variations of ground water table. Subsidence may be subdivided into three stages: (1) immediate consolidation due to elastic deformation of matrix material; (2) primary consolidation due to fluid phase (water) outflow from pore space; and (3) secondary consolidation due to creep and plastic modifications of material texture. In this exercise the second stage will be considered and numerically modelled. The vertical stress acting on solid phase is $\Delta\sigma_z$, the pore pressure is u, and the

effective stress is $\Delta\sigma_z' = \Delta\sigma_z - u$ (in sense of the Terzaghi definition). The relationship between the change of void fraction Δe and the vertical strain is: $\varepsilon_z = \frac{\Delta e}{1+e_0}$ (Fig. ME 6.1.1). By applying the vertical stress $\Delta\sigma_z$ the pore pressure becomes instantaneously equal the stress $\Delta\sigma_z$ and the initial effective stress $\Delta\sigma_{z,0}' = 0$ is zero. When the drainage path for fluid escape is open pore pressure *u* will decrease and the total external load is transferred now to the solid matrix.

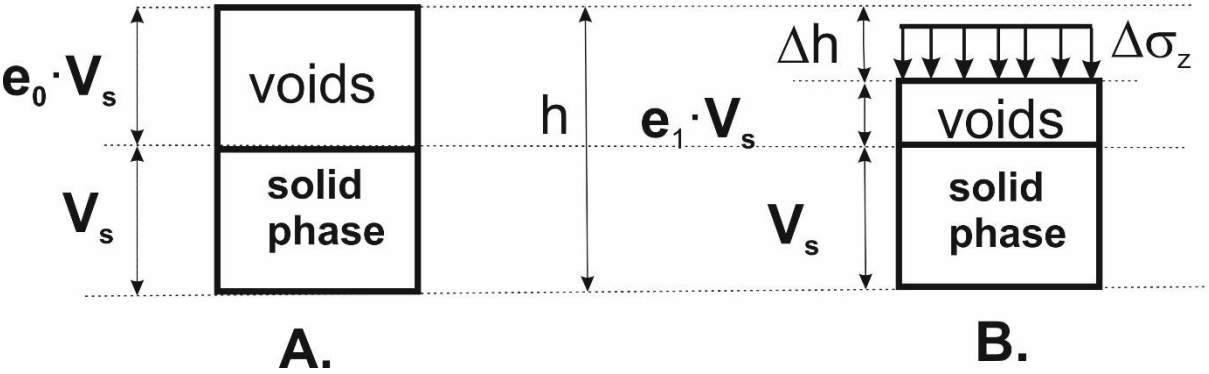


Fig. ME 6.1.1 Principle of consolidation: an element of rock or soil consists (solid phase) and porous space (voids). The initial void fraction is defined trough the ratio of volumes $e_0 = V_{(voids)}/V_{(solid\ phase)}$. (A). After applying the vertical stress $\Delta\sigma_z$ the volume of porous space is e_1 . (B) The vertical deformation of volume is given by $\varepsilon_z = \frac{\Delta h}{h} = \frac{\Delta h \cdot A}{h \cdot A} = \frac{(e_0 - e_1)\cdot V_S}{(1+e_0)\cdot V_S} = \frac{\Delta e}{1+e_0}$, where Δe is the change of void volume fraction. So during the consolidation process $\Delta\sigma_z$ remains constant but the pore fluid pressure *u* decreases due to drainage, and the load transferred continuously from water to the matrix resulting in the effective stress $\Delta\sigma_z'$ increase.

Consider the small element of consolidating soil having base area *A* at depth *z*, the hydraulic conductivity of the solid matrix is *k*. The excess of pore pressure over the hydrostatic pressure $\rho \cdot g \cdot z$ is denoted by $\bar{u}$. During time *dt* the amount of fluid which flows in the small volume having the height δz is $q = A \cdot k \cdot \frac{d\bar{u}}{\rho \cdot g \cdot dz}$ and the fluid amount flowing out of this volume is:

$q + \delta q = A \cdot k \cdot \frac{d(\bar{u}+\delta\bar{u})}{\rho \cdot g \cdot dz} = q + \overbrace{A \cdot k \cdot \frac{d^2\bar{u}}{\rho \cdot g \cdot dz^2} \cdot \delta z}^{\delta q}$. The increase of fluid flux is due to the volume contraction and extraction of some fluid amount having volume decrement: $-A \cdot \delta h$ per unit of time (Fig. ME 6.1.2), i.e. $\delta q = -A \cdot \frac{\left(\frac{\delta h}{\delta z}\right)}{dt} \cdot \delta z = -A \cdot \frac{d\varepsilon_z}{dt} \cdot \delta z$.

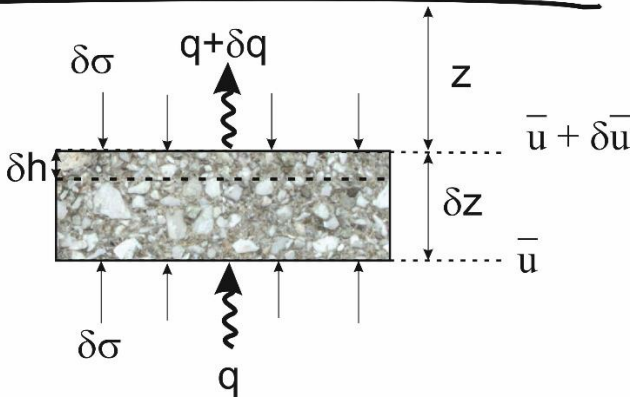


Fig. ME 6.1.2 Principle of consolidation.

Finally, one may write the *consolidation equation* as follows:

$$\frac{\delta\varepsilon_z}{\delta t} = -\frac{k \cdot d^2\bar{u}}{\rho \cdot g \cdot dz^2} \qquad \text{(ME 6.1.1).}$$

The relationship between vertical strain ε_z and effective vertical stress $\Delta\sigma_z$' is assumed to be linear, and this proportionality defines the coefficient of compressibility or the so called *coefficient of volume change* m_v through the relationship: $m_v = \frac{\varepsilon_z}{\Delta\sigma_z - \bar{u}}$. In the present case the external load is considered to be time independent, and the final form of the consolidation equation is as follows:

$$\frac{\partial \bar{u}}{\partial t} = \frac{k \cdot \partial^2 \bar{u}}{\rho \cdot g \cdot \partial z^2} = c_v \cdot \frac{\partial^2 \bar{u}}{\partial z^2} \qquad \text{(ME 6.1.2)},$$

where C_v is called the *coefficient of consolidation*. The analytical solutions of the diffusion type of partial differential equation (ME 6.1.1) may be taken from the textbooks on thermal conductivity (i.e. Carslaw& Jaeger, 1959). Degree of consolidation at depth *z* is defined as the ratio $\bar{U} = \frac{\bar{u}_0 - \bar{u}}{\bar{u}_0}$, where $\bar{u}_0$ is the initial excess of pore pressure. If one introduces the dimensionless time $T_v = \frac{c_v \cdot t}{h^2}$, and the dimensionless drainage path ratio *Z=z/h*, where *h* is the total depth of drainage, then (ME 6.1.2) may rewritten in the dimensionless form: $\frac{\partial \bar{U}}{\partial T_v} = \frac{\partial^2 \bar{U}}{\partial Z^2}$. For the initial conditions at *t*=0 the pressure excess is constant through the depth $\Delta\sigma_z = \bar{u}_0$, and at the top boundary *z=0* there is a drainage condition $\bar{u}$=0, and at the bottom *z=h* there is no influx condition $\frac{\partial \bar{u}}{\partial z}$=0. Then, the analytical solution is given by the series (Carslaw& Jaeger, 1959):

$$\bar{U} = 1 - \sum_{m=0}^{\infty} \frac{2}{M} \cdot sin(M \cdot Z) \cdot e^{-M^2 \cdot T_v} \qquad \text{(ME 6.1.3)},$$

where $M = \frac{\pi}{2} \cdot (2m + 1)$. Here below is the MATLAB code to calculate $\bar{U}$ as a function of depth *z* and time *t*:

```
clear all;
cv = 2e-3; % consolidation coefficient
H = 4.;% drainage depth in m
zstep=51; Z = linspace(0,1,zstep); % normalized depth
% discretize time domain
duration = 10000; dt =250;nstep = round(duration/dt);
time = [0:dt:duration];
T = cv/H/H.*time % normalized time
zv = H*linspace(1,0, zstep); %depth in m
U = zeros(zstep,nstep); % normalized pore pressure
for i=1:zstep
    for j=1:nstep
        for m = 0:5000
U(i,j) = U(i,j) + …
4/pi()/(2*m+1).*exp(-(pi()/2)^2*((2*m+1)^2).*T(j))*
…sin(pi()/2*(2*m+1).*Z(i));
        end
    end
end
% plot consolidation vs. depth at the end
pore_pressure = 1-U;
figure (1)
for i = 1:nstep
    plot(pore_pressure(:,i),zv,'r-'); hold on;
    grid on; xlabel('consolidation U');
    ylabel('z position, m');
end
```

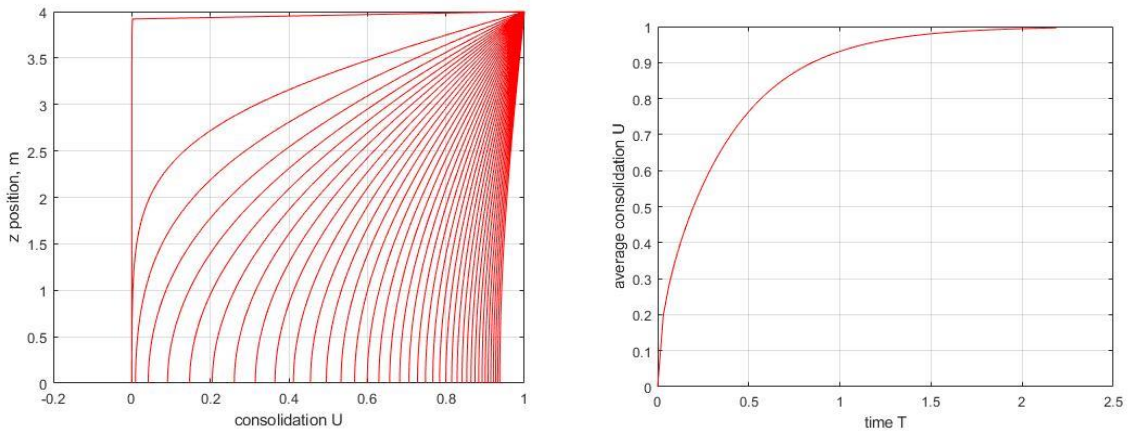


Fig. ME 6.1.3 (left panel) Consolidation U versus depth position z. The initially zero consolidation changes with time from the stepwise shape to zero. (right panel) Average consolidation $\overline{U}_{mean}$ as a function of dimensionless time T_v.

The mean value of consolidation calculated by integration of (ME 6.1.4) over entire depth from 0 to h ($0 \leq Z \leq 1$) is:

$$\overline{U}_{mean} = 1 - \sum_{m=0}^{\infty} \frac{2}{M} \cdot e^{-M^2 \cdot T_v} \cdot \overbrace{\int_0^1 \sin(M \cdot Z) \cdot dZ}^{-\frac{1}{M} cos\frac{\pi}{2}} = 1 - \sum_{m=0}^{\infty} \frac{2}{M^2} \cdot e^{-M^2 \cdot T_v} \quad \text{(ME 6.1.4).}$$

The time $T_{v,c}$ by which the bottom boundary excess pore pressure $\overline{U}(Z=1)$ is affected by the drainage at the upper boundary $\overline{U}(Z=0)=0$ may be estimated from (ME 6.1.3) by inserting Z=1 and leaving only the first term in the series: $\sum_{m=0}^{\infty} \frac{2}{M} \cdot (-1)^m \cdot e^{-M^2 \cdot T_v}$=1→ $\left\{ \frac{4}{\pi} \cdot e^{-\frac{\pi^2}{4} \cdot T_{v,c}} + \cdots \right\} = 1$, which provides the relationship: $T_{v,c} \approx \frac{4}{\pi^2} \ln(\frac{4}{\pi}) \approx 0.098$. The exact solution at $\overline{U}(Z=1)$ and the first term in the series of this solution are shown in Fig. ME 6.1.4.

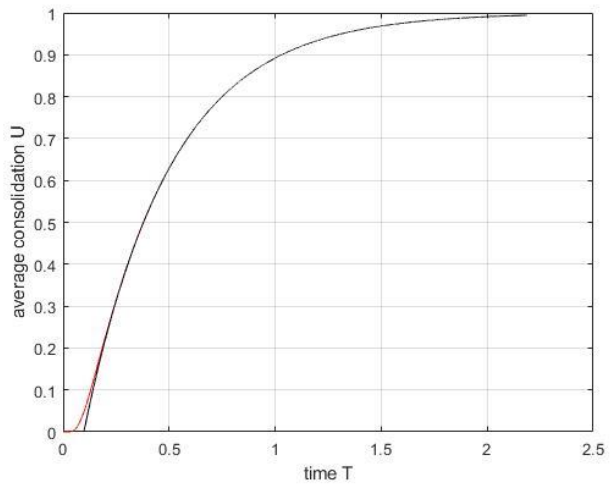


Fig. ME 6.1.4 (left panel) Consolidation $\overline{U}(Z=1)$ versus dimensionless time T_v: red curve is the full series solution, black line is the first term of the series solution.

The analysis based on $\overline{U}_{mean}$ (Fig. ME 6.1.3 right panel) results in $T_{v,c} \approx \frac{1}{12} = 0.083$, and at this time the average consolidation is $\overline{U}_{mean}$ =1/3. At $T_v < T_{v,c}$ the evolution of average

consolidation may be approximated as $\overline{U}_{mean} = \frac{2}{\sqrt{3}} \cdot \sqrt{T_v}$, and at $T_v > T_{v,c}$ the time dependence of average consolidation is: $\overline{U}_{mean} = 1 - \frac{2}{3} \cdot \exp\left(\frac{1}{4} - 3 \cdot T_v\right)$.

MATLAB Exercise 6.2. Pore compressibility and shear compliances

The exercise deals with calculations of compressibility and shear compliances used in The Mori-Tanaka differential scheme: $\check{P}, \check{Q}$ as a function of the Poisson ratio ν of matrix with spheroidal pores having aspect ratio α: $\check{P}(\alpha)$& $\check{Q}(\alpha)$ are shown in normalized form Fig. ME6.2.1.

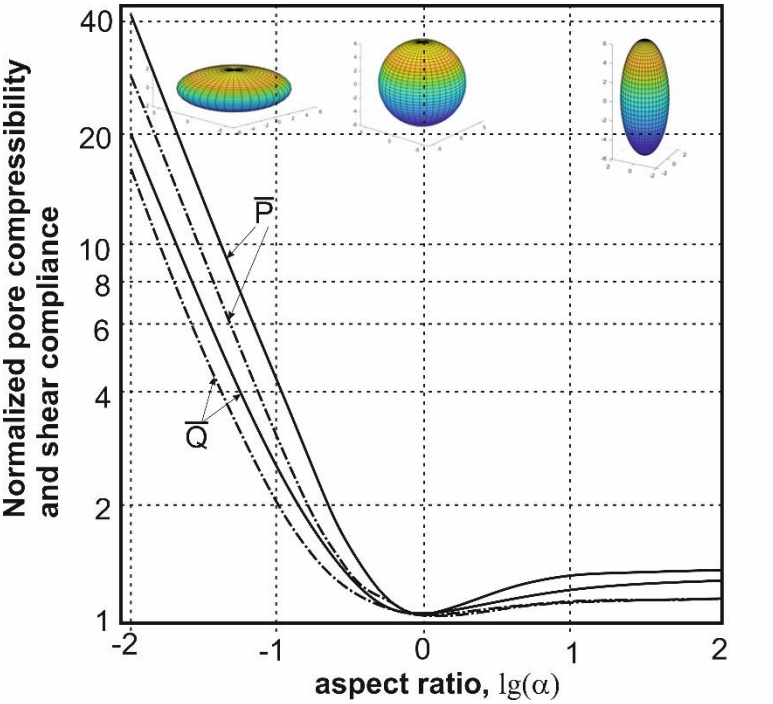
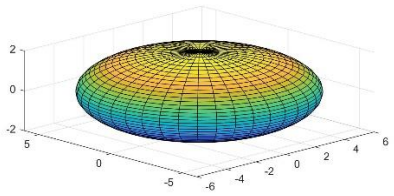
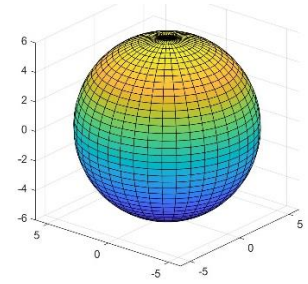
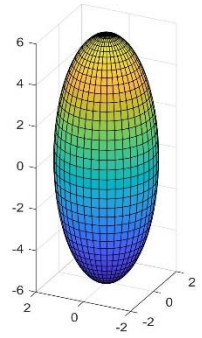


Fig. ME6.2.1 Normalized pore compressibility $\bar{P} = \frac{\check{P}}{\check{P}_S}$ and shear compliance $\bar{Q} = \frac{\check{Q}}{\check{Q}_S}$ of spheroidal pores, as a function of α shape aspect ratio, for two values ν of the matrix Poisson's ratio. Thick line ν=0.5, dashed-dotted line ν=0. Expressions for the pore compressibility compliance $\check{P}$, and the shear compliance, $\check{Q}$, for 2 limiting cases (α=0 and α=∞) are given in Table ME6.2.1. In Fig. $\check{P}$ & $\check{Q}$ have been normalized respectively to $\check{P}$s and $\check{Q}$s, compressibility and shear compliances of spherical pores, α=1 (replotted from David & Zimmerman, 2011).

The normalization factors $\check{P}_s$, $\check{Q}_s$ and the limit expressions of $\check{P}, \check{Q}$ are shown in Table ME6.2.1.

Table ME6.2.1 Limit values of compressibility $\check{P}$ and shear $\check{Q}$ compliances of pores having aspect ratio α (from David & Zimmerman, 2011)			
Pore geometry	$\check{P}$	$\check{Q}$	**Comments,α**

	$\frac{4}{3\pi\alpha}\cdot\frac{1-\nu^2}{1-2\nu}$	$\frac{4}{3\pi\alpha}\cdot\frac{\left(1-\frac{\nu}{5}\right)}{\left(1-\frac{\nu}{2}\right)}$	Oblate spheroid or penny shape pore with a*spect ratio*: $\alpha \to 0$
	$\breve{P}_s=\frac{3}{2}\cdot\frac{1-\nu}{1-2\nu}$	$\breve{Q}_s=\frac{15}{7}\cdot\frac{1-\nu}{7-5\nu}$	Sphere A*spect ratio*: $\alpha = 1$
	$\frac{5-4\nu}{3\cdot(1-2\nu)}$	$\frac{8}{15}\cdot(5-3\nu)$	Prolate spheroid or needle-shape pore with a*spect ratio*: $\alpha \to \infty$

To calculate $\breve{P}$ & $\breve{Q}$one uses the aspect ratio of two half axis α, the matrix Poisson's ratio *n*, and the aspect ratio function $g(\alpha)$:

$g(\alpha)=\frac{\alpha}{(1-\alpha^2)}\cdot\left[\frac{arccos\alpha}{\sqrt{1-\alpha^2}}-\alpha\right]$ for α<1,

and

$g(\alpha)=\frac{\alpha}{(\alpha^2-1)}\cdot\left[\alpha-\frac{arccosh\alpha}{\sqrt{\alpha^2-1}}\right]$ for α>1 (ME6.2.1).

The code below calculates $\breve{P},\breve{Q}$ as a function of the aspect ratio of pores according to Eqs. (29-30) from David & Zimmerman (2011).

```
clc; nui=[0,0.25,0.35];%input initial Poisson's ratio at zero prorosity
alpha=logspace(-1,1,100);%input aspect ratio

figure (1)
for i=1:3
           plot(log10(alpha),Q(alpha,nui(i),G(alpha)),'LineWidth',2);%put
data poin on a graph
        hold on
    Ymx =  max(Q(alpha,nui(i),G(alpha)));
titlemn = strcat('\nu = ',num2str(nui(i)));
        text(min(log10(alpha)),Ymx, titlemn,'FontSize', 14, 'Color', 'b',
'FontWeight', 'bold')
```

```
end
xlabel('log(\alpha)') %put label on horizontaö axis
ylabel('Q(\alpha)')%put label on vertical axis
set(gca,'fontsize', 18);%define the font size
   hold on
grid on

figure(2)
for i=1:3
           plot(log10(alpha),P(alpha,nui(i),G(alpha)),'LineWidth',2);%put
data poin on a graph
        hold on
        Ymx =  max(P(alpha,nui(i),G(alpha)));
titlemn = strcat('\nu = ',num2str(nui(i)));
        text(min(log10(alpha)),Ymx, titlemn,'FontSize', 14, 'Color', 'b',
'FontWeight', 'bold')
end
        xlabel('log(\alpha)') %put label on horizontaö axis
ylabel('P(\alpha)')%put label on vertical axis
set(gca,'fontsize', 18);%define the font size
hold on
grid on

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function p=P(a,nu,g)% function to calculate bulk compleance
if g==NaN
  p=3.*(1-nu)./2./(1-2.*nu);
else
nr=4.*(1+nu)+2*a.^2.*(7-2.*nu)-(3.*(1+4.*nu)+12.*a.^2.*(2-nu)).*g;
br=2.*a.^2+(1-4.*a.^2).*g+(a.^2-1).*(1+nu).*g.^2;
p=(1-nu)./6./(1-2.*nu).*nr./br;
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function q=Q(a,nu,g)% function to calculate shear compleance
if g==NaN
  q=15./7.*(1-nu)./(7-5.*nu);
else
nr1=4.*(a.^2-1).*(1-nu)./15;
br1=8*(nu-1)+2.*a.^2.*(3-4.*nu)+((7-8.*nu)-4.*a.^2.*(1-2.*nu)).*g;

nr2=8.*(1-nu)+2.*a.^2.*(3+4.*nu)+((8.*nu-1)-
4.*a.^2.*(5+2.*nu)).*g+6.*(a.^2-1).*(1+nu).*g.^2;
br2=2.*a.^2+(1-4.*a.^2).*g+(a.^2-1).*(1+nu).*g.^2;

nr3=8.*(nu-1)+2.*a.^2.*(5-4.*nu)+(3.*(1-2.*nu)+6.*a.^2.*(nu-1)).*g;
br3=-2.*a.^2+((2-nu)+a.^2.*(1+nu)).*g;

q=nr1./br1.*(nr2./br2-3.*nr3./br3);
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function g=G(a)% aspect ratio function
if a==1
    g=NaN;
elseif a<1
    g=a./(1-a.^2).^(1.5).*(acos(a)-a.*sqrt(1-a.^2));
else a>1
    g=a./(a.^2-1).^(1.5).*(a.*sqrt(a.^2-1)-acosh(a));
    end
```

```
end
```

MATLAB Exercise 7.1 Density - V_p – $\bar{m}$ multivariant regression

The background of (7.18) formalism follows from the empirical relationship

$\frac{\partial(\rho \cdot V_P)}{\partial V_P} \approx const$ (ME 7.1.1).

Using (7.6) this derivative may be expressed as follows:

$\rho \cdot V_P = \sqrt{\rho \cdot \left(K + \frac{4}{3}\mu\right)}$ and $\frac{\partial(\rho \cdot V_P)}{\partial V_P} = \frac{1}{2} \cdot V_P \cdot \frac{\partial(\rho)}{\partial V_P} \approx const$ (ME 7.1.2).

This relationship may be reformulated as the differential equation: $\frac{dV_P}{V_P} \sim d\rho$, the solution of which is given by the relationship $\ln(V_P) \sim \rho$.

Here, the dataset of $ln(V_p)$, $\bar{m}$, and density ρ of rocks are adapted from Birch (1961). It consists of three vectors: $ln(V_p)$ values (*x1*), mean atomic weight (*x2*) and rock density (*y*). The multivariant regression is given by $y=b(1)+b(2)\cdot ln(V_p)+b(3)\cdot m+b(4)\cdot ln(V_p)\cdot \bar{m}$. The fitting coefficients are 3 elements of vector *b*. MATLAB procedure to find vector *b* elements is: `b = regress(y,X)`. The results of the multivariant linear regression are plotted in 3D space as the plane (Fig. ME 7.1). The last cross-correlation term results in a negligible contribution, because $b(4) \approx 10^{-3}$. The color indicated on plane corresponds to the relative contribution of cross-correlation term of regression in %.

```
%Database of rocks from Birch, F., 1961.
%Velocity of compressional waves in rocks to 10 kilobars, Part 2, J.
geophys. Res., 66, 2199-2224
% loading dataset
x1 = [2.36 2.19 2.14 2.17 1.89 2.05 1.97 1.95 2.08 1.82 2.12 1.97
1.98 1.91 1.91 1.91 1.92 1.84 2.06 1.92 1.85]';
% ln(Vp, km/s)
x2 = [20.4 22.8 22.6 20.4 30.9 31.9 33.1 30.9 24.1 20.9 20.4 21.3
24.3 21.8 21.8 21.8 22 20.8 21.7 21.1 20.7]';
% mean atomic weight of a rock
y = [3.8 3.57 3.85 3.32 4.55 4.97 4.65 4.5 3.95 2.61 3.2 2.76 3.75
2.9 3.09 2.87 2.9 2.75 3.4 2.7 2.72]';
% density in g/cm³
X = [ones(size(x1)) x1 x2 x1.*x2];
%generate the matrix of 3 columns ln(Vp), m, and a cross
%correlation term Ln(Vp)*m
b = regress(y,X)     % applied multivariant regression to the
%dataset %vector b consists of 4 coefficients
scatter3(x1,x2,y,'filled')
hold on
x1fit = min(x1):0.1:max(x1);%generate a grid-vector of ln(Vp)
x2fit = min(x2):2:max(x2);% generate a grid-vector of m
[X1FIT,X2FIT] = meshgrid(x1fit,x2fit);
% generate a meshgrid ln(Vp) x m
YFIT = b(1) + b(2).*X1FIT + b(3).*X2FIT + b(4).*X1FIT.*X2FIT; %
calculate density using vector b(i) of the regression %coefficients
C=abs(b(4).*X1FIT.*X2FIT./YFIT*100);% create a matrix of the
%relative cross correlation contributions ~ln(Vp)*m̄/ρ
surf(X1FIT,X2FIT,YFIT,C)
colorbar
hold on
xlabel('ln(V_p, km/s)')
ylabel('mean atomic weight m')
```

```
zlabel('density g/cm³')
set(gca,'FontSize',15)
view(50,10)
hold off
```

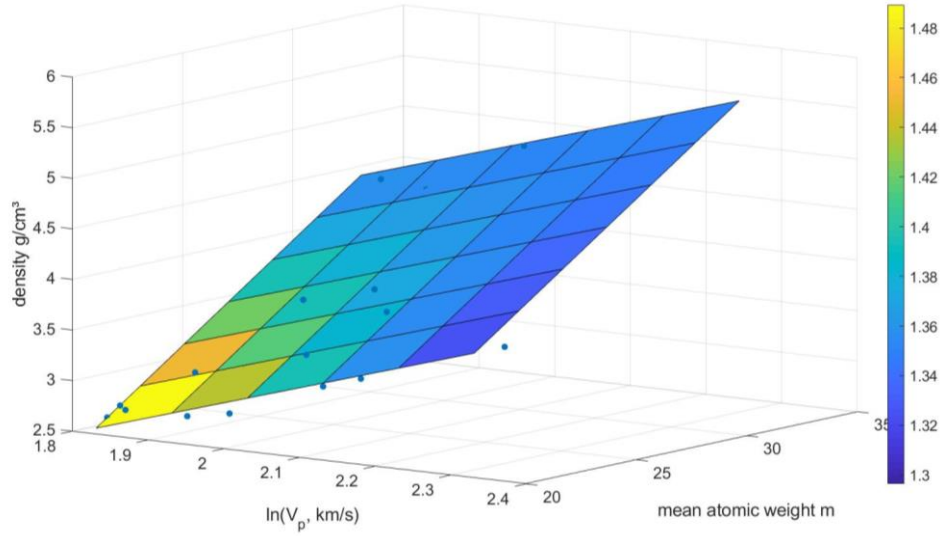


Fig. ME 7.1 Representation of the multivariant regression as the plane $\rho = b(1) + b(2) * \ln(V_P) + b(3) * \bar{m} + b(4) * \ln(V_P) * \bar{m}$ in 3D. Blue phe points are the dataset of density vs. velocity and atomic weight from (Birch, 1961b). The color bar indicate the contribution of cross-correlation term of regression, ~ $b(4)*ln(V_p)*\bar{m}/\rho$ in %.

MATLAB Exercise 7.2 Pole diagrams

To represent vector characteristics of an object, for example acoustic velocities in anisotropic crystals or positions of fault planes, one frequently uses pole diagrams. The idea of such a representation is to indicate a 3D vector by a point in 2D plane by using the interception point of vector with the equatorial plane of unity sphere (see Fig. ME 7.2.1 A). The vector is plotted starting from South pole of sphere with azimuth angle α and altitude angle β/2. (Alternatively, it may be plotted starting from point (0 0 0) using altitude angle β indicated in Fig. ME 7.2.1A, then, the interception point of vector with unity sphere will be projected on the equatorial plane). Interception point in the equatorial plane is characterised by polar angle α and radius vector $= \left|sin\frac{\beta}{2}\right|$. In the case of a crystal the orientations of crystallographic planes with the help of pole diagrams are illustrated in Fig. ME7.2.1 C&D.

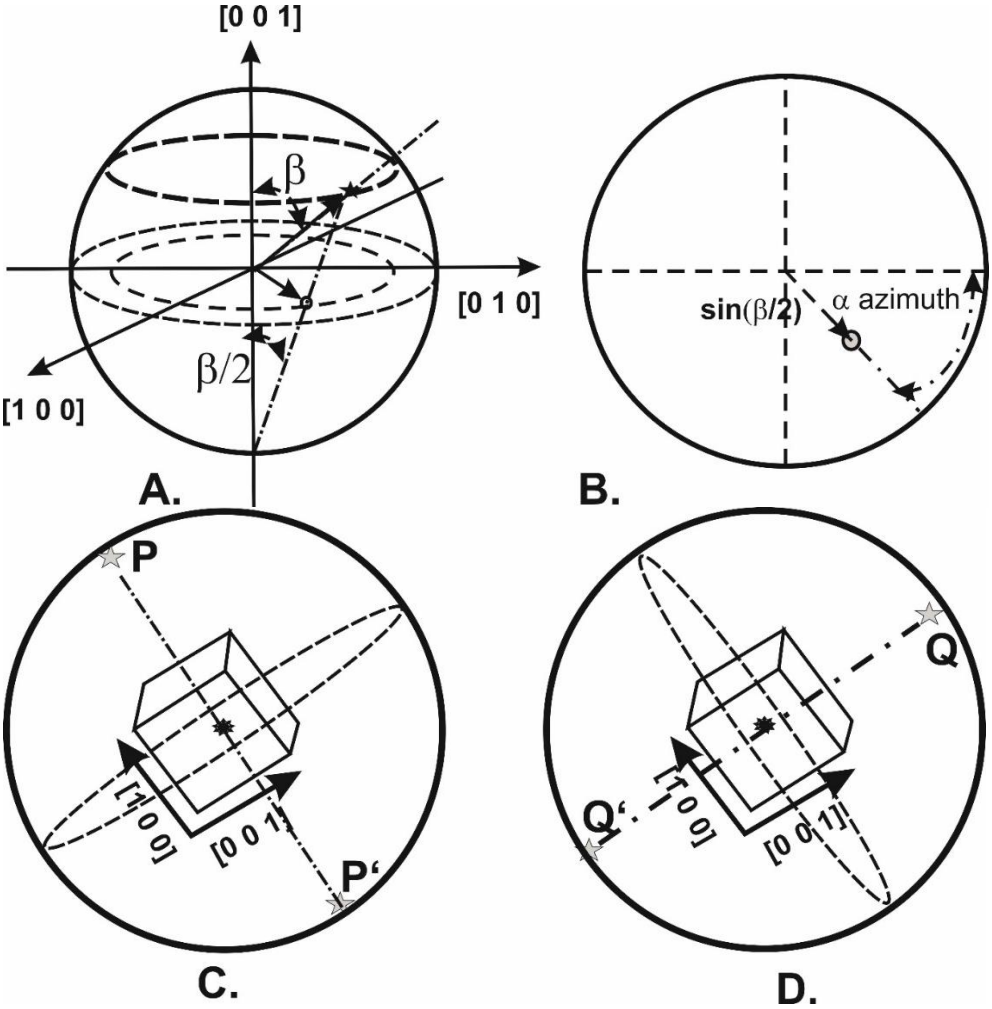


Fig. ME 7.2.1 Construction of pole diagrams. A. Vectors having an angle β with the axe [0 01] are represented in plane (0 0 1) by their interception point (small open circle) of the line connecting the point $\{0\ 0\ \bar{1}\}$ with the end of the vector on the unity sphere (black star). B. The interception point on the (0 0 1) plane in polar coordinates is characterized by azimuth angle (α) and radial distance from coordinate centre =$|sin(\beta/2|$. C & D: Projections (0 1 0) of cubic crystal. C. P [1 0 0] and P' $[\bar{1}\ 0\ 0]$ directions are the poles of anisotropy along [1 0 0] axis. D: Q [0 0 1] and Q' $[0\ 0\ \bar{1}]$ are the poles of anisotropy along [0 0 1] axis.

The MATLAB program below plots and illustrates the construction of vector representation in equatorial plane on unit sphere.

```
% representation of a pole figure using a unit sphere
lat = 0:(pi/19):pi; long = 0:(2*pi/37):2*pi;%calculate the grid points
% of latitude and longitude
[LAT,LONG] = meshgrid(lat, long);% assamble the grid point in a meshgrid
X = sin(LONG).*sin(LAT); Y = sin(LAT).*cos(LONG); Z = cos(LAT);
figure, mesh(X,Y,Z), axis equal %plot longitude and latitude lines
set(gca,'fontweight','bold','FontSize',14, 'FontName', 'Times New Roman')
alpha 0.3 % set the transparency of the sphere surface
hold on
plot(sin(long),cos(long), 'r-','LineWidth', 2)% plot equatorial plane
h=fill(sin(long),cos(long),'r')% fill equatorial plane
set(h,'facealpha',.5) % set the transparency of the equatorial plane
hold on
plot3(0,0,0,'ko') %plot ponit [0 0 0]
hold on
latr=30/180*pi; longr=120/180*pi; %calculate coordinates of the vector with
% a given lattitude 30° % and longuitude 120°
```

```
xr = sin(longr)*sin(latr);
yr = sin(latr)*cos(longr);
zr = cos(latr);%calculate coordinates on the unit sphere
plot3(xr,yr,zr,'r*') %plot point [x y z]
p1 = [0 0 0]; % centre of the sphere
p2 = [xr yr zr]; % point on a unit sphere with the given coordinates
dp = p2-p1; % vector of a distance between two points
p3=[xr yr 0]; % point projection in the equatorial plane
dpp=p3-p1;% vector of a distance between two points
quiver3(p1(1),p1(2),p1(3),dp(1),dp(2), dp(3),1.5,'LineWidth', 2)
% plot a vector connecting two points
quiver3(p1(1),p1(2),p1(3),dpp(1),dpp(2),dpp(3), 2.5,'LineWidth', 2)
% plot a vector connecting two points
plot3(sin(longr)*sin(latr/2), cos(longr)*sin(latr/2), 0, 'ko')
%plot a point in the equatorial plane
text(p1(1),p1(2),p1(3), sprintf('(%.0f,%.0f,%.0f)',p1))
%put the text on the graph
h=line([0 xr],[0 yr],[-1 zr],'LineWidth', 2)%plot a line between two points
s = h.LineStyle; h.LineStyle = ':';%set the line style
hold on
```

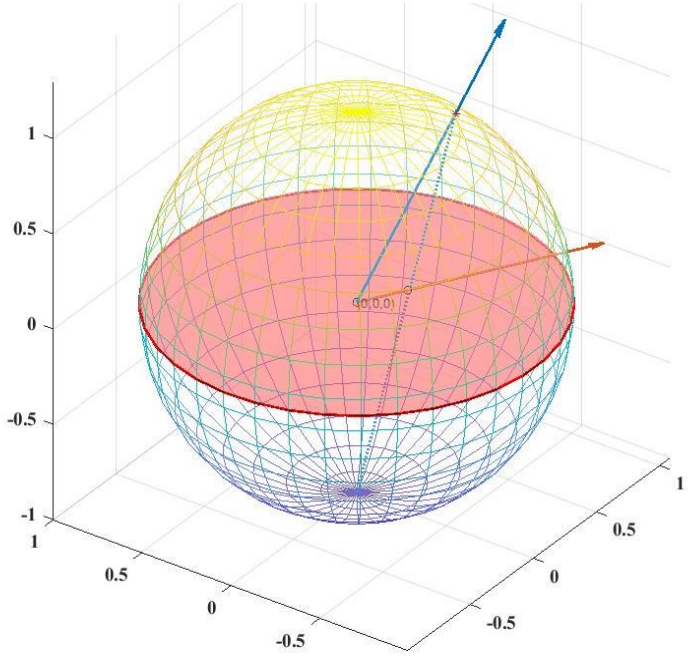


Fig. ME 7.2.2 MATLAB plot of unity sphere illustrating the construction of pole diagram.

The Cartesian coordinate system ise related to the spherical coordinates (for unity sphere radius $r = 1$) as:

$X = \cos(\lambda) \cdot \sin(\varphi)\,,\ \ Y = \sin(\lambda) \cdot \sin(\varphi)\,,\ \ Z = \cos(\varphi)$ (ME 7.2.1),

where λ is the longitude angle and φ is the latitude angle. The projection of lines having equal latitudes and longitudes on the equatorial slice of sphere is called the *equal-angle stereonet*. In structural geology the use of this type of stereonet is common for representation of planar and linear structural elements (Pollard & Fletcher, 2005).

.

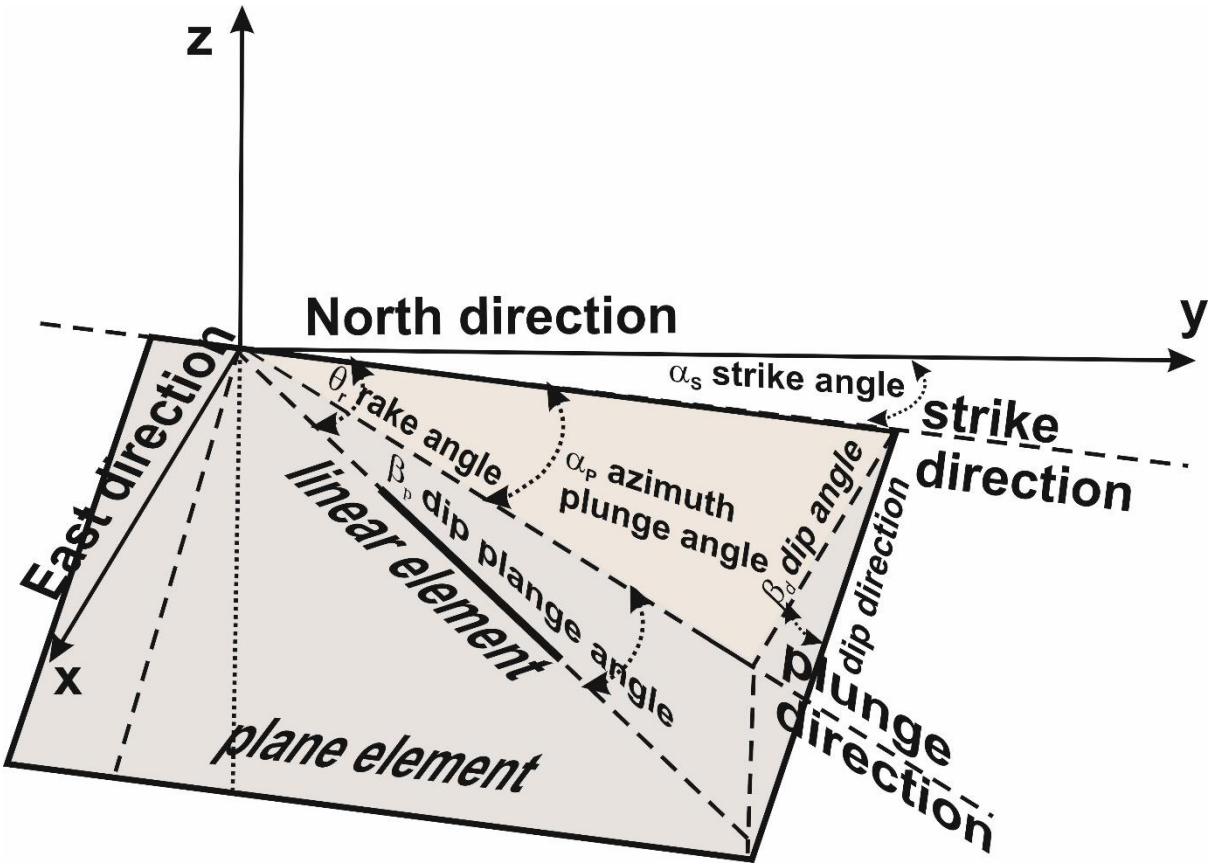


Fig. ME 7.2.3 Characteristics of planar and linear elements in 3D. The planar element is defined by the *strike direction* α_s (the interception planar element line with the surface) and the *dip angle* β_d (the angle built in the vertical cross section normal to the strike direction between the planar element surface and the horizontal surface). The linear element can be defined additionally through the *rake angle* θ_r of the planar element, the angle between the strike direction and the linear element direction. Another characterization of linear elements may be given by using the azimuth of *plunge direction* α_p (projection azimuth of the linear element on the surface) and the *dip plunge angle* β_p (the angle built in the vertical plane between the plunge direction and the linear element).

MATLAB program below plots planar element orientations using the given strike (`als=120°`) and dip (`phid` =60°) angles, and calculate coordinates of the normal vector to this plane on unity sphere. For the given rake angle (`thr` =60°) the plunge direction and the plunge angle will be calculatedas follows:

$\alpha_p = \alpha_s + \arctan(\tan\theta_r \cdot cos\beta_d)$, $\beta_p = \arcsin(\frac{sin\theta_r}{sin\beta_d})$ (ME 7.2.2),

(Pollard & Fletcher, 2005).

```
% Plot stereonet, point, and great circle (Goodman and Shi, 1985, p. 75)
% modified from Pollard, David D./Fletcher, Raymond % C. 2005
axis equal, axis off, box off % equal scaling in x and y,
no axes or box
axis ([-1 1 -1 1]) % sets scaling for x- and y-axes
 plot([-1 1],[0 0],'k:',[0 0],[-1 1],'k:') % plot x- and y-axes
hold on
set(gca,'fontweight','bold','FontSize',14, 'FontName', 'Times New Roman')
r = 1; % radius of reference circle
TH = linspace(0,2*pi,3601); % polar angle, range 2 pi, 1/10 degree
increment
[X,Y] = pol2cart(TH,r); % Cartesian coordinates of reference circle
plot(X,Y,'k','LineWidth', 2) % plot reference circle
 hold on
```

```
for j = 1:8 % loop to plot great circles at 10 degree increments
phid = j*(10*pi/180); % dip angle, phid
h = -r*tan(phid); rp = r/cos(phid); % x-coord of center, h, and radius, rp
X = -h + rp*cos(TH); Y = rp*sin(TH); % coordinates of points on great
circle
X(find(X.^2+Y.^2>r)) = nan; % eliminate points outside stereonet
plot(X,Y,'k:',-X,Y,'k:','LineWidth', 1) % plot two sets of great circles
hold on
end
 for j = 1:8 % loop to plot small circles at 10 degree increments
gam = j*(10*pi/180); % cone angle, gam
k = r/cos(gam); rp = r*tan(gam); % y-coord of center, k, and radius, rp
X = rp*cos(TH); Y = k + rp*sin(TH); % coordinates of points on small circle
Y(find(X.^2+Y.^2>r)) = nan; % eliminate points outside stereonet
plot(X,Y,'k:',X,-Y,'k:','LineWidth', 1) % plot two sets of small circles
hold on
end
 als = 120*pi/180; ald = als+pi/2; % strike 120°
phid = 60*pi/180; % dip angle 60°
h = -r*tan(phid)*sin(ald); % x-coord of center
k = -r*tan(phid)*cos(ald); % y-coord of center
rp = r/cos(phid); % radius
X = h + rp*cos(TH); Y = k + rp*sin(TH); % coordinates of points
X(find(X.^2+Y.^2>r)) = nan; % eliminate points outside stereonet
X1 = -h + rp*cos(TH); Y1 = -k + rp*sin(TH); % coordinates of points
X1(find(X1.^2+Y1.^2>r)) = nan; % eliminate points outside stereonet
plot(X,Y,'b',X1,Y1,'b:','LineWidth', 2) % plot planar element as great
%circle
aln = ald + pi;  % plunge direction of normal to planar element
phin = (pi/2) - phid; % plunge angle of normal
x = r*tan(pi/4 - phin/2)*sin(aln);
y = r*tan(pi/4 - phin/2)*cos(aln);
plot(x,y,'ko') %plot pole to planar element as a point
thr = 60*pi/180; % rake angle =60°
alp = als + atan2(sin(thr)*cos(phid),cos(thr)); % plunge direction
phip = asin(sin(thr)*sin(phid)); % plunge angle
xr = r*tan(pi/4 - phip/2)*sin(alp);
yr = r*tan(pi/4 - phip/2)*cos(alp);
plot(xr,yr,'bo', 0,0,'k*') %plot linear element as point on great circle
hold on
p1 = [0 0]; % centre of the plane
p2 = [xr yr]; %Linear element direction
p3=[x,y]; % pole direction
dp = p2-p1; dpp=p3-p1;% Difference between points
quiver(p1(1),p1(2),dp(1),dp(2),0,'LineWidth', 2);
%plot an arrow between two points
quiver(p1(1),p1(2),dpp(1),dpp(2),0,'LineWidth', 2)
text(p1(1),p1(2), sprintf('(%.0f,%.0f)',p1));
% put the text of a point coordinates (0,0)
```

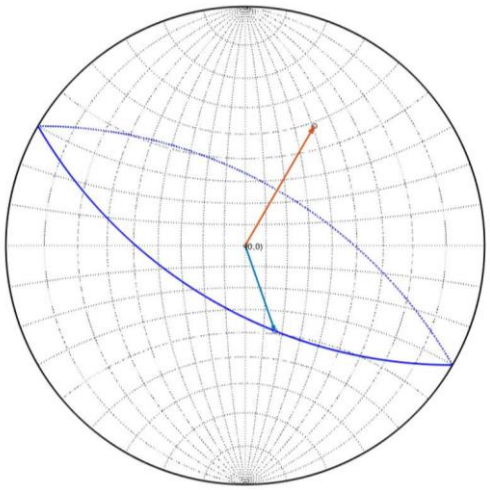


Fig. ME 7.2.4 MATLAB construction of the plane pole using the strike and dip angles of planar element. The intercept of normal vector (red arrow) of the planar element (blue circle) with the unity sphere defines the plane pole position on 2D projection of unity sphere.

In order to represent a set of pole locations corresponding to a family of planar elements one may use the mean position of normal vector (Fig. ME 7.2.5 upper panel). The mean position can be calculated as the vector sum of normal vectors for the group planar elements (red star in Fig. ME 7.2.5 upper panel), or by calculating the average density of pole points per square unit of projection.
In the MATLAB program below the set of 100 planes with the strike and dip angles is randomly generated by two Gaussian distributions: `st(ni)=normrnd(100,20)` and `di(ni)=normrnd(50,10),` where the mean strike angle is 100° an the dispersion 20° in one group of planes, and the dip angle is 50° and the dispersion 10° in another group. The deviation angle from the mean value of pole positions is estimated through the arc lengths:

$$\Delta X_i = \cos(\bar{\lambda}) \cdot \cos(\bar{\varphi}) - \cos(\lambda_i) \cdot \cos(\varphi_i),$$
$$\Delta Y_i = \cos(\bar{\lambda}) \cdot \sin(\bar{\varphi}) - \cos(\lambda_i) \cdot \sin(\varphi_i),$$
$$\Delta Z = \sin(\bar{\varphi}) - \sin(\varphi_i) \qquad \text{(ME 7.2.3)},$$

and

$$deviation\ angle_i = \sqrt{\Delta X_i^2 + \Delta Y_i^2 + \Delta Z_i{}^2} \qquad \text{(ME 7.2.4)}.$$

```
function stereo_plane
% modified from Pollard, David D./Fletcher, Raymond % C. 2005
% Plot stereonet, and poles to planar elements
axis equal, axis off, box off % equal scaling in x and y, axis ([-1 1 -1
1]) % sets scaling for x- and y-axes
 plot([-1 1],[0 0],'k:',[0 0],[-1 1],'k:','LineWidth',2) % plot x- % and y-
axes
r = 1; % radius of reference circle
TH = linspace(0,2*pi,3601); % polar angle, range 2 pi, 1/10 degree
%increment
[X,Y] = pol2cart(TH,r); % Cartesian coordinates of reference circle
plot(X,Y,'k') % plot reference circle
 for j = 1:8 % loop to plot great circles at 10 degree increments
phid = j*(10*pi/180); % dip angle, phid
h = -r*tan(phid); rp = r/cos(phid); % x-coord of center, h, and %radius, rp
X = -h + rp*cos(TH); Y = rp*sin(TH); % coordinates of points on great
circle
X(find(X.^2+Y.^2>r)) = nan; % eliminate points outside stereonet
```

```
plot(X,Y,'k:',-X,Y,'k:') % plot two sets of great circles
 end
for j = 1:8 % loop to plot small circles at 10 degree increments
gam = j*(10*pi/180); % cone angle, gam
k = r/cos(gam); rp = r*tan(gam); % y-coord of center, k, and %radius, rp
X = rp*cos(TH); Y = k + rp*sin(TH); % coordinates of points on %small
circle
Y(find(X.^2+Y.^2>r)) = nan; % eliminate points outside stereonet
plot(X,Y,'k:',X,-Y,'k:') % plot two sets of small circles
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%create two random vectors of strike and dip angles
ndp=100; %length of vectors
mu1=100;sigma1=20;mu2=50;sigma2=10;% define mean and dispersions
for ni=1:ndp
    st(ni)=normrnd(mu1,sigma1);
    di(ni)=normrnd(mu2,sigma2);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
UX=0;UY=0;UZ=0;
for j = 1:length(st)
aln = (st(j)+270)*pi/180;
phin = (90 - di(j))*pi/180;
alx(j) = atan2(sin(aln)*cos(phin),cos(aln)*cos(phin));
alx(j)=alx(j)+(alx(j)<0)*2*pi;
aly(j) =asin(-cos(phin+pi/2));
x(j) = r*tan(pi/4 - phin/2)*sin(aln);
y(j) = r*tan(pi/4 - phin/2)*cos(aln);
UX = UX + sin(aln)*cos(phin); UY = UY + cos(aln)*cos(phin);
UZ = UZ + cos(phin+pi/2); % components of resultant vector
end
U = sqrt(UX^2 + UY^2 + UZ^2) % magnitude of resultant vector
alU = atan2(UX/U,UY/U); alU = alU + (alU<0)*2*pi;
phiU = asin(-UZ/U); % azimuth and inclination of resultant vector
xU = r*tan(pi/4 - phiU/2)*sin(alU); %x ccordinate of resultant %vector on
2D
yU = r*tan(pi/4 - phiU/2)*cos(alU);%y ccordinate of resultant %vector on 2D
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
DX=cos(phiU)*cos(alU)-cos(alx).*cos(aly);%calculated X-length of arc in a
unit sphere
DY=cos(alU)*sin(phiU)-cos(alx).*sin(aly);% Y-length
DZ=sin(alU)-sin(alx);% Z length
alc=180/pi*(sqrt(DX.^2+DY.^2+DZ.^2));% deviation angle in 3D
scatter(x,y,[],alc,'filled') %scatter plot of poles to planar %elements
        g=colorbar;
w = g.LineWidth;
g.LineWidth = 1.5;
g.Label
g.Label.String = 'deviation angle°';
g.Label.FontSize = 14;
 plot(xU,yU,'r*','MarkerSize',12,'LineWidth',2)
 end
```

In Fig. ME 7.2.5 (right panel) pole positions of the same set of planar elements have been represent by using the density map. For estimations of point density the function `datadensity(x,y)` is used. The set of points has been divided into the *Voronoi cells* on the surface. The area of each Voronoi cell has been calculated. For each point on the pole diagram the inverse of specific area per point used to plot the *counter map*. For an open area containing no points the density is set to 0.

```
%modified from Pollard, David D./Fletcher, Raymond % C. 2005
```

```
clear all, clf, hold on % clear variables, current figure, hold plot
axis equal, axis off, box off % equal scaling in x and y, no axes or box
axis ([-1 1 -1 1]) % sets scaling for x- and y-axes
 plot([-1 1],[0 0],'k:',[0 0],[-1 1],'k:','LineWidth',2)
% plot x- and y-axes
r = 1; % radius of reference circle
TH = linspace(0,2*pi,3601); % polar angle, range 2 pi, 1/10 degree
increment
[X,Y] = pol2cart(TH,r); % Cartesian coordinates of reference circle
plot(X,Y,'k') % plot reference circle
 for j = 1:8 % loop to plot great circles at 10 degree increments
phid = j*(10*pi/180); % dip angle, phid
h = -r*tan(phid); rp = r/cos(phid); % x-coord of center, h, and radius, rp
X = -h + rp*cos(TH); Y = rp*sin(TH); % coordinates of points on great
circle
X(find(X.^2+Y.^2>r)) = nan; % eliminate points outside stereonet
plot(X,Y,'k:',-X,Y,'k:') % plot two sets of great circles
end
for j = 1:8 % loop to plot small circles at 10 degree increments
gam = j*(10*pi/180); % cone angle, gam
k = r/cos(gam); rp = r*tan(gam); % y-coord of center, k, and radius, rp
X = rp*cos(TH); Y = k + rp*sin(TH); % coordinates of points on small circle
Y(find(X.^2+Y.^2>r)) = nan; % eliminate points outside stereonet
plot(X,Y,'k:',X,-Y,'k:') % plot two sets of small circles
end
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%create two random vectors of strike and dip angles
ndp=100; %length of vectors
mu1=100;sigma1=20;mu2=50;sigma2=10;% define mean and dispersions
for ni=1:ndp
    st(ni)=normrnd(mu1,sigma1);
    di(ni)=normrnd(mu2,sigma2);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
aln = (st+270).*pi/180; % normal direction
phin = (90 - di).*pi/180; % plunge of normal
x = r*tan(pi/4 - phin./2).*sin(aln);
y = r*tan(pi/4 - phin./2).*cos(aln);
dd=datadensity(x,y);%density data calculations
N = 30; %length of grid
%------------- Gridding ------------------
xi = repmat(linspace(min(x),max(x),N),N,1);
yi = repmat(linspace(min(y),max(y),N)',1,N);
zi = griddata(x,y,dd,xi,yi);
%plotting data
ms=8; % define the size of symbol
             [c,h] = contour(xi,yi,zi);
            out.c = c;
                hs = gsp(x,y,dd,ms);
        out.hs = hs;
        g=colorbar;
w = g.LineWidth;
g.LineWidth = 1.5;
g.Label
g.Label.String = 'Density = poles per area unit';
g.Label.FontSize = 14;
        hold on
end
```

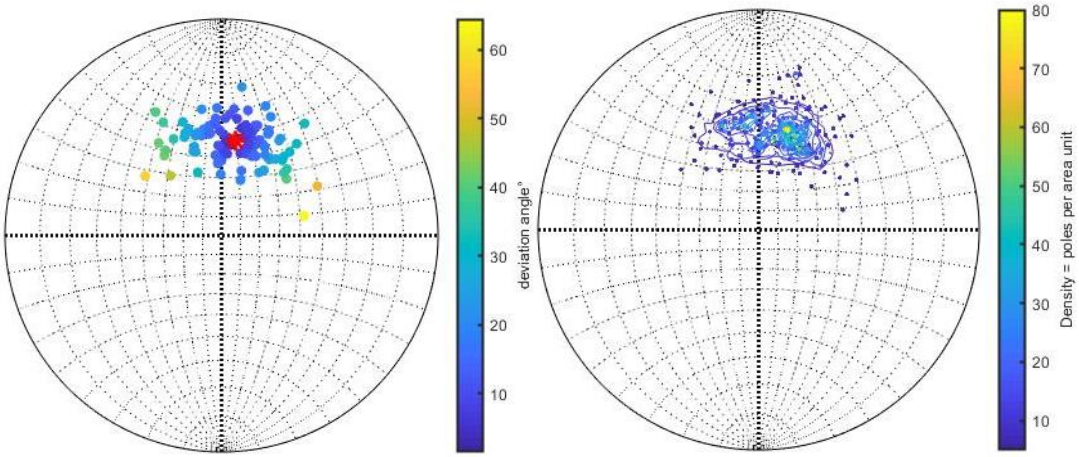


Fig. ME 7.2.5 MATLAB plot of the set of pole positions (n=100) corresponding to two Gaussian distributions havin (1) the mean value of strike angle 110° and dispersion 20°, and (2) the mean value of dip angle 50° and dispersion 10°. (left panel) Pole positions and the mean value (red star). The mean position corresponds to the normalized sum of normal vectors for the planar elements set. (right panel) Density plot of pole positions (number of poles per square unit of circle area) estimated by using the Voronoi cell formalism.

```
function dd = datadensity(x,y)
%Copy-Left, Alejandro Sanchez-Barba, 2005
%Computes the data density (points/area) of scattered points
% USAGE:    dd = datadensity(x,y)
% INPUT:    (x,y) -  coordinates of points
 x = x(:); y = y(:);
%Asuming x and y match
idat = isfinite(x); x = x(idat); y = y(idat);
holdstate = ishold;
if holdstate==0
    cla
end
hold on

Ld = length(x);
dd = zeros(Ld,1);
%----- Using Voronoi cells ------
        [v,c] = voronoin([x,y]);
        for k=1:length(c)
 %If at least one of the indices is 1, then it is an open region, its area
 %is infinity and the data density is 0
            if all(c{k}>1)
                a = polyarea(v(c{k},1),v(c{k},2));
                dd(k) = 1/a;%density of points estimated by an %inverse of
Voronoi cell area per point
            end %if
        end %for
        dd=dd.*pi/length(x);
end
function varargout = gsp(x,y,c,ms)%
%Graphs scattered points instead of MATLAB scatter-function
map = colormap;
ind = fix((c-min(c))/(max(c)-min(c))*(size(map,1)-1))+1;
h = [];%much more efficient than matlab's scatter plot
for k=1:size(map,1)
```

```
    if any(ind==k)
        h(end+1) = line('Xdata',x(ind==k),'Ydata',y(ind==k), ...
            'LineStyle','none','Color',map(k,:), ...
            'Marker','.','MarkerSize',ms);
    end
end
if nargout==1
    varargout{1} = h;
end
end
```

MATLAB Exercise 8.1 Generation and plotting of a square lattice of network resistors

MATLAB program below generates the square lattice with N^2 nodes connected with the resistors (conductance=1) having the concentration ***conc***. Each resistor corresponds to the third index of the matrix element: ***netMat***. The size of ***netMat***. matrix is *2*N*(N-1) x 3: 2*N*(N-1)* is the total number of all possible resistors connecting nearest nodes of lattice. The first matrix index is the starting node and the second index is the end node of resistor. The generation of resistors in lattice is provided by the random choice function: `R(randperm(numel(R), n)) = 1.` The results of lattice generation are shown in Fig. **ME 8.1** for the square lattice 12 x 12.

```
global conc
N=12;%number of nodes N x N
conc=0.66; % concentration of conductances =1, the rest is 0
resistivity=Res(N);% random generation of conductances between %the
nearest nodes
k=randi([1 N*(N-1)/2]); l= k+1; terminals = [k,l];% k<=N*(N-1)/2 and
l<N*(N-1)/2+1 are coordinates of two terminals nodes between nearest
nodes
[netMat] = lattice_gen(N); % resistor network lattice generation
lattice_plot(N,netMat,terminals)% plot of results
%**************************************************************
****************
function [netMat] = lattice_gen(N)
% https://de.mathworks.com/matlabcentral/fileexchange/42521-
%resistance-calculator
% Generate a square lattice of resistor network
% N is the number of nodes on each side, N must be even
if rem(N,2) == 1
    N = N+1;
end
R = 1.*Res(N);%conductance values
netMat = zeros(2*(N-1)*N,3); %number of conductances 2*N*(N-1)
cnt = 0;% count of conductances
% Horizontal conductances between nearest nodes
for k = 1:N:N^2
    for kk = k+1:k+N-1
        cnt = cnt+1;
        netMat(cnt,:) = [kk-1,kk,R(cnt)];
    end
end
% Vertical conductances
for k = 1:N
```

```
    for kk = k+N:N:k+(N-1)*N
        cnt = cnt+1;
        netMat(cnt,:) = [kk-N,kk,R(cnt)];
    end
end
end
%*************************************************************
function lattice_plot(N,netMat,terminals)
[x,y] = meshgrid(1:N,1:N);
x = rot90(x,3); y = rot90(y,3);
figure('name','Resistance Lattice')
axes('xlim',[1,N],'ylim',[1,N],'dataaspectratio',[1,1,1])
hold on
for k = 1:N^2
    text(x(k),y(k),num2str(k));
end
for k = 1:size(netMat,1)
    if netMat(k,3)==1
line([x(netMat(k,1)),x(netMat(k,2))],[y(netMat(k,1)),y(netMat(k,2))]
,'linestyle','-','color', 'red','linewidth',2); %plotting of
conductances=1
    else

line([x(netMat(k,1)),x(netMat(k,2))],[y(netMat(k,1)),y(netMat(k,2))]
,'linestyle','--','color', 'blue','linewidth',1);
% plotting of conductances=0
    end
end
if nargin > 2
    for k = 1:size(terminals,1)
        a = terminals(k,1);
        b = terminals(k,2);
line([x(a),x(b)],[y(a),y(b)],'marker','o','linestyle','none',
'markersize',10,'linewidth',2,'color',rand(1,3));
    end
end
axis off
end
%*************************************************************
*
function R=Res(N)
global conc
R=zeros((2*N*N-2*N),1); % generation of zero R-array of
%conductances
n=round(conc*(2*N*N-2*N));   % number of 1 in the R array
R(randperm(numel(R), n)) = 1;%  randomly chosen positions of n
conductances =1
R=R(:);% reshaping the array
end
```

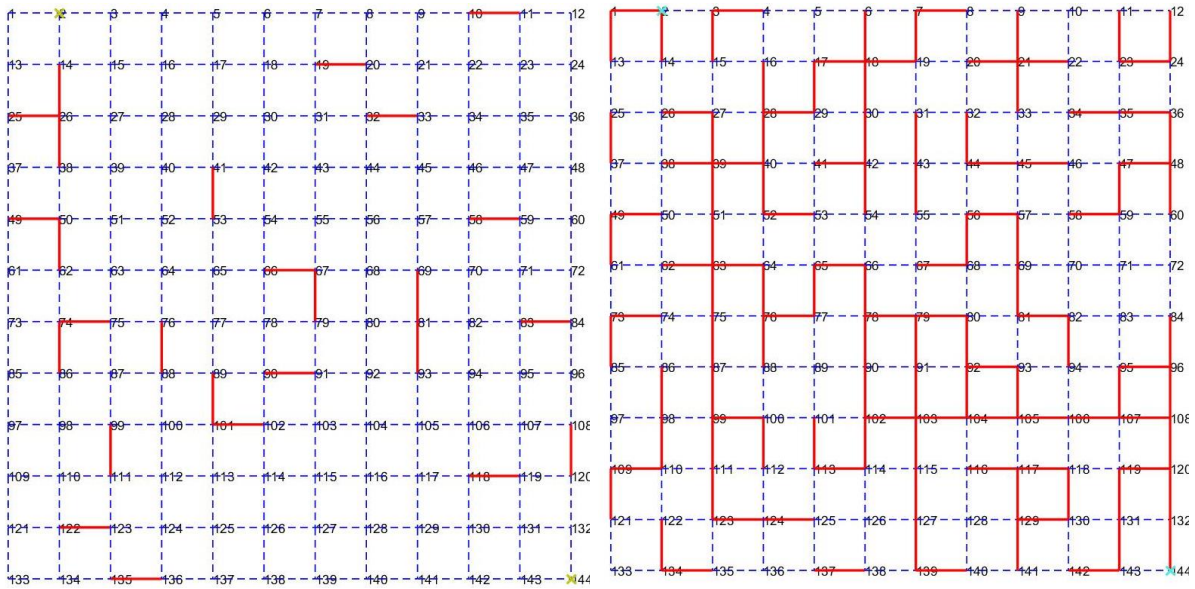


Fig. ME 8.1 Generation of square lattice 12 x 12 = 144 nodes and 264 sites of randomly distributed conductances: solid line– conductance = 1, dotted line – conductance = 0: (left panel) concentration =10% of conductances, (right panel) concentration = 66% of conductances.

MATLAB Exercise 8.2 Calculations of electric potential in a grid of electrical resistances

Consider finite a 2D square grid of resistors. The neighbor nodes of resistor grid are connected via electrical resistive elements. The case will be considered where the nodes are connected or disconnected with a certain probability. It may be related with a situation when there are two sorts of resistors with finite small values and infinite large values of resistivity. In general each interior node is connected with its four neighbors by means of conductance bonds, which may be finite and positive or zero.

Each boundary node is connected only to three nodes and four corner nodes are connected with two nodes (Fig. ME 8.2.1).

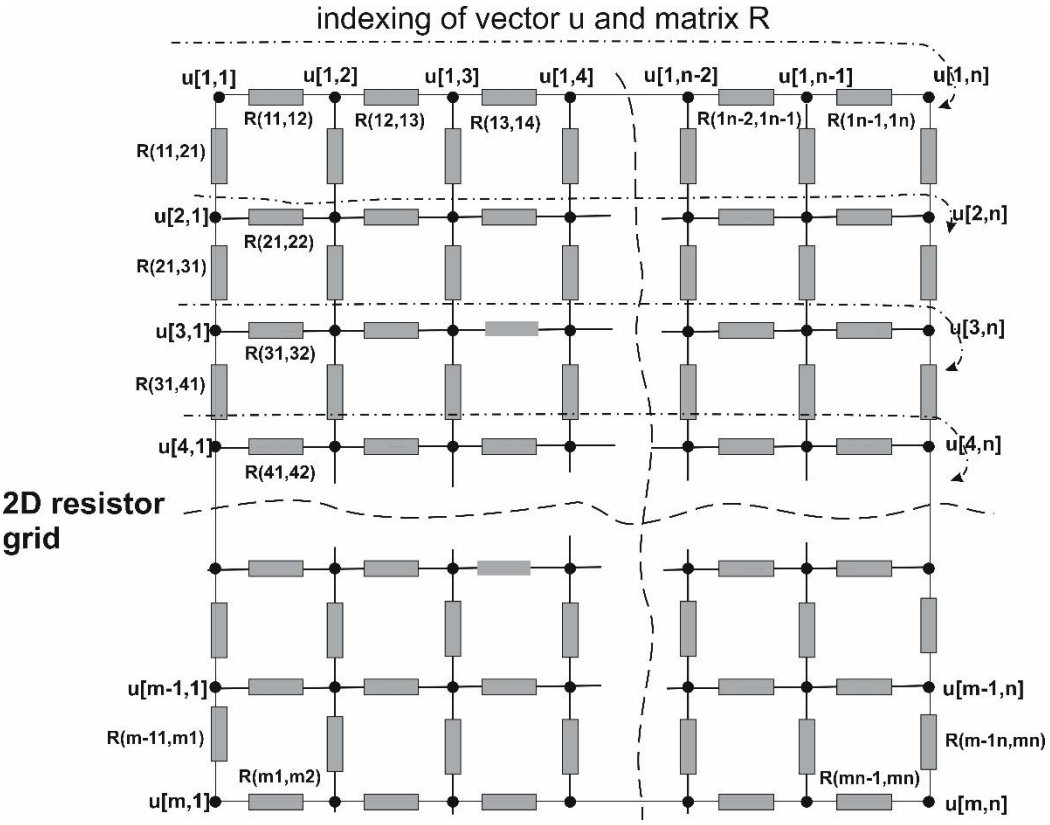


Fig. ME 8.2.1 Indexing of electric potential vector u and resistance matrix R at resistor grid nodes.

The size of a grid is defined as the number of columns and rows, size = $nx \cdot ny$. The bonds between nodes representing resistors are the R-matrix elements of size $(nx - 1) \cdot ny + (ny - 1) \cdot nx = 2 \cdot nx \cdot ny - nx - ny$. The electric potential of the node lying in the *i*-th row and the *j-th* column is denoted by *U[i j]*. One denotes the resistance connecting nodes (*i,j*) and (*k,l*) nodes by *R(ij kl)* . The total current flowing in/out the node (*i,j*) is *B(ij).* The sum of all currents for each node of the grid is equal to zero, which follows from the Kirchhoff's current law:

$$\frac{1}{R_{ij,i+1j}} \cdot (U[i\,j] - U[i+1\,j]) + \frac{1}{R_{ij,i-1j}} \cdot (U[i\,j] - U[i-1\,j]) + \frac{1}{R_{ij,j+1j}} \cdot (U[i\,j] - U[i\,j+1]) + \frac{1}{R_{ij,jj-1}} \cdot (U[i\,j] - U[i\,j-1]) = B(ij) \qquad \text{(ME 8.2.1)}$$

The linear system of equations (ME 8.2.1) corresponds to the *Kirckhoff matrix* A which elements, when $i \neq j$ are conductances with sign minus and equal the negative reciprocals of the resistor R-matrix elements $-\frac{1}{R_{ij,ij}}$. When $i = j$ the diagonal element of A are $\sum_{i \neq j} \frac{1}{R_{ij,ij}}$ (Curtis E.& J. Morrow, 2000. Inverse problems for electrical networks, vol. 13.

World Scientific Publishing Co. Pte. Ltd.). Only in two points $B(ij)$ where the external electric potential is applied, the current is +I (source) and –I (sink), all other elements of vector *B(ij)* are zeros. The mathematical formulation of the problem is reduced to the solution of the linear system of equations for the potentials at nodes *U[ij]* for the given values of

resistance matrix *R(ij kl)* and electric currents *B(ij)*. The matrix R contains the information of nodes and the resistance of bonds between them, and have a format: [Index node 1, Index node 2, conductivity between nodes 1 and 2]. The two index potential at nodes *U[ij]* is converted into the array vector *u(k)* of length, $n_x \cdot n_y$ by the scheme indicated by arrows in Fig. ME8.2.1. In matrix form, the set of equations (ME8.2.1) can be written as: $A \cdot u = B$. When all resistances in the grid has the value 1 the system of equations looks like as follows:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \vdots \\ n_y \\ n_y+1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{array}\begin{bmatrix} \overbrace{\begin{matrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 3 & -1 & \dots & 0 \\ 0 & -1 & 3 & \dots & 0 \\ \vdots & \vdots & -1 & 3 & -1 \\ 0 & 0 & 0 & -1 & 2 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & \ddots & 0 \\ 0 & 0 & 0 & \ddots & \ddots \\ 0 & 0 & 0 & \ddots & \ddots \\ \dots & \dots & \dots & \dots & \dots \end{matrix}}^{n_x} & \overbrace{\begin{matrix} -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 0 & \dots & 0 \\ 0 & 0 & -1 & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 3 & -1 & 0 & \dots & 0 \\ -1 & 4 & \ddots & 0 & 0 \\ 0 & -1 & \ddots & -1 & 0 \\ \ddots & 0 & \ddots & 4 & -1 \\ \ddots & 0 & \ddots & -1 & 3 \\ \dots & \dots & \dots & \dots & \dots \end{matrix}}^{n_x} & \ddots \\ \ddots & \ddots \\ \ddots & \ddots \\ \ddots & \ddots \\ \ddots & \ddots \\ \ddots & \ddots \\ \ddots & \ddots \\ \ddots & \ddots \\ \ddots & \ddots \\ \ddots & \ddots \\ \ddots & \ddots \end{bmatrix} \cdot \begin{bmatrix} U[1\,1] \\ \vdots \\ U[1\,n] \\ U[2\,1] \\ \vdots \\ U[2\,n] \\ \vdots \\ \vdots \\ \vdots \\ U[m-1\,n] \\ U[m\,n] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ +I \\ \vdots \\ -I \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

, where ±I is non-zero at nodes with applied potential. The matrix *A* is five diagonal matrix of size ($n_x \cdot n_y$)× ($n_x \cdot n_y$) and in the code it is defined as a sparse matrix. In each row there are at least 5 non-zero elements. The sum of elements in a row is 0:

The next situation is when the nodes of the grid are connected by resistances 1 and 10^6, i.e. the conductance between adjacent nodes with a certain probability is 1 or 0. The input of the code is the percentage of zero conductivity bonds in the grid.

When the determinant of the A matrix is 0, the Kirckhoff matrix is singular, then the special procedure of the matrix inversion is applied (see the code below). The MATLAB code is as follows:

```
% Author: Viktor NAWA(R) University Frankfurt am Main
clc; clear all; close all;
nx = 25;          % number of grid nodes in X-direction
ny = 25;          % number of grid nodes in Y-direction
k1 = [2,2];       % location of 1-st electrode (+) [x,y]
k2 = [nx-1,ny-1];     % location of 2-d electrode (-) [x,y]
current=1; % impose the electric current between electrodes
% Initialisation of electric field
k=0;
for j=1:1:ny
    for i=1:1:nx
        k=k+1;
        I(i,j) = k;
        P(k).n = zeros(1,nx*ny);
        Ir(k,:) = [i,j];
    end
end
k=0;
% finding adjacent grid nodes and give an idex to them
% define matrix R which contains the information of resistivity between
% two adjacent grid nodes, its format is: [Index Point 1, Index Point 2,
bond resistivity
for j=1:1:ny
    for i=1:1:nx-1
```

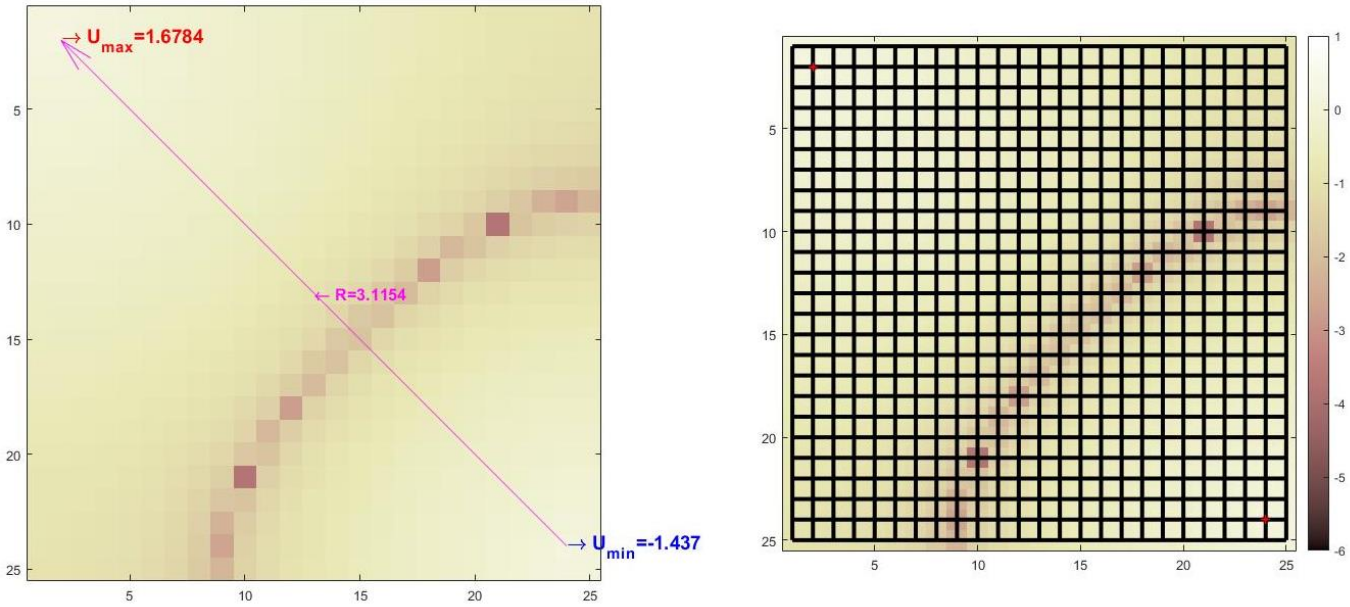
```
        k=k+1;
        R(k,:) = [I(i,j),I(i+1,j),1];
        P(I(i,j)).n(I(i,j)) = k;
        P(I(i,j)).n(I(i+1,j)) = k;
        P(I(i+1,j)).n(I(i,j)) = k;
        P(I(i+1,j)).n(I(i+1,j)) = k;
    end
end
for i=1:1:nx
    for j=1:1:ny-1
        k=k+1;
        R(k,:) = [I(i,j),I(i,j+1),1];
        P(I(i,j)).n(I(i,j)) = k;
        P(I(i,j)).n(I(i,j+1)) = k;
        P(I(i,j+1)).n(I(i,j)) = k;
        P(I(i,j+1)).n(I(i,j+1)) = k;
    end
end
prompt = 'What is the percentage of ~0 conductivity bonds? ';
%input percentage of zero conductance elements
percent = input(prompt)
prompt = 'What is the relative resitivity of ~0 conductivity bonds? ';
%input percentage of zero conductance elements
Rmax = input(prompt)
[num, dem] = rat(percent/100);% convert percentage into fraction
K=num*length(R)/dem; N= (dem-num)*length(R)/dem;
vec=ones(N+K,1);%initialize vector of R=1 of length(R)
 positions=[1:K]; vec(positions)=Rmax;%insert ~zero conductivity in vec
 shuffle = @(v)v(randperm(numel(v)));%shuffle function of array elements
 vec=shuffle(vec);% shuffle K 1e6 and N ones among K+N positions of vec
 R(:,3)=vec; % set random in (percentage) grid points of conductance ~0,
 % and in the rest grid points the conductance set to 1
% compose the system of the Kirchhoff electric current equations, finding
% solution by keeping balance of electric currents at nodes: +1 in sorce -1
% in sink and everywhere else is 0. Vector of electric currents is b.
for i=1:1:nx*ny
    v = P(i).n~=0;
    l = 1:1:nx*ny;
    p = P(i).n(v);
    l = l(v);
    P(i).n = l;
end
ni=i;
A = sparse(i,i); %use sparce matrix
for i=1:1:nx
    for j=1:1:ny
        p = P(I(i,j)).n;
        a=0;
        for k=1:1:length(p)
            if R(p(k),3)~=0
                A(I(i,j),p) = -1/R(p(k),3);% set in connected nodes the
conductnace -1/R
                a = a+1/R(p(k),3);% sum of conductances in the matrix row
            end
        end
        A(I(i,j),I(i,j)) = A(I(i,j),I(i,j))+a;% sum of conductances in the
matrix row
        %excluding the the node i=j
    end
end
det(A)
B = zeros(nx*ny,1);
```

```
B(I(k1(1),k1(2))) = current;% source of electric current =+1
B(I(k2(1),k2(2))) = -current;% sink of electric current =-1
if det(A)~=0
u = A\B; %if matrix A is not singular
else
 %If matrix A is singular : Perform SVDS on A
[W,S,V] = svds(A);%perform singular value decomposition
% A == W*S*V' returns the left singular vectors W,
%diagonal matrix S of singular values, and right singular vectors V.
% Calc number of singular values
sing = diag(S);    % vector of singular values
tolerance = max(size(A))*eps(max(sing));% eps(sing)returns the positive
%distance from abs(sing) to the next larger floating-point number of the
%same precision as sing
m = sum(sing>tolerance);
% Define spaces
Up = W(:,1:m);
Vp = V(:,1:m);
SpInv = spdiags( 1.0./sing(1:m), 0, m, m );%extracts all nonzero diagonals
%from the matrix A
% Calc AInv such that u = AInv * b
AInv = Vp * SpInv * Up';
u = AInv * B;    % calculation of potential vector
end
% convert the vector u(i) into the matrix U(i,j)
for i=1:1:nx*ny
    U(Ir(i,1),Ir(i,2))=u(i);
end
% plotting of electric potential U
subplot(1,2,1)
    colormap(jet(256))
        imagesc((log10(abs(U)))')
    axis equal tight
    caxis([-6,1])
    hold on
   mx=max(U(:));mn=min(U(:)); %find maximum and minimum of electric
potential
   [xmax ymax]=find(U==mx); [xmin ymin]=find(U==mn);% ccordinates of max
and min
       t1=num2str(mx); t1=strcat('\rightarrow U_m_a_x= ',t1);
    t3=num2str(mn); t3=strcat('\rightarrow U_m_i_n= ',t3);
%calculation of the resistance between two electrodes
deltaU=U(I(k1(1),k1(2)))-U(I(k2(1),k2(2)));
%calculate the resistance between two points with applied electric current
    Resistance=abs(deltaU/current);
    t2=num2str(Resistance);t2=strcat('\leftarrow R= ',t2);
    k3(1)=round((k1(1)+k2(1))/2); k3(2)=round((k1(2)+k2(2))/2);
    xt=[xmax xmin k3(1)]; yt=[ymax ymin k3(2)]; str={t1,t3,t2};
    t=text(xt,yt,str);
     t(1).Color='red'; t(2).Color='blue'; t(3).Color='magenta';
     t(1).FontSize=14;t(2).FontSize=14; t(3).FontSize=12;
     t(1).FontWeight='bold';t(2).FontWeight='bold'; t(3).FontWeight='bold';
    q=quiver(k2(1),k2(2),k1(1)-k2(1),k1(2)-k2(2),1); q.Color='magenta';
    hold on
% plotting electric circuit of grid resistors
subplot(1,2,2)
    colormap(pink)
    imagesc((log10(abs(U)))')
    hold on;
    for i=1:1:size(R,1)
        q='k';
        s=1;
```

```
        if R(i,3)==Rmax
            q='r';
            s=0;
        end
        if s==1

plot([Ir(R(i,1),1),Ir(R(i,2),1)],[Ir(R(i,1),2),Ir(R(i,2),2)],q,'linewidth',
3);
        end
          k = [k1;k2];sz=350;
        scatter(k(:,1),k(:,2),'r*')
    end
% checking the solution conversion
ff = A*u; ff(B~=0)=0;
ff = ff'*ff
caxis([-6,1])
hold off;
axis equal tight
colorbar
```

The results of the simulation are shown in Fig. ME8.2.2 for 100 of R=1 (upper panel) and for 65% of R=1 (lower panel. The coordinates of *k1* und *k2* are the points where the electrical potential is applied, i.e. the source and the sink of electric current. Left plot represents the magnitude of electric potentials U in $\log_{10}$(abs(U)) units throughout the grid. Right plot is the network of resistors: black stretches connect the nodes with the electrical conductivity =1, the absence of black stretches indicate on the infinite resistance or zero conductivity. The solution conversion of is estimated by *ff*, the norm of the solution vector.



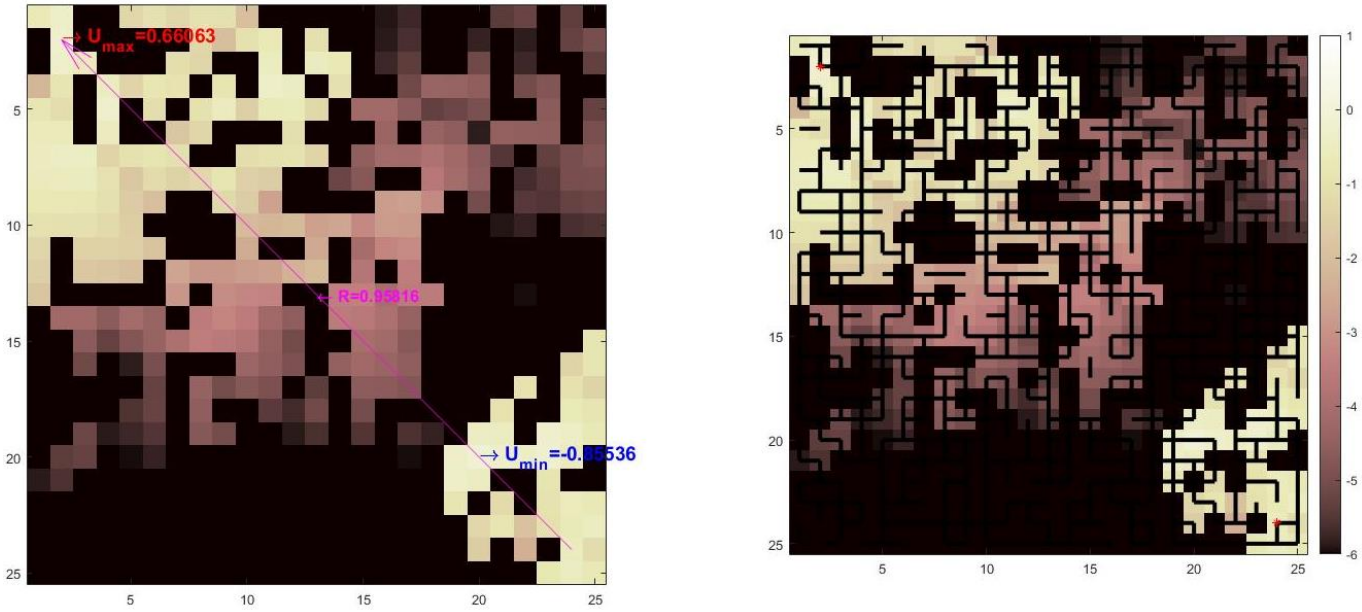


Fig. ME9.1 MATLAB calculations of electrical potential in 2D grid of resistors having conductivities: upper panels stand for 100% of R=1. Lower panels stand for 65% of R=1 and 35% of R=10^6. The electric field applied to the nodes marked by two red stars.

MATLAB Exercise 9. Dielectric sphere in uniform electric field

Consider a uncharged dielectric sphere of radius *a* having the dielectric constant ε_1, surrounded by a medium of dielectric constant ε_2 , and the uniform electric field E_0 is applied The origin of coordinate system is taken to be at the sphere center and z-axis is along E_0. Let Φ_1 and Φ_2 be two potential functions inside and outside the sphere, respectively. At far distances from the sphere, the electric field is uniform and equal to: $E_2 = E_0 \cdot \overrightarrow{e_z}$, which corresponds to the potential: $\boldsymbol{\Phi}_2(\infty) = -E_0 \cdot r \cdot cos\theta$. The solution of this problem must satisfy the following boundary conditions:

(i) $\nabla^2\boldsymbol{\Phi}_1 = 0$ and $\nabla^2\boldsymbol{\Phi}_2 = 0$, both inside and outside respectively, since there is no space charges inside and outside the sphere.

(ii) $\boldsymbol{\Phi}_1$ must remain finite for all $r \leq a$, and $\boldsymbol{\Phi}_2$ must also remain finite at infinity.

(iii) $\boldsymbol{\Phi}_1 = \boldsymbol{\Phi}_2$ for $r = a$ at all angles θ.

(iv) The normal component of displacement vector **D** must be continuous at $r = a$, i.e., $D_{1n} = D_{2n}$.

The potentials outside and inside the sphere can be written as:

$\boldsymbol{\Phi}_2 = - E_0 \cdot r \cdot \cos\theta + (\frac{A}{r^2}) \cdot \cos\theta$ for $r \geq a$, (ME 9.1.1)

$\boldsymbol{\Phi}_1 = B \cdot r \cdot \cos\theta + (\frac{C}{r^2}) \cdot \cos\theta$ $r \leq a$ (ME 9.1.2).

The potential would be finite at the origin ($r = 0$), therefore, C=0, hence

$\boldsymbol{\Phi}_1 = B \cdot r \cdot \cos\theta$ (ME 9.1.3)

According to the boundary condition (iii), $\boldsymbol{\Phi}_1 = \boldsymbol{\Phi}_2$, for $r = a$

$B \cdot a \cos\theta = -E_0 \cdot a \cos\theta + (\frac{A}{a^2}) \cos\theta$ or $B = -E_0 + \frac{A}{a^3}$ (MB 9.1.4)

According to the boundary condition (iv),

$D_{1n} = D_{2n}$ or $\varepsilon_1 \cdot \varepsilon_0 E_{1n} = \varepsilon_2 \cdot \varepsilon_0 \cdot E_{2n}$

According to the definition, $E_n = -\partial\boldsymbol{\Phi}/\partial r$, and hence: $- \varepsilon_1 (\partial\boldsymbol{\Phi}_1/\partial r)_{r=a} = -\varepsilon_2 (\partial\boldsymbol{\Phi}_2/\partial r)_{r=a}$, then, it follows: $- \varepsilon_1 \cdot B \cos = -\varepsilon_2 [-E_0 \cos\theta - (\frac{2A}{a^3}) \cos\theta]$, or

$\varepsilon_1 \cdot B = -\varepsilon_2 \cdot (E_0 + \frac{2A}{a^3})$ … (ME 9.1.5)

Solving (ME 9.1.4) and (ME 9.1.5), one gets: $A = \frac{(\varepsilon_1-\varepsilon_2)}{\varepsilon_1+2\varepsilon_2} \cdot E_0 \cdot a^3$ and $B = -\frac{3\varepsilon_2}{\varepsilon_1+2\varepsilon_2} \cdot E_0$. Thus the potential functions inside and outside the spheres are:

$\boldsymbol{\Phi}_1 = -\frac{3\varepsilon_2}{\varepsilon_1+2\varepsilon_2} \cdot E_0 \cdot r \cdot \cos\theta$ (ME 9.1.6),

$\boldsymbol{\Phi}_2 = -\left[1 - \frac{(\varepsilon_1-\varepsilon_2)}{\varepsilon_1+2\varepsilon_2} \cdot \frac{a^3}{r^3}\right] \cdot E_0 \cdot r \cdot \cos\theta$ (ME 9.1.7),

The electric field at any point is given by:

inside the sphere $r<a$: $E_1 = -\partial\boldsymbol{\Phi}_1/\partial z = 3\varepsilon_2/(\varepsilon_1 + 2\varepsilon_2)\, E_0 = \left[1 - \frac{(\varepsilon_1-\varepsilon_2)}{\varepsilon_1+2\varepsilon_2}\right] \cdot E_0$,

and

outside the sphere $r>a$: $E_2 = -\partial\boldsymbol{\Phi}_1/\partial z = \left[1 + 2 \cdot \frac{(\varepsilon_1-\varepsilon_2)}{\varepsilon_1+2\varepsilon_2} \cdot \frac{a^3}{r^3}\right] \cdot E_0$ (ME 9.1.8).

The electric field outside the sphere is equivalent to the applied field E_0 plus the field of electric dipole at the sphere center having the dipole moment:

$\vec{p} = 4\pi\varepsilon_0 \cdot \varepsilon_2 \frac{(\varepsilon_1-\varepsilon_2)}{\varepsilon_1+2\varepsilon_2} E_0 \cdot a^3$ … (ME 9.1.9),

which is oriented in the direction of applied field. The electric field of dipole p is: $\vec{E}_{dipole} = \frac{1}{4\pi\varepsilon_0} \cdot \frac{2\vec{p}}{r^3}$. When a dielectric sphere acquires the dipole moment in electric field, it is polarized. The polarization is defined as the dipole moment per unit volume i.e.,

$\vec{P} = \vec{p}/(4/3)\pi a^3 = 3 \cdot \varepsilon_0 \cdot \varepsilon_2 \frac{(\varepsilon_1-\varepsilon_2)}{\varepsilon_1+2\varepsilon_2} \cdot E_0$ (ME 9.1.10),

or

$\frac{\vec{P}}{3\varepsilon_0 \cdot \varepsilon_2} = \frac{(\varepsilon_1-\varepsilon_2)}{\varepsilon_1+2\varepsilon_2} \cdot E_0$ (ME 9.1.11).

Thus, the electric field inside the sphere is reduced relative to the applied field E_0 by:

$\overrightarrow{E_d} = -\frac{\vec{P}}{3\varepsilon_0 \cdot \varepsilon_2}$ (ME 9.1.12),

where $\overrightarrow{E_d}$ is the field due to dielectrics ε_2 inside the sphere.

Here below is MATLAB program to calculate the electric field inside and outside the polarized sphere under a uniform external electric field.

```
  global  eps1 eps2 a E0
     eps2=25.0;      % dielectric comstant inside the sphere
    eps1=2.;      % dielectric constant outside the sphere
    E0=1.0;%external field
    a=1.0;% radius of the sphere
    n=3; %scale factor of the plot
    xmin=-n*a; xmax= n*a; ymin=-n*a; ymax= n*a;
    x = linspace(xmin,xmax,200); y=x;
        [X,Y] = meshgrid(x,y);
    R=@(x,y) sqrt(x.^2+y.^2);% Radius vector calculatios
       V=voltage(X,Y,R);%function to calculate the electric
potential
    [Ex,Ey] = gradient(-V); % components of electric field
    figure;
        contour(X,Y,V,[-n*a:a/n:n*a]);
      colormap('jet');
       axis tight; caxis([-n*a,n*a]);
    hold on
    streamslice(X,Y,Ex,Ey);
m = 100; %number of points to draw a circle
angle = 0:2*pi/m:2*pi;% vector of angles at which points are drawn
x = a*cos(angle); y = a*sin(angle);   % Coordinates of the circle
```

```
plot(x,y,'r', 'LineWidth',2);
    hold off
    end
function v = voltage(x,y,R)% to calculate the electric potential
    global  eps1 eps2 a E0
v=-E0.*x.*(1-(eps1-
eps2)./(2.*eps2+eps1).*(a^3)./R(x,y).^3).*heaviside(R(x,y)-a)
-E0.*x.*(1-(eps1-eps2)./(2.*eps2+eps1)).*heaviside(-(R(x,y)-a));
% calculation of the potential inside and outside the sphere
end
```

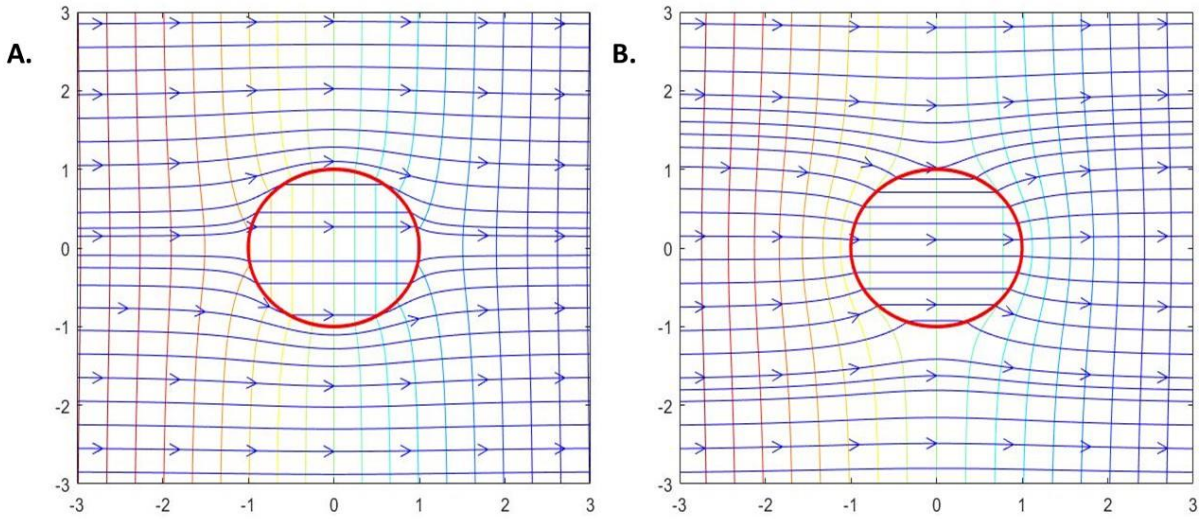


Fig. ME9.1 MATLAB calculations of 2D electric field of polarized sphere in uniform external electric field, ε_1 is the dielectric constant of medium outside sphere, ε_2 is the dielectric constant of medium inside sphere: A. $\varepsilon_1=25 > \varepsilon_2=5$; B. $\varepsilon_1=5 < \varepsilon_2=25$. The vertical lines are equipotential surfaces, horizontal arrow lines are electric field lines. The density of arrow lines inside the sphere is proportional to the electric field strength. In the case (A) the dielectric field $\overrightarrow{E_d} = -\frac{\vec{P}}{3\varepsilon_0 \cdot \varepsilon_2} = -\frac{(\varepsilon_1-\varepsilon_2)}{\varepsilon_1+2\varepsilon_2} \cdot E_0$ is negative and is subtracted from the applied field E_0. In the case (B) it is positive and added to the field E_0.

In order to program the electric potential $\boldsymbol{\Phi}_{1,2}$ (ME 9.1.6) and (ME 9.1.7) consisting of two functions the *Heaviside piecewise constant function H(x)* has been used:

$H(x) = \begin{Bmatrix} 0 \; at \; x < 0 \\ \frac{1}{2} \; at \; x = 0 \\ 1 \; at \; x > 0 \end{Bmatrix}$. The derivative of the step function *H(x)* is the *Dirac delta function* $\delta(x)$. The electric potentials in two regions $\boldsymbol{\Phi}_1(x<a)$ and $\boldsymbol{\Phi}_2(x>a)$ may be expressed with the help of the Heaviside function as follows: $\boldsymbol{\Phi}(x) = \boldsymbol{\Phi}_1(x) \cdot H(-(x-a)) + \boldsymbol{\Phi}_2(x) \cdot H(x-a)$.

****

MATLAB Exercise 10. Computation of electric current loop magnetic field

Electric charges interact with the electric field that they produce. Since moving charges, i. e. electric current, interact with the magnetic field, so it also creates its own magnetic field. The equation used to calculate the magnetic field produced by electric currents is the well-known *Biot-Savart law*, which enables to calculate the magnitude and direction of the magnetic field

produced by current in a wire. The Biot-Savart law states that at any point P (Fig. ME10.1.1), the magnetic field $d\vec{B}$ due to the length element $\vec{I} \cdot dl$ of current-carrying wire is given by $d\vec{B} = \mu_0 \cdot \mu_r \cdot \frac{[\vec{I}x\vec{r}]}{4\pi r^3} \cdot dl$, where $\mu_0 = 1.257 \cdot 10^{-6}\ H/m$ is the vacuum magnetic permeability and μ_r is the relative magnetic permeability of medium. A current loop produces the magnetic field produced by a current loop at point *P* is given by the Biot-Savart law: $d\vec{B} = \mu_0 \cdot \mu_r \cdot \oint \frac{[\vec{I}x\vec{r}]}{4\pi r^3} \cdot dl = \mu_0 \cdot \mu_r \cdot \vec{I} \cdot \oint \frac{sin\theta}{4\pi r^2} \cdot dl$.

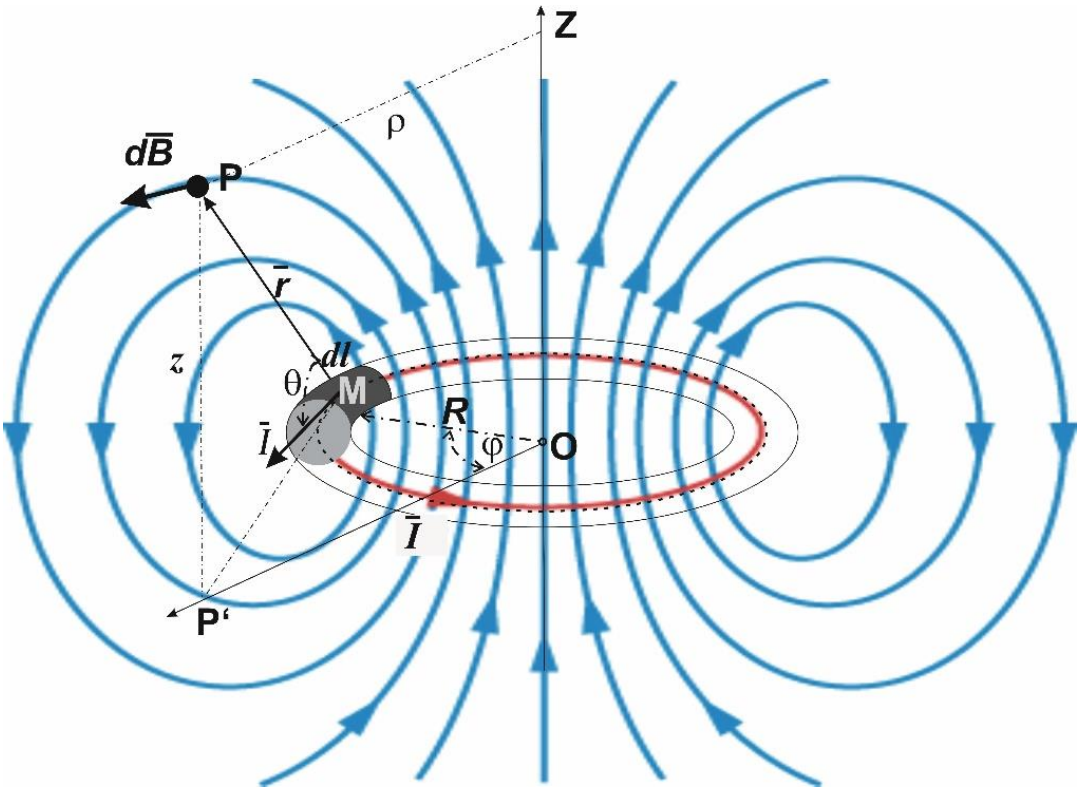


Fig. ME 10.1.1 Electric loop wire segment dl carrying electric current $\vec{I}$. Loop length element *dl*, radial direction $\vec{r}$, and angle θ between them are indicated.

If point P lies on the symmetry axis of loop, then the integration of current elements in polar coordinates $dl=R\cdot d\varphi$ gives:

$\vec{B} = B_z = \mu_0 \cdot \mu_r \cdot \frac{I}{2} \cdot \frac{R^2}{(R^2+z^2)^{3/2}} = B_0 \cdot \frac{R^3}{(R^2+z^2)^{3/2}}$ (ME10.1.1),

where $B_0 = \mu_0 \cdot \mu_r \cdot \frac{I}{2R}$ is the magnetic field in the centre of electric current loop (at *z*=0)

If one denotes the distances PP'=OZ=z, and PZ=P'O=ρ, then

MP'=$\sqrt{R^2 + \rho^2 - 2R \cdot \rho \cdot \cos\varphi}$. From the geometry in Fig. ME10.1.1 it follows that in the Cartesian coordinate system $\overrightarrow{dl} = R \cdot\ d\varphi \cdot \{-\sin\varphi, \cos\varphi\}$, and $\vec{r} = \{\rho - R \cdot \cos\varphi, -R \cdot \sin\varphi, z\}$. The vector product of them is given by $[\overrightarrow{dl}x\vec{r}] = R \cdot d\varphi \cdot \{z \cdot \cos\varphi, z \cdot \sin\varphi, R - \rho \cdot cos\varphi\}$. The magnetic field in axisymmetric cylindrical coordinate system is: $d\vec{B} = \{dB_z,\ dB_r\} = \frac{B_0}{2} \cdot R^2 \cdot \frac{\{z,\ \ R-\rho\cdot\cos\varphi\}}{\left(\sqrt{R^2+\rho^2+z^2-2R\cdot\rho\cdot\cos\varphi}\right)^3} \cdot d\varphi$. After integration of current elements in the loop plane the following identity may be written for $\vec{B}$ in cylindrical coordinates:

$$\vec{B}(\rho,z)=\{B_z,\ B_r\}=\begin{cases} B_z=\frac{B_0\cdot R}{\pi\cdot\sqrt{(R+\rho)^2+z^2}}\cdot\left[E(k)\cdot\frac{R^2-\rho^2-z^2}{(R-\rho)^2+z^2}+K(k)\right] \\ B_r=\frac{B_0\cdot z\cdot R}{\pi\cdot\rho\cdot\sqrt{(R+\rho)^2+z^2}}\cdot\left[E(k)\cdot\frac{R^2+\rho^2+z^2}{(R-\rho)^2+z^2}-K(k)\right] \end{cases} \quad \text{(ME10.1.2)},$$

where $K(k^2)=\int_0^{\frac{\pi}{2}}\frac{d\theta}{\sqrt{1-k^2\cdot\sin^2\theta}}$ is the complete elliptic integral function of the first kind,

$E(k^2)=\int_0^{\frac{\pi}{2}}\sqrt{1-k^2\cdot\sin^2\theta}\cdot d\theta$ is the complete elliptic integral function of the second kind,

and the modulus $k=2\cdot\sqrt{\frac{R\cdot\rho}{(R+\rho)^2+z^2}}$.

In MATLAB code shown below the function `[Bz,Br]= m_field_loop(i,R,ro,z)` is calculated for two components of magnetic field, B_r and B_z, using the build-in function `[K,E] = ellipke(k²)` for two elliptic integrals, *K(k²)* and *E(k²)*. The results of calculations are presented in Fig. Fig. ME 10.1.2.

```
clc; close all; clear all;
%-------------------------------------------------------------------
------
%---electric loop is in the z=0 plane and magnetic field B is
evaluated
%-------------at every point in the ro-Z plane----------------------
--
%-------------------------------------------------------------------
------
Nz=201;  % No. of grids in Z-axis
Nro=51;  % No. of grids in Y-axis
Ra=0.25;    % Radius of the coil in the X-Y plane
I=10;    % current in the coil
phi=-pi/2:2*pi/(Nro-1):3*pi/2; % For describing a circle (loop)
Xc=Ra*cos(phi); % X-coordinates of the loop
Yc=Ra*sin(phi); % Y-coordinates of the loop
Zc=zeros(length(phi)); % Z-coordinate of the loop

figure(1)%plot electric loop
plot3(Xc,Yc,Zc,'linewidth',3)
axis([-2*Ra 2*Ra -2*Ra 2*Ra -2*Ra 2*Ra])
daspect([1 1 1]);
xlabel('X-axis','fontsize',14)
ylabel('Y-axis','fontsize',14)
zlabel('Z-axis','fontsize',14)
hold on
quiver3(0,0,-2*Ra,0,0,4*Ra,'linewidth',3); %plot the direction of
dipole moment
hold on
title('electric loop co-ordinates','fontsize',14)
h=gca; get(h,'FontSize')
set(h,'FontSize',14)
h = get(gca, 'ylabel');fh = figure(1);set(fh, 'color', 'white');
grid on
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ro=0:Ra/(Nro-1):2*Ra; %vector of ro
z=-2*Ra:Ra/(Nz-1):2*Ra; %vector of z
[RO,Z]=meshgrid(ro,z);
[Bz,Br]=m_field_loop(I,Ra,RO,Z);%calculate 2 components of vector B
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
figure(2)
lim1=min(min(Bz)); lim2=max(max(Bz));
steps=(lim2-lim1)/100; %defines number of contour lines
contour(Z,RO,Bz,lim1:steps:lim2)
axis([-2*Ra 2*Ra 0 2*Ra])
xlabel('Z-axis','fontsize',14)
ylabel('\rho-axis','fontsize',14)
title('Bz component','fontsize',14)
colorbar('location','eastoutside','fontsize',14);
h=gca; get(h,'FontSize')
set(h,'FontSize',14)
h = get(gca, 'ylabel');fh = figure(2);set(fh, 'color', 'yellow');
grid on
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(3)
contour(Z,RO,Br,lim1:steps:lim2)
axis([-2*Ra 2*Ra 0 2*Ra])
xlabel('Z-axis','fontsize',14)
ylabel('\rho-axis','fontsize',14)
title('Br component','fontsize',14)
colorbar('location','eastoutside','fontsize',14);
h=gca;get(h,'FontSize')
set(h,'FontSize',14)
h = get(gca, 'ylabel');fh = figure(3); set(fh, 'color', 'green');
grid on
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Bz,Br]= m_field_loop(i,R,ro,z)
%%%%%%%%%%%%%%%%%%%%%%%%
% [Bz,Br]=m_field(i,R,r0,z)
% Bz - axial comp. of mag. field
% Br - radial comp. of mag. field
% i=coil current;
% R=coil radius;
%ro= distance of a point from Z-axis
%z= vertical co-ordinate along Z-axis
%%%%%%%%%%%%%%%%%%%%%%%%
mu0=4*pi*10^-7;% vacuum permeability
B0=(i*mu0)/(2*R);% magnetic field Bo in the center of electric loop
        if (ro==0)
         Bz=B0.*R^3./((R.^2+z.^2).^(3./2));
            Br=0;
        else
            k=sqrt(4.*ro.*R./((R+ro).^2+z.^2));
            [K,E] = ellipke(k.^2); %calculate elliptic integrals
 Bz=B0./pi./sqrt((R+ro).^2+z.^2).*(E.*(R.^2-ro.^2-z.^2)./((R-
ro).^2+z.^2)+K);
 Br=B0.*z./ro./pi./sqrt((R+ro).^2+z.^2).*(E.*(R.^2+ro.^2+z.^2)./((R-
ro).^2+z.^2)-K);
        end
                    end
```

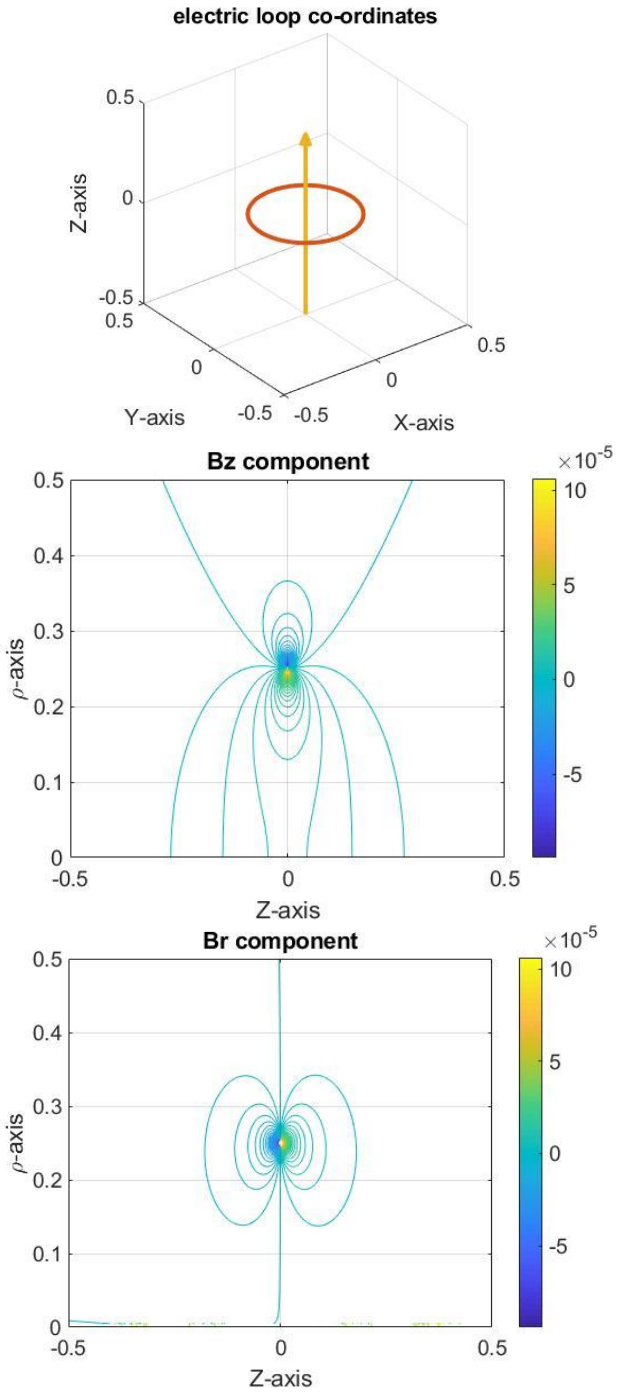


Fig. ME 10.1.2 Electric current loop in 3D (upper panel). Contour lines are the axial (middle panel) and radial (lower panel) $\vec{B}$ components.

MATLAB Exercise 11.1 Thermal inertia of surface rocks

When the surface heat flux is a harmonic function of time $\propto \cos(\omega \cdot t)$, then the temperature in half space beneath the surface at depth *x* should satisfy 1D-heat transport equation $\frac{\partial T}{\partial t} = \kappa \cdot \frac{\partial^2 T}{\partial x^2}$ with the surface boundary condition at *x*=0: $(-\frac{\partial T}{\partial x} + h \cdot T)_{x=0} = (h + c) \cdot \cos(\omega \cdot t)$. This relationship may be considered as the periodic temperature boundary condition, when the thermal conductivity of half space is low (i.e. *h*>>c~1), or as the periodic heat flux boundary condition, when the thermal conductivity is high (i.e. *h*<<c~1). The solution for the steady

state temperature distribution as a function of time and depth is also the periodic function of time but shifted by phase δ relative to the phase of boundary oscillating condition. After introducing the "thermal" wave length $1/\mu = \sqrt{\frac{2\cdot\kappa}{\omega}}$, the temperature solution may be written in the form (Carslaw & Jaeger, 1959):

$T(x,t) = T_0 \cdot \cos(\omega \cdot t - \mu \cdot x + \delta) \cdot e^{-\mu \cdot x} \cdot A(h,\mu)$ (ME 11.1),

where $\delta = arctan\left(\frac{\mu}{h+\mu}\right)$, and $A(h,\mu) = \frac{1}{\sqrt{\left(1+\frac{\mu}{h}\right)^2+\left(\frac{\mu}{h}\right)^2}}$. In the case of periodic boundary temperature δ=0 and $A=T_0$, in the case of periodic boundary heat flux: $\delta=\pi/4$ and $A = \frac{q_0}{\lambda\cdot\mu}$, where q_0 is the amplitude of flux on the surface. In the case when the surface boundary condition is of the mixed type, the phase shift δ is between 0 and π/4. The results are presented in Fig. ME11.1.

```
lamda=5;q0=10;
kappa=0.5;omega=0.5;
mu=sqrt(omega/2/kappa);
x=linspace(-5,0,100);% create a vector x
t=linspace(0,pi*5,21);% create a vector t
[X,Y]=meshgrid(x,t);% create a meshgrid of two vectors x and t
Z=q0/lamda/mu/sqrt(2)*exp(X.*mu).*cos(Y.*omega-pi/4+X.*mu);
figure
waterfall(X,Y,Z)
xlabel('depth','Fontsize',18)
ylabel('time','Fontsize',18)
zlabel ('Temperature','Fontsize',18)
hold on
```

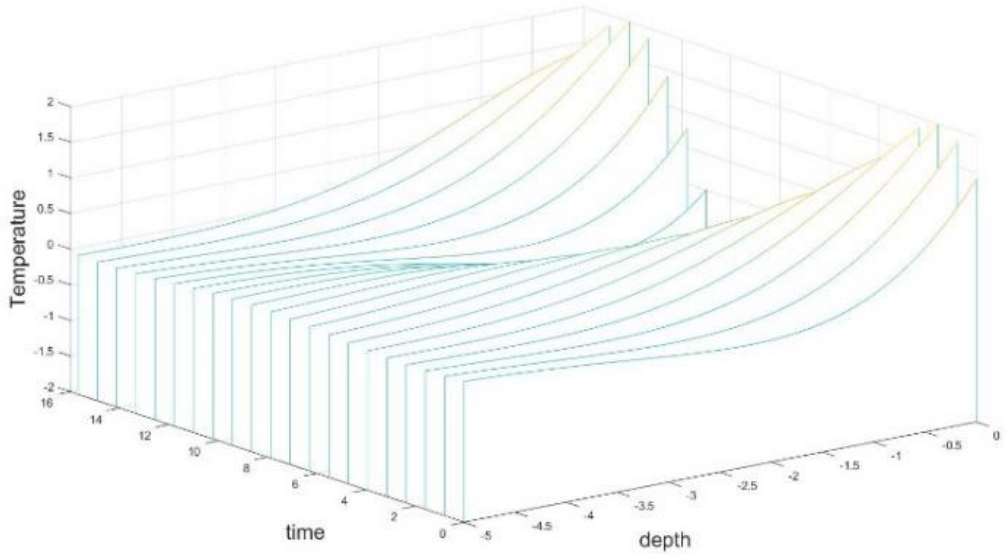


Fig. ME11.1 Time dependent solution of 1D temperature distribution in half space with periodic time dependent heat flux on the surface.

****

MATLAB Exercise 11.2 Solution of time dependent 1D thermal transport equation

If one considers 1D-rod with perimeter Π and cross section area A, then, the increment of heat in length element dx is given by $\lambda \cdot A \cdot \frac{\partial^2 T}{\partial x^2} \cdot dx$, and the amount of heat lost through the side surface is $\Lambda \cdot \Pi \cdot (T - T_0) \cdot dx$, where T_0 is the surrounding medium temperature and Λ is the heat exchange (transfer) coefficient in W/m²/°K. The total increase of the heat in volume per unit of time is $\cdot C_p \cdot \frac{\partial T}{\partial t} \cdot A \cdot dx$. Thus, 1D-thermal transport equation with the lateral heat exchange in medium at temperature T_0 is formulated as follows:

$$\rho \cdot C_p \cdot \frac{\partial T}{\partial t} = \lambda \cdot \frac{\partial^2 T}{\partial x^2} - \Lambda \cdot \frac{\Pi}{A} \cdot (T - T_0) \qquad \text{(ME 11.2.1).}$$

The following program is written using the semi-discretization method (method of lines). The time t and space coordinate *x* are discretized independently. For each line of x-vector the solution is obtained in respect with time by solving the ordinary differential equation using MATLAB solver *ode15s* (Driscoll, 2009). In the example below, the interval x = [-2,0] mm and the time span *t = [0,10]* sec are considered. 1D-thermal transport equation with the thermal exchange in medium at T=0 is taken in the form:

$$\frac{\partial T}{\partial t} = \kappa \cdot \frac{\partial^2 T}{\partial x^2} - \nu \cdot T \qquad \text{(ME 11.2.2),}$$

where $\kappa = \frac{\lambda}{\rho \cdot C_p}$ is the thermal diffusivity, and $\nu = \frac{\Lambda}{\rho \cdot C_p} \cdot \frac{\Pi}{A}$. The rod is located in medium with T=0, and at the ends *x*=0 and x= -2 the temperature is maintained constant T=0. At t=0 the initial temperature is T=50° in *x* range from [-0.1 0] mm and T=0 elsewhere. The right hand side of (ME 11.2.2) is replaced by the matrix-vector multiplication:

$$\kappa \cdot \frac{\partial^2 T}{\partial x^2} - \nu \cdot T = \kappa \cdot \frac{1}{h^2} \cdot \begin{bmatrix} -2 & 1 & 0 & & \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & \ddots & & \\ & 0 & & -2 & 1 \\ & 0 & & 1 & -2 \end{bmatrix} \cdot [u_i] - \nu \cdot [u_i] \qquad \text{(ME 11.2.3),}$$

where u_i is the temperature vector. ME11.23 is integrated numerically in respect of time as an ordinary differential equation using the solver: `ode15s`. The sample calculation results are presented in Fig. ME 11.2.1.

```
%MATLAB program adapted from [Driscoll, 2009].
n=500; h=2/n;% n is the number of nodes, h is the interval length
kappa=2.5e-3;nu=0.015;
%thermal diffusivity and thermal exchange coefficient
x=(-2+h*(1:n-1)).*1e-3;%generation of x-vector of nodes
D2=toeplitz([-2 1 zeros(1,n-3)]/h^2);
%generation of the three-diagonal matrix
f=@(t,u) kappa*D2*u-nu*u;%differential equation in a matrix form
u0=50*heaviside(x+1e-4);%initial temperature profile
[t,u]= ode15s(f,[0 10],u0);
%solving of the differential equation in respect to time
figure
waterfall(x,t,u)
c.LineWidth = 1; box on
xlabel('depth,m','Fontsize',18)
ylabel('time, sec','Fontsize',18)
zlabel ('Temperature,°K','Fontsize',18)
hold on
```

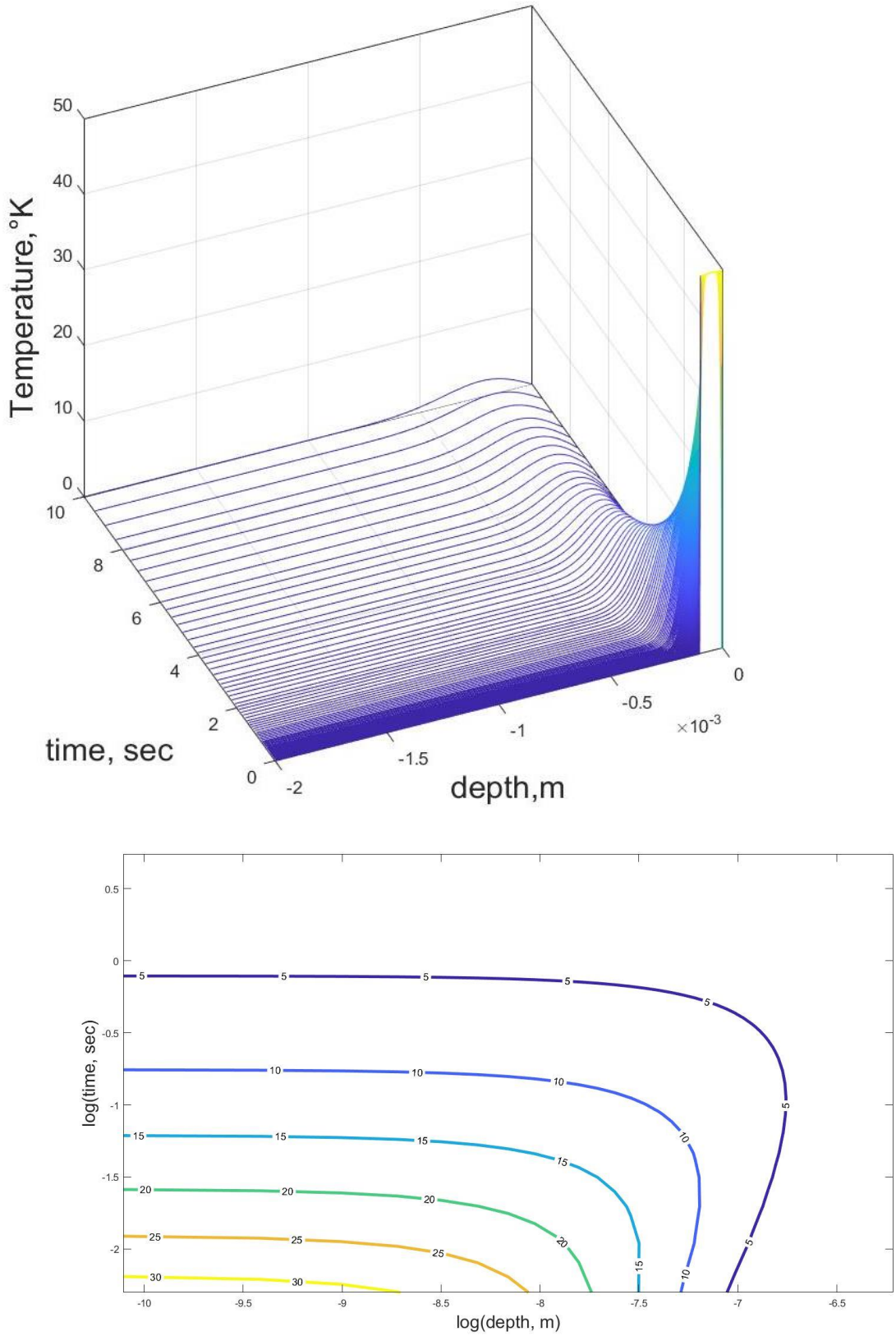


Fig. ME11.2.1 Method of lines realized in 1D for the time dependent thermal transport equation. (upper panel) 3D plot in coordinates depth-time-temperature. (lower panel) Isoclines of temperature solution in log(depth)-log(time) coordinated. .

MATLAB Exercise 12.1 Radioactive decay

This example of MATLAB program illustrates the stochastic character of radioactive decay process using the Monte-Carlo simulation. Starting from *n* nuclides at t=0 and using the time step *dt*, the program generates *n* random numbers between 0 and 1 using the uniform PDF. The probability of nuclide decay is $\lambda \cdot dt$. So, if one counts how many numbers are less than $\lambda \cdot dt$ among the set of *n* numbers, this will be the number of decayed nuclides during the time step *dt*. On the next time step the total number of nuclides is corrected for the number of decayed nuclides, and the MC-simulation is repeated.

```
% To simulate radio active decay by Monte Carlo method % % modified
from Mahesha MG, MIT
_________________INPUT_____________________________________

clc; clear;

nmc=input('Enter number of Monte Carlo simulations (>10000): ');

lambda=input('Enter decay constant \lambda in 1/sec: ');
n=input('Enter number of nuclides at the beginning: ');
dt=1/nmc/lambda;n0=n;%Defining a time-step nt=zeros(nmc,1); %nt
holds MC result

nta=zeros(nmc,1); %nta holds analytic result

tmax=nmc*dt; t=(0:dt:tmax)'; %Define span of time

nta=exp(-lambda*t); %Analytic solution

nt(1)=n; % _Monte Carlo ethod_____________________________

 for i=2:nmc+1

ran=rand(n,1); %random choice of n-numbers between 0 and 1
count=length(ran(ran<=1/nmc));

% count of numbers which are <= lambda*dt

n=n-count;% a new count of nuclides

 if(n<=0)

        break;

    end

    nt(i)=n; end

% ____________________________________________________
plot(t,nt/n0,'r',t,nta,'b');

 xlabel('\lambda*t, dimensionless time');
```

```
ylabel('Relative number of nuclides');

grid on

hold on

legend('MC-simulation','e^{-\lambda*t}')
```

When the starting number of nuclides is small the stochastic method to estimate the decayed number of nuclides deviates from the exact exponential solution (Eq. 12.3).

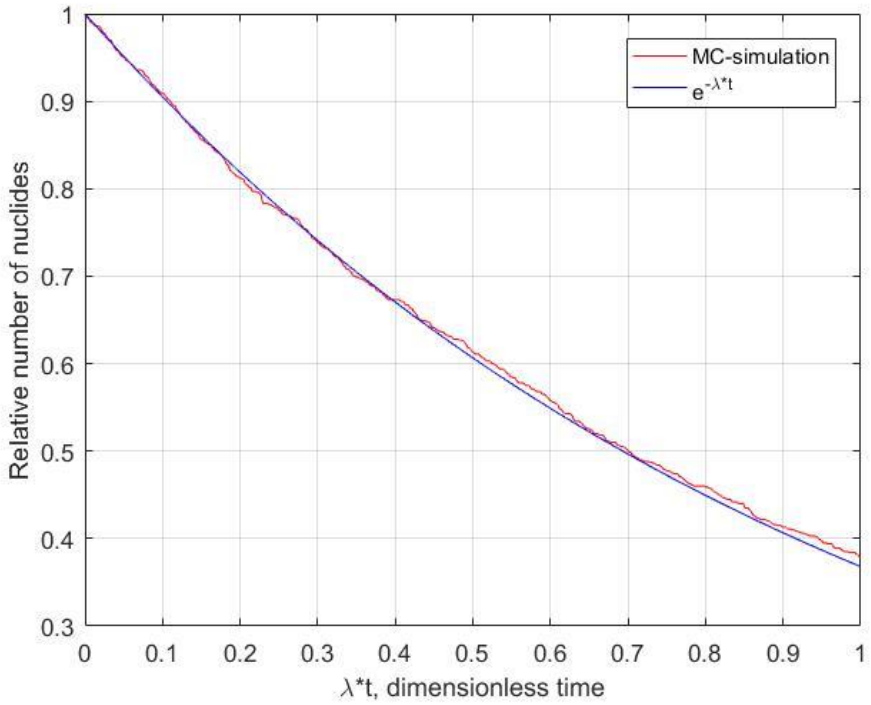


Fig. ME12.1 Results of 10^5 MC simulations for starting 1000 nuclides at λ=1. At larger number of starting nuclides the difference between the stochastic solution and the analytical one is almost undistinguishable.

MATLAB Exercise 12.2 Radioactive chain decay.

In this example one considers the decay chain with *n* differing types of nuclides such that the *i*th nuclide type decays into the *(i + 1)*th nuclide type of chain. If $N_i(t)$ is the number of *i*th nuclide at time *t* and λ_i is its decay constant, one may write the system of decay equations in the form:

$$\frac{dN_1(t)}{dt} = -\lambda_1 \cdot N_1(t)$$

$$\frac{dN_1(t)}{dt} = \lambda_1 \cdot N_1(t) - \lambda_2 \cdot N_2(t)$$

.

.

$$\frac{dN_n(t)}{dt} = -\lambda_n \cdot N_n(t) + \lambda_{n-1} \cdot N_{n-1}(t) \qquad \text{(ME12.2.1).}$$

In the matrix form the system of ordinary differential equation is as follows: $\frac{d\vec{N}(t)}{dt} = \Lambda \cdot \vec{N}$,

where Λ is the triangular bidiagonal matrix

$$\Lambda = \begin{bmatrix} -\lambda_1 & 0 & \dots & \dots & \dots & 0 \\ -\lambda_1 & -\lambda_2 & 0 & \dots & \dots & \vdots \\ \vdots & \lambda_2 & -\lambda_3 & \dots & \dots & \vdots \\ \vdots & 0 & \ddots & \ddots & & \vdots \\ \vdots & 0 & 0 & -\lambda_{n-1} & 0 & \\ 0 & \dots & \dots & \lambda_{n-1} & -\lambda_n & \end{bmatrix} \quad \text{(ME12.2.2).}$$

For the more general case when the decay is with branching, the matrix Λ should be written as follows:

$$\Lambda = \begin{bmatrix} -\lambda_{11} & 0 & & \dots & \dots & \dots & 0 \\ \lambda_{21} & -\lambda_{22} & & 0 & \dots & \dots & \vdots \\ \lambda_{31} & \lambda_{32} & -\lambda_{33} & & \dots & \dots & \vdots \\ \vdots & \vdots & \ddots & & \ddots & & \vdots \\ \vdots & \vdots & \vdots & -\lambda_{n-1,n-1} & & 0 & \\ \lambda_{n1} & \dots & \dots & \lambda_{n,n-1} & & -\lambda_{nn} & \end{bmatrix} \quad \text{(ME12.2.3).}$$

The branching means that from the decay of one parent nuclide several daughter nuclides may be produced. For example, the number of disintegrated nuclides N_1 over unit time λ_{11} may produce λ_{21} of the daughter nuclide N_2 , λ_{31} of the daughter nuclide N_3 , λ_{n1} of the daughter nuclide N_n, i.e. for the element of Λ matrix one can write *n* identities as follow:

$\sum_{j=i+1}^{n} \lambda_{ji} + \lambda_{ii} = 0$, which means the balance between the disintegrated number of the parent isotope and the produced number of daughter isotopes. The system of differential equations (12.2.1) may very stiff because of the wide range of λ_{ij} constants in Λ matrix. In MATLAB program below the system composed of 3 isotopes is considered.

```
function rad_decay

clc;clear;    %Radioactive decay

n=3; %Number of isotopes

y0=zeros(n,1); y0(1)=10^6;  %Intial number of  parent isotope
nuclides

t = [0 25]; %Time of integration

[T,Y] = ode23s(@(t,y) odefcn(t,y,n),t,y0); % integrates the system
of stiff differential equations

%y'=f(t,y) from t0 to tf with initial conditions y0.

figure
```

```
plot(T,(Y(:,1)),'.',T,(Y(:,2)),'--', T,(Y(:,3)),'+');
xlabel('Time'); ylabel('Number of nuclides');

    hold on;grid on;

      legend('Isotope N_1','Isotope N_2','Isotope N_3')         end

function dxdt = odefcn(t,x,n)

dxdt = zeros(n,1);

 lamda = [-1 0 0;0.5 -0.1 0;0.5 0.1 0]; % Matrix of isotope decay
constants

for i=1:n

s=0;

for j=1:i

s = s+lamda(i,j)*x(j); %Isotope decay+gain mother isotope decay

end

dxdt(i)=s;

end

end
```

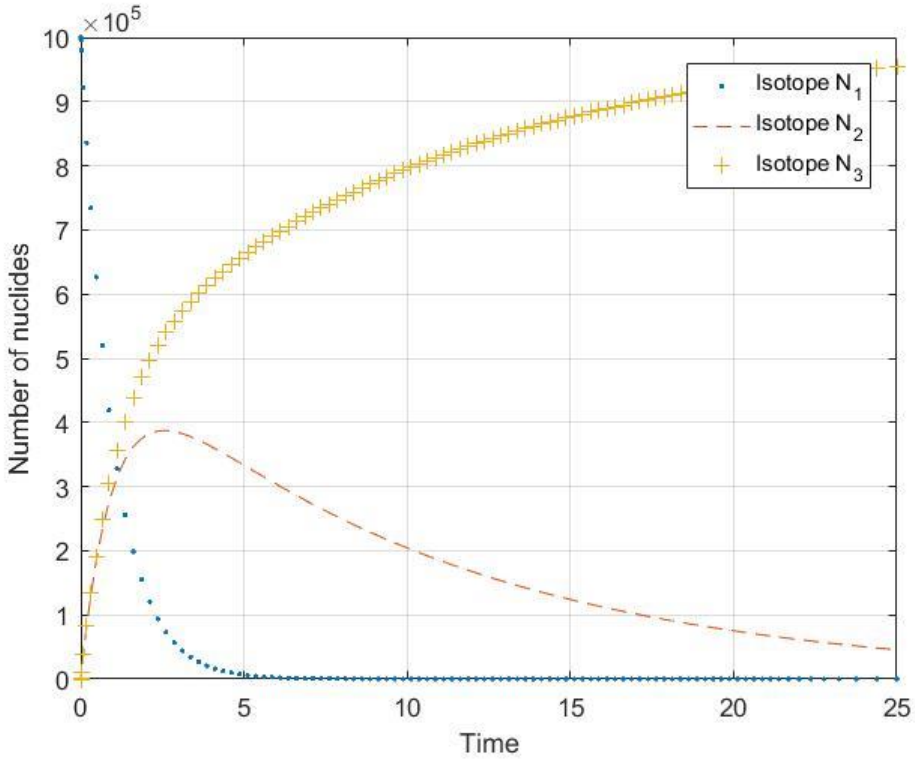


Fig. ME12.2 Branching decay of parent isotope N_1 into unstable N_2 and stable N_3.

Literature:

Birch, F. (1961). Velocity of compressional waves in rocks to 10 kilobars, Part 2. *J. Geophys. Res.* **66**, 2199-2224.

Brantut, N.&E. C. David (2019). Influence of fluids on V_P/V_S ratio: increase or decrease? *Geophys Journ Int* **216,** 2037–2043.

Carslaw, H. S. & J. C. Jaeger 1959. *Conduction of heat in solids*. Oxford: Clarendon Press.

David, E. C. (2012). *The effect of stress, pore fluid and pore structure on elastic wave velocities in sandstones*, PhD thesis, Imperial College London, London.

David, E. C.&R. W. Zimmerman (2011). Compressibility and shear compliance of spheroidal pores: Exact derivation via the Eshelby tensor, and asymptotic expressions in limiting cases. *Intern Journ of Solids and Structures* **48,** 680–686 https://doi.org/10.1016/j.ijsolstr.2010.11.001

Goodman, R. & G.-h. Shi (1985). Block Theory and its Application to Rock Engineering. Prentice-Hall, 352 pp .

Meng, C., Heltsley, W.&D. D. Pollard (2012). Evaluation of the Eshelby solution for the ellipsoidal inclusion and heterogeneity. *Computers &Geosciences* **4**0, 40–48.

Pollard, D. D. & R. C. Fletcher (2005). *Fundamentals of structural geology.* Cambridge University Press. Exercise Solutions.

Driscoll, T. A. (2009). *Learning MATLAB*, SIAM, Philadelphia.