## Appendix E: Matlab Code

### 1. Data extraction:

```
clc
clear all
%all runs pick 100 points
%T=20C
data1=xlsread('DATA_lab1E.xlsx',1,'A36:E172');
t1=data1(:,1);%time(s)
T1=data1(:,2);%Temp(C)
At1=data1(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A01=xlsread('DATA_lab1E.xlsx',1,'E12');
Ainf1=xlsread('DATA_lab1E.xlsx',1,'E381');
%T=23C
data2=xlsread('DATA_lab1E.xlsx',1,'A410:E503');
t2=data2(:,1);%time(s)
T2=data2(:,2);%Temp(C)
At2=data2(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A02=xlsread('DATA_lab1E.xlsx',1,'E393');
Ainf2=xlsread('DATA_lab1E.xlsx',1,'E772');
%T=24.5C
data3=xlsread('DATA_lab1E.xlsx',2,'A22:E272');
t3=data3(:,1);%time(s)
T3=data3(:,2);%Temp(C)
At3=data3(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A03=xlsread('DATA_lab1E.xlsx',2,'E12');
Ainf3=xlsread('DATA_lab1E.xlsx',2,'E378');
%T=26C
data4=xlsread('DATA_lab1E.xlsx',2,'A400:E500');
t4=data4(:,1);%time(s)
T4=data4(:,2);%Temp(C)
At4=data4(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A04=xlsread('DATA_lab1E.xlsx',2,'E390');
Ainf4=xlsread('DATA_lab1E.xlsx',2,'E802');
%T=27.5C
data5=xlsread('DATA_lab1E.xlsx',3,'A25:E215');
t5=data5(:,1);%time(s)
T5=data5(:,2);%Temp(C)
At5=data5(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A05=xlsread('DATA_lab1E.xlsx',3,'E12');
Ainf5=xlsread('DATA_lab1E.xlsx',3,'E381');
%T=29C
data6=xlsread('DATA_lab1E.xlsx',3,'A396:E596');
t6=data6(:,1);%time(s)
```

```
T6=data6(:,2);%Temp(C)
At6=data6(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A06=xlsread('DATA_lab1E.xlsx',3,'E393');
Ainf6=xlsread('DATA_lab1E.xlsx',3,'E773');
%T=30.5C
data7=xlsread('DATA_lab1E.xlsx',6,'A15:E115');
t7=data7(:,1);%time(s)
T7=data7(:,2);%Temp(C)
At7=data7(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A07=xlsread('DATA_lab1E.xlsx',6,'E12');
Ainf7=xlsread('DATA_lab1E.xlsx',6,'E302');
%T=32C
data8=xlsread('DATA_lab1E.xlsx',6,'A316:E416');
t8=data8(:,1);%time(s)
T8=data8(:,2);%Temp(C)
At8=data8(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A08=xlsread('DATA_lab1E.xlsx',6,'E314');
Ainf8=xlsread('DATA_lab1E.xlsx',6,'E613');
%T=33.5C
data9=xlsread('DATA_lab1E.xlsx',4,'A22:E115');
t9=data9(:,1);%time(s)
T9=data9(:,2);%Temp(C)
At9=data9(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A09=xlsread('DATA_lab1E.xlsx',4,'E12');
Ainf9=xlsread('DATA_lab1E.xlsx',4,'E232');
%T=35C
data10=xlsread('DATA_lab1E.xlsx',4,'A253:E347');
t10=data10(:,1);%time(s)
T10=data10(:,2);%Temp(C)
At10=data10(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A010=xlsread('DATA_lab1E.xlsx',4,'E244');
Ainf10=xlsread('DATA_lab1E.xlsx',4,'E429');
%T=36.5C
data11=xlsread('DATA_lab1E.xlsx',5,'A31:E117');
t11=data11(:,1);%time(s)
T11=data11(:,2);%Temp(C)
At11=data11(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A011=xlsread('DATA_lab1E.xlsx',5,'E12');
Ainf11=xlsread('DATA_lab1E.xlsx',5,'E225');
%T=38C
data12=xlsread('DATA_lab1E.xlsx',5,'A250:E342');
t12=data12(:,1);%time(s)
T12=data12(:,2);%Temp(C)
```

```
At12=data12(:,5);%Conductivity(ms/cm)
%initial condition and S.S. consition
A012=xlsread('DATA_lab1E.xlsx',5,'E237');
Ainf12=xlsread('DATA_lab1E.xlsx',5,'E468');
%save extracted data to separate matlab data file .mat
save('lab1E_extracted_data','A01','A02','A03','A04','A05','A06','A07','A08','
A09','A010','A011','A012','Ainf1','Ainf2','Ainf3','Ainf4','Ainf5','Ainf6','Ai
nf7','Ainf8','Ainf9','Ainf10','Ainf11','Ainf12','t1','t2','t3','t4','t5','t6'
,'t7','t8','t9','t10','t11','t12','T1','T2','T3','T4','T5','T6','T7','T8','T9
','T10','T11','T12','At1','At2','At3','At4','At5','At6','At7','At8','At9','At
10','At11','At12')
```

2. Data analysis:

```
clc
clear all
load lab1E_extracted_data.mat

%% matlab plot marker type
marker=['o','+','*','.','x','s','d','^','v','>','<','h']
for i=1:12
    %% A0-- Initial condition
    %clear 's' from previous group(i)'s run
    s=0;
    s=eval(['At' num2str(i) '']);
    A0(i)=eval(['A0' num2str(i) '']);
    %until this step, I extracted every initial At, i.e. A0 at different run
and name the corresponding variable: A01,A02,A03,...,A012
    %% CA0
    %convert to initial concentration CA0 at different T
    %clear 'sT' from previous group(i)'s run
    sT=0;
    sT=eval(['T' num2str(i) '']);
    eval(['CA0' num2str(i) '=CA0(A0' num2str(i) ',sT(1))']);
    % save S.S. T of each run for future use( 'Activation Energy
    % Calculation' section)
    Tss(i)=sT(end);
    %% Ainf-- final S.S. value of At
    Ainf(i)=eval(['Ainf' num2str(i) '']);
    %until this step,
    %I extracted every final S.S. at, i.e. Ainf at different run
    %and name the corresponding variable: Ainf1,Ainf2,Ainf3,...,Ainf12

    %% CCinf (=CB0)
    %Now convert Ainf to S.S. concentration CCinf which is =CB0 at different
T
    eval(['CCinf' num2str(i) '=CCinf_or_CB0(A0' num2str(i) ',Ainf' num2str(i)
',sT(end))']);
    eval(['CB0' num2str(i) '=CCinf' num2str(i) '']);
    % assign CB0 from run i to one vector for future use( see Integral
method--> linear regression--> compare CB0 and CB0_pred)
    CB0(i,:)=eval(['CB0' num2str(i) '']);



    %% CA and x(conversion of A)
    %Calculate CA at any time at each group( @diff T)

    %need to span A0 and Ainf of each group to the same size as the vector
    %size of At in each group
    n=length(s);
    eval(['A0vec' num2str(i) '=repmat(A0(i),n,1)']);
    eval(['Ainfvec' num2str(i) '=repmat(Ainf(i),n,1)']);

    %need to clear 'ca' and 'xa' from previous groups(i)'s run
    ca=0;
    xa=0;
    for j=1:n
        %CA
        ca(j,:)=eval(['CA(CA0' num2str(i) ',CCinf' num2str(i) ',A0'
num2str(i) ',s(j),Ainf' num2str(i) ')']);
        %x (conversion of A corresponding to CA)
        xa(j,:)=eval(['xA(ca(j,:),CA0' num2str(i) ')']);
    end
    %put CA from each run to a vector, 12 runs=12 vectors
    eval(['CA' num2str(i) '=ca']);
    %put xA from each run to a vector, 12 runs=12 vectors
    eval(['xA' num2str(i) '=xa']);
    %% CBt( last point of truncated CB)
    eval(['CBt' num2str(i) '=CB0' num2str(i) '*(1-xa(end))']);
    CBt(i,:)=eval(['CBt' num2str(i) '']);
    %% CA vs t
    figure(1)
    subplot(6,2,i)
    scatter(eval(['t' num2str(i) '']),eval(['CA' num2str(i) '']),1); % set 1
as the size(area) of the marker in scatter plot
    title([' @ T ' num2str(i) ''])
    %% xA vs t
    figure(2)
    subplot(6,2,i)
    scatter(eval(['t' num2str(i) '']),eval(['xA' num2str(i) '']),1);% set 1
as the size(area) of the marker in scatter plot
    title([' @ T ' num2str(i) ''])


    %% Differential method(Y_axis vs X_axis)
    % pick delta t=25 s, means pick every 5 points from original data. (
original delta t=5s)
```

```
    %% dx/dt(i.e. 'xdot')
    % clear 'xdot' 'xvec' 'X_axis' from previous run
    xdot=0;
    xvec=0;
    xvecd=0;
    X_axis=0;
    % call each runs vector xi to a new vector, use in following function
    xvec=eval(['xA' num2str(i) '']);
    % pick delta t=20 s, means pick every 4 points from original data. xvecd
pick every 4 pts from xvec for differential method use( original delta t=5s)
    xvecd=xvec(1:4:length(xvec));
    for j=4:length(xvecd)-2
        %using 4th order(using xvec(t-2),xvec(t-1),xvec(t+1),xvec(t+2))
        %approximation of dx/dt, also because 1st and 2nd pt has time
        %interval not equal to 5s, so ignore first point as x(t-2)
        %so start from j=4, end with j=n-2
        xdot(j-3,:)=xderivative(xvecd(j-2),xvecd(j-1),xvecd(j+1),xvecd(j+2));
        % define X_axis=CA0*(1-x)*(CB0/CA0-x)
        X_axis(j-3,:)=eval(['CA0' num2str(i) ''])*(1-xvecd(j))*(eval(['CB0'
num2str(i) ''])/eval(['CA0' num2str(i) ''])-xvecd(j));
    end
    %put xAdot from each run to a vector and save in workspace, 12 runs=12
vectors
    eval(['xAdot' num2str(i) '=xdot']);
    eval(['Y_axisd' num2str(i) '=xdot']);
    %put X_axis from each run to a vector and save in workspace, 12 runs=12
vectors
    eval(['X_axisd' num2str(i) '=X_axis']);

    %% Linear regression Y_axis(=dx/dt=xAdot=xdot) vs. X_axis(=CA0*(1-
x)*(CB0/CA0-x))
    % function of linear regression below coming from online
    % clear p from previous run
    p=0;
    %using function fitlm(x,y) to linear fit
    mdl = fitlm(X_axis,xdot,'linear','RobustOpts','on');
    %save mdl to workspace for each run
    eval(['mdld' num2str(i) '=mdl']);
    p=mdl.Coefficients.Estimate;%p(1)=intercept, p(2)=slope
    R2=mdl.Rsquared.Ordinary;%R^2
    SE=mdl.Coefficients.SE;%standard error SE(1) for intercept; SE(2) for
slope
    % y predicted from mdl:
    eval(['yd' num2str(i) '=p(1)+p(2)*X_axis']);

    % save vector p from each run to workspace
    eval(['p' num2str(i) '=p']);  %p(1)=intercept, p(2)=slope
    % save rsq from each run to workspace
    eval(['R2' num2str(i) '=R2']);
    % save SE from each run to workspace
    eval(['SE' num2str(i) '=SE']);
    % estimated rate constant k using differential method: kd= slope=p(2)
    kd(i)=p(2);

    %CI of coefficients
    CI=coefCI(mdl,0.05) %row1~CI for p(1)=intercept; row2~CI for p(2)=slope
    % save CI from each run to workspace
    eval(['CI' num2str(i) '=CI']);




%% Intergral method (Y_axisI vs X_axisI)
    %% define Y_axisI=ln((CB0/CA0-x)/(1-x))
    %clear Y_axisI from previous run
    Y_axisI=0;
    for j=1:n
        if (eval(['CB0' num2str(i) ''])/eval(['CA0' num2str(i) ''])-
xvec(j))/(1-xvec(j)) < 0
            break
        else
            Y_axisI(j,:)=log((eval(['CB0' num2str(i) ''])/eval(['CA0'
num2str(i) ''])-xvec(j))/(1-xvec(j)));
        if Y_axisI(j,:) == -inf
            Y_axisI=Y_axisI(1:j-1,:);
            break
        end
        end
    end

    %% linear regression ( X_axisI=t, Y_axisI=ln((CB0/CA0-x)/(1-x)))
    X_axisI=eval(['t' num2str(i) '']);
    %truncate X_axisI to the same size as truncated(in if else above) Y_axisI
    X_axisI=X_axisI(1:length(Y_axisI));

    %put Y_axisI from each run to a vector and save in workspace, 12 runs=12
vectors
    eval(['Y_axisI' num2str(i) '=Y_axisI']);
    %put X_axis from each run to a vector and save in workspace, 12 runs=12
vectors
    eval(['X_axisI' num2str(i) '=X_axisI']);

    % function of linear regression below coming from online
    % clear p from previous run
    pI=0;
    %using function fitlm(x,y) to linear fit
    mdl = fitlm(X_axisI,Y_axisI,'linear','RobustOpts','on');
    % save mdl to workspace
    eval(['mdlI' num2str(i) '=mdl']);
    pI =mdl.Coefficients.Estimate;%p(1)=intercept, p(2)=slope
    R2I=mdl.Rsquared.Ordinary;%R^2
    SEI=mdl.Coefficients.SE;%standard error SE(1) for intercept; SE(2) for
slope
    % y predicted from mdl:
    eval(['yI' num2str(i) '=pI(1)+pI(2)*X_axisI']);

    % save vector pI from each run to workspace
    eval(['pI' num2str(i) '=pI']); %p(1)=intercept, p(2)=slope
    % save rsqI from each run to workspace
    eval(['R2I' num2str(i) '=R2I']);
    % save SEI from each run to workspace
    eval(['SEI' num2str(i) '=SEI']);
    % estimated rate constant k using integral method:
    % slope=pI(2)=k*(CB0-CA0)
    kI(i)=pI(2)/(eval(['CB0' num2str(i) ''])-eval(['CA0' num2str(i) '']));
    % we can also get estimated(predicted) value of CB0 from linear fit and
compare it
    % with experimental value. ( intercept= pI(1)= ln(CB0/CA0))
    CB0_pred(i,:)=exp(pI(1))*eval(['CA0' num2str(i) '']);
    %compare CB0 and CB0_pred by put them into same matrix for easiness of
    %comparison between CB0 and CB0_pred
     compareCB0(i,:)=[CB0(i,:),CB0_pred(i,:)];

     %(1). CI of coefficients
    CII=coefCI(mdl,0.05) %row1~CI for p(1)=intercept; row2~CI for p(2)=slope
    % save CI from each run to workspace
    eval(['CII' num2str(i) '=CII']);


end



%% plot for k
%% Differential method plot
for i=1:12
    figure(3)
    h=0
    h=plot(eval(['mdld' num2str(i) '']));
    eval(['h' num2str(i) '=h']);
    h1(i)=h(1)
    h2(i)=h(2)
    h3(i)=h(3)
    h4(i)=h(4)
    h(3).Visible= 'off';
    h(4).Visible= 'off';
    h(1).Marker = marker(i); %h(1) is the first line, i.e. corresponding to
'experimental'
    h(1).MarkerEdgeColor='k';% set to black
    h(2).LineWidth = 0.3; %h(2) is the second line, i.e. corresponding to
'predicted'
    h(2).Color='k'; %set all predicted line to black color
    Tend=Tss(i);
    legendinfo{i}=['experimental @T=' num2str(Tend) ' degree C'];

hold on
end
l=legend([h1' h2(12)], legendinfo,'predicted'); % incerse h1 to h1' ( 12*1
line to 1*12 line) so that dimension matches legendinfo's dimension!!!!!!
l.Location='southeast';% specify location of the legend: southeast=@bottom
right of plot area
xlabel('CA0*(1-x)*(CB0/CA0-x) [mol/L]');
ylabel('dx/dt [1/s]');
title ' '
     %% format:
% 1. remove legend box
    legend boxoff
% 2. only want to keep tick at left and bottom, remove tick at right and
upper of the box
% get handle to current axes
a = gca;
% set box property to off and remove background color
set(a,'box','off','color','none')
% create new, empty axes with box but without ticks
b = axes('Position',get(a,'Position'),'box','on','xtick',[],'ytick',[]);
% set original axes as active
axes(a)
% link axes in case of zooming
linkaxes([a b])
%3. set tick direction point out of box
set(gca,'TickDir','out')
%4. set background colour to be transparent
set(gcf, 'Color', 'w');
hold off

%% Integral method plot
for i=1:12
    figure(4)
    h=0
    h=plot(eval(['mdlI' num2str(i) '']));
    eval(['hI' num2str(i) '=h']);
    hI1(i)=h(1)
    hI2(i)=h(2)
    hI3(i)=h(3)
    hI4(i)=h(4)
    h(3).Visible= 'off';
    h(4).Visible= 'off';
    h(1).Marker = marker(i); %h(1) is the first line, i.e. corresponding to
'experimental'
    h(1).MarkerEdgeColor='k';% set to black
    h(2).LineWidth = 0.3; %h(2) is the second line, i.e. corresponding to
'predicted'
    h(2).Color='k'; %set all predicted line to black color
    Tend=Tss(i);
    legendinfo{i}=['experimental @T=' num2str(Tend) ' degree C'];
    hold on
end
legend([hI1' hI2(12)], legendinfo,'predicted'); % inverse h1 to h1' ( 12*1
line to 1*12 line) so that dimension matches legendinfo's dimension!!!!!!
xlabel('t [s]');
ylabel('ln((CB0/CA0-x)/(1-x))');
title ' '

     %% format:
% 1. remove legend box
    legend boxoff
% 2. only want to keep tick at left and bottom, remove tick at right and
upper of the box
% get handle to current axes
a = gca;
% set box property to off and remove background color
set(a,'box','off','color','none')
% create new, empty axes with box but without ticks
b = axes('Position',get(a,'Position'),'box','on','xtick',[],'ytick',[]);
% set original axes as active
axes(a)
% link axes in case of zooming
linkaxes([a b])
%3. set tick direction point out of box
set(gca,'TickDir','out')
%4. set background colour to be transparent
set(gcf, 'Color', 'w');
hold off


%% Activation energy( Ea)
    %% Using estiamted k from Differential method( kd)
    % According to Arrhenius Equation:
    for i=1:12
        X_axisAd(i,:)=1/(Tss(i)+273.15);% T should be in [K]!!!!!!
        Y_axisAd(i,:)=log(kd(i));
    end
    %linear regression:
    % function of linear regression below coming from online
    %using function fitlm(x,y) to linear fit
    mdl1 = fitlm(X_axisAd,Y_axisAd,'linear','RobustOpts','on');
    pAd = mdl1.Coefficients.Estimate;%p(1)=intercept, p(2)=slope
    R2Ad =mdl1.Rsquared.Ordinary;%R^2
    SEAd=mdl1.Coefficients.SE;%standard error SE(1) for intercept; SE(2) for
slope

    % Slope =pAd(2)=-Ea/R, intercept=pAd(1)=ln(k0)
    Ead=-pAd(2)*8.314;
    k0d=exp(pAd(1));

    %(1). CI of coefficients
    CIAd=coefCI(mdl1,0.05) %row1~CI for p(1)=intercept; row2~CI for
p(2)=slope

    %(2). CI for individual response(prediction) and plot:
    figure(27)
    h5=plot(mdl1) %this plot includes 95% CI for individual
response(prediction)
    h5(1).Marker = '+'; %h(1) is the first line, i.e. corresponding to
'experimental'
    h5(1).MarkerSize=6;
    h5(1).Color='k';

    h5(2).LineStyle = '-';
    h5(2).LineWidth = 1;
    h5(2).Color = 'k';

    h5(3).Color ='k';
    h5(4).Color ='k';

    hold on


    %% Using estiamted k from Integral method( kI)
    % According to Arrhenius Equation:
    for i=1:12
        X_axisAI(i,:)=1/(Tss(i)+273.15);% T should be in [K]!!!!!!
        Y_axisAI(i,:)=log(kI(i));
    end
    %linear regression:
    % function of linear regression below coming from online
    %using function fitlm(x,y) to linear fit
    mdl = fitlm(X_axisAI,Y_axisAI,'linear','RobustOpts','on');
    pAI = mdl.Coefficients.Estimate;%p(1)=intercept, p(2)=slope
    R2AI=mdl.Rsquared.Ordinary;%R^2
    SEAI=mdl.Coefficients.SE;%standard error SE(1) for intercept; SE(2) for
slope
```

```
    % Slope =pAI(2)=-Ea/R, intercept=pAI(1)=ln(k0)
    EaI=-pAI(2)*8.314;
    k0I=exp(pAI(1));

    %(1). CI of coefficients
    CIAI=coefCI(mdl,0.05); %row1~CI for p(1)=intercept; row2~CI for
p(2)=slope

    %(2). CI for individual response(prediction) and plot:
    h6=plot(mdl) %this plot includes 95% CI for individual
response(prediction)
    legend([h5(1) h5(2) h5(3) h6(1) h6(2) h6(3)],'experimental from
differential method','predicted from differential method','95% CI from
differential method','experimental from integral method','predicted from
integral method','95% CI from integral method')
    title(' ')
    xlabel('1/T [1/K]')
    ylabel('ln(k)')
    h6(1).Marker = '*'; %h(1) is the first line, i.e. corresponding to
'experimental'
    h6(1).MarkerSize=6;
    h6(1).Color='k';

    h6(2).LineStyle = '-.';
    h6(2).LineWidth = 1;
    h6(2).Color = 'k';

    h6(3).LineStyle='--';% h(3)= lower CI bound;
    h6(4).LineStyle='--';%h(4)= higher CI bound;
    h6(3).Color ='k';
    h6(4).Color ='k';

    %% format:
% 1. remove legend box
    legend boxoff
% 2. only want to keep tick at left and bottom, remove tick at right and
upper of the box
% get handle to current axes
a = gca;
% set box property to off and remove background color
set(a,'box','off','color','none')
% create new, empty axes with box but without ticks
b = axes('Position',get(a,'Position'),'box','on','xtick',[],'ytick',[]);
% set original axes as active
axes(a)
% link axes in case of zooming
linkaxes([a b])
%3. set tick direction point out of box
set(gca,'TickDir','out')
```

## 3. Functions used in data analysis:

```
function y=CA(CA0,CCinf,A0,At,Ainf)
y=CA0-CCinf*((A0-At)/(A0-Ainf))
end

function CA0= CA0(A0,T)
CA0=A0/(192.7*(1+0.01667*(T-20)))
end

function y=CCinf_or_CB0(A0,Ainf,T)
y=(A0-Ainf)/(124.62+1.5892818*(T-20))
end

%Calculate xA( conversion)
function y=f(CA,CA0)
y=1-(CA/CA0)
end

%Calculate dx/dt in differential method
function y=f(xm2,xm1,x1,x2)
y=(xm2-8*xm1+8*x1-x2)/(12*20) %delta_t=20s
end
```

## 4. Export truncated data to Excel:

```
clc
clear all
load lab1E_extracted_data.mat

delete('truncateddata.xlsx');

for i=1:12
    T = table(eval(['t' num2str(i) '']),eval(['T' num2str(i) '']),eval(['At'
num2str(i) '']));
    T.Properties.VariableNames = {['time_t' num2str(i) '__s'],
['temperature_T' num2str(i) '__degreeC'], ['conductivity_At' num2str(i)
'__mS_per_cm']}
filename = 'truncateddata.xlsx';
writetable(T,filename,'Sheet',i,'Range','A2')
end
```

## 5. Plot original and truncated data for Appendix A&B

```
clc
clear all
load lab1E_extracted_data.mat
for i=1:12
```

```
    sT=0;
    sT=eval(['T' num2str(i) '']);
    figure(i)
    h1=scatter(eval(['t' num2str(i) '']),eval(['At' num2str(i) '']),20);
    h1.MarkerEdgeColor='k';% set to black
    %legend(['conductivity @T' num2str(i) '=' num2str(sT(end)) ' degree C'])
    xlabel('t [s]');
    ylabel('conductivity [mS/cm]');
   %% format:
% 1. remove legend box
    legend boxoff
% 2. only want to keep tick at left and bottom, remove tick at right and
upper of the box
% get handle to current axes
a = gca;
% set box property to off and remove background color
set(a,'box','off','color','none')
% create new, empty axes with box but without ticks
b = axes('Position',get(a,'Position'),'box','on','xtick',[],'ytick',[]);
% set original axes as active
axes(a)
% link axes in case of zooming
linkaxes([a b])
%3. set tick direction point out of box
set(gca,'TickDir','out')
%4. set background colour to be transparent
set(gcf, 'Color', 'w');
hold off
end
```