# Created by Oliver Arent Heilmann, 26121093

This code takes 10 datasets from a nanoindentation experiment on an unknown sample. It is worth noting that this code is able to handle any number of samples so, had we been presented with more data, a more robust (and therefore representative) set of values could be calculated.

## Contents


- Setting up Data
- Replacing all Zero Values in Matrix with NaN
- Plotting All Given Data
- Percentage Error of Dataset 8 & 10
- Interpreting the Data
- Calculating Mechanical Properties
- Finding Percentage Errors
- Exporting to an Excel File


## Setting up Data

Data has been printed out as one continuous file. In order to display a graph with several plots it must fist be broken down into 10 distinct sections.

```
clear
num=xlsread('/Users/OliverHeilmann/Documents/SESG6034-C2 data.xlsx');
m=1; n=1;
fixed_table=[];
for i=1:length(num)
    if isnan(num(i,1))==0
        fixed_table(m,n:(n+1))=num(i,1:(end-1));
        m=m+1;
    elseif isnan(num(i:i+1,1))==1
        m=1;
        n=n+2;
    end
end
```

## Replacing all Zero Values in Matrix with NaN

As each indent has a different number of samples, the overall fixed_table matrix contains cell elements with zeros. If an average plot was to be constructed now, the zeros would skew the later stages of the data. For this reason they must be replaced with NaN instead.

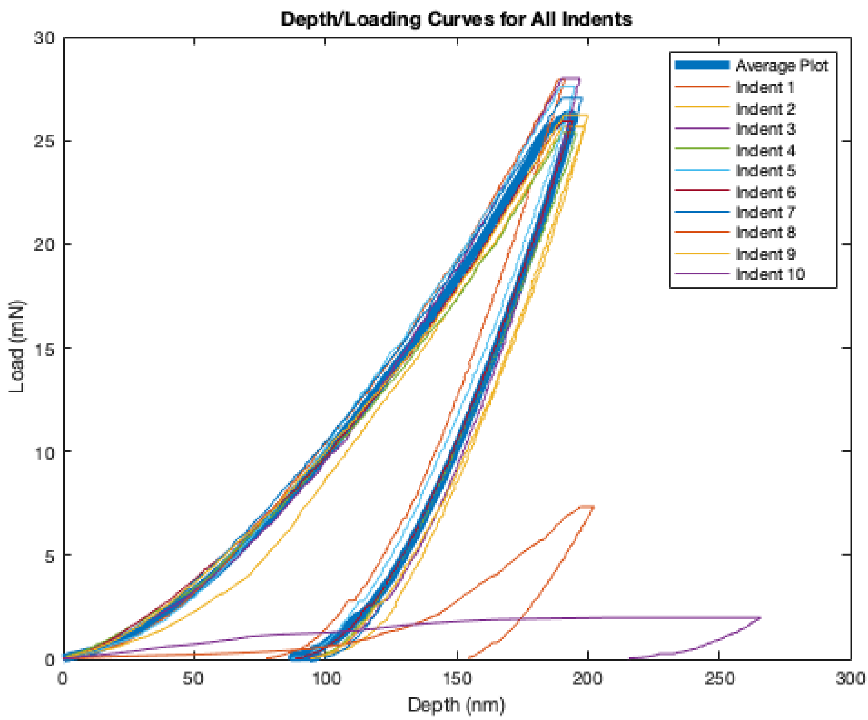
```
dim=size(fixed_table);
for i=1:dim(1)
    for j=1:dim(2)
        if fixed_table(i,j)==0
            fixed_table(i,j)=NaN;
        end
    end
end
```

## Plotting All Given Data

With the table prepared, we are able to plot the graph. Upon plotting, it is clear to see that two curves are dramatically different from the rest and, therefore, should be excluded from the average plot. This code presents a graph with all individual indent curves while also plotting the average (which excludes the two anomolous curves).

```
figure(1);
fixed_table_edit=fixed_table;           % Sorting Data for Average Plot
fixed_table_edit(:,[15,16,19,20])=[];   % Deleting anomolous plots
%fixed_table_edit=fixed_table(:,[15,16,19,20]);   % Only anomolous plots
allDepth=fixed_table(:,1:2:end-1);                  % Odd matrix
allLoad=fixed_table(:,2:2:end);                     % Even matrix
```

```
allDepth_select=fixed_table_edit(:,1:2:end-1);  % Average Odd matrix
allLoad_select=fixed_table_edit(:,2:2:end);      % Average Even matrix
averageDepth=nanmean(allDepth_select,2);  % Taking Average Values of Depth
averageLoad=nanmean(allLoad_select,2);    % Taking Average Values of Load
list={'Average Plot'};
for i=1:(dim(2)/2)                         % dim(2)/2 = 'number of indents'
    x=['Indent ',(num2str(i))];
    list{end+1}=x;
end
plot(averageDepth,averageLoad,'LineWidth',5);        % Plots average curve
hold on
plot(allDepth,allLoad);                              % Plots all indents
title('Depth/Loading Curves for All Indents')        % Table Formatting
xlabel('Depth (nm)')
ylabel('Load (mN)')
axis([0 300 0 30]);
legend(list);
```




## Percentage Error of Dataset 8 & 10

Earlier, we qualitatively determined that datasets 8 & 10 should be excluded. By comparing areas under the curve, one is able to quantitatively determine how much these plots were off the average Depth/Load curve areas. While this method is faily crude, it is suitable enough for this instance.

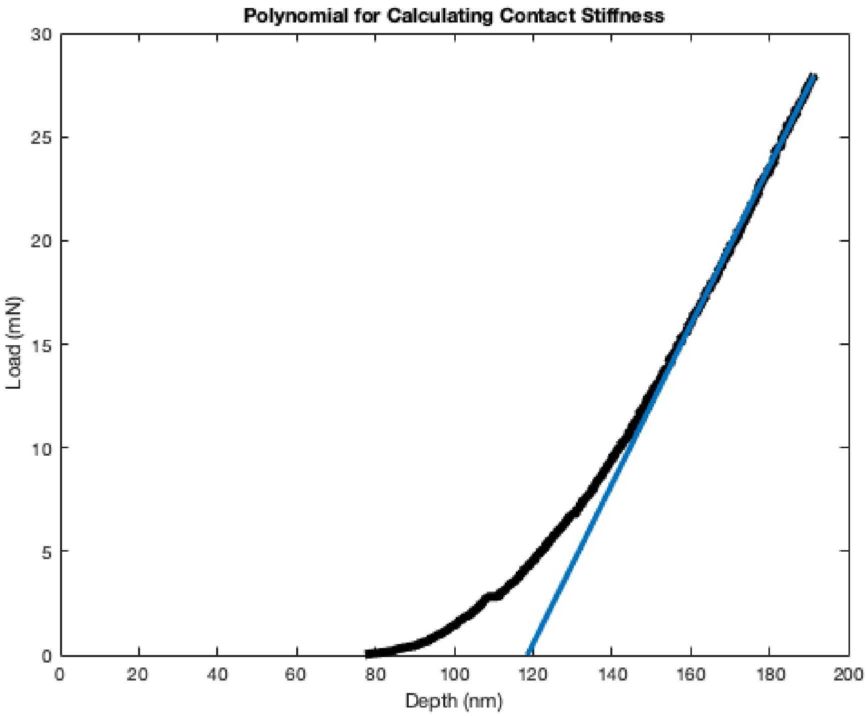
```
AreaAv=trapz(averageDepth,averageLoad);                      % Area of Av curve
Area8=trapz(fixed_table(1:249,15),fixed_table(1:249,16)); % Area of 8 curve
Area10=trapz(fixed_table(1:69,19),fixed_table(1:69,20));  % Area of 10 curve
percent_error8=100-((100/AreaAv)*Area8);
percent_error10=100-((100/AreaAv)*Area10);
```

## Interpreting the Data

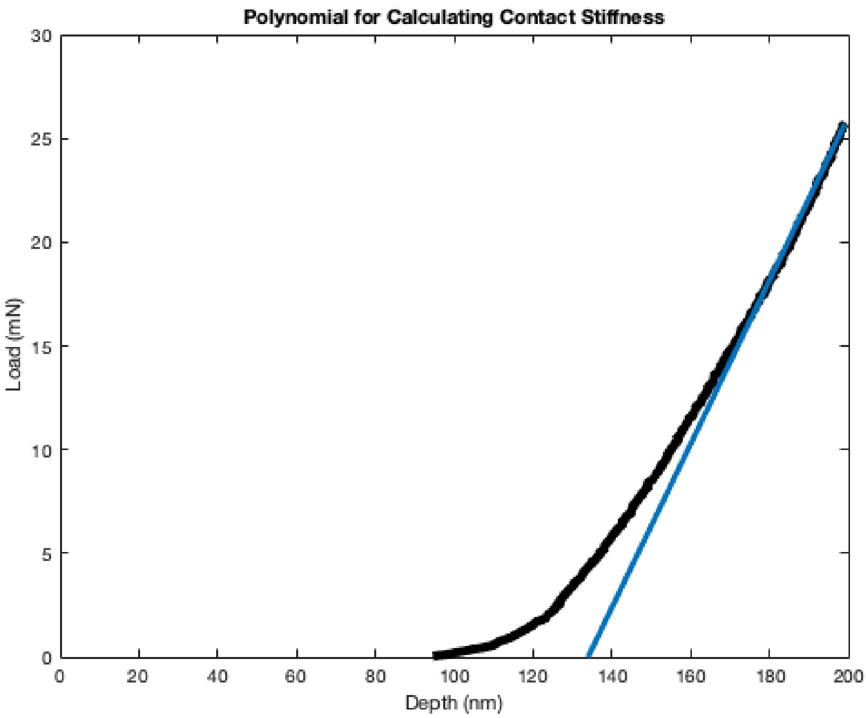
Finding the Contact Stiffness 'S' relies upon the gradient of the unloading curve. For this reason it is important to find a line of best fit which runs through the max amount of data-points possible. This portion of the code itterates through every possible line of best fit for the unloading dataset until it finds the one which intersects the most points (i.e. where the straightest portion of the unloading curve is). A for loop is used to iterate through every individual loading condition (for which the mechanical properties are calculated and stored is lists). At the end the properties are averaged and presented as 'bulk' properties.

```
Elastic_Modulus=[];
Hardness=[];
for i=1:length(allDepth_select(1,:))
```

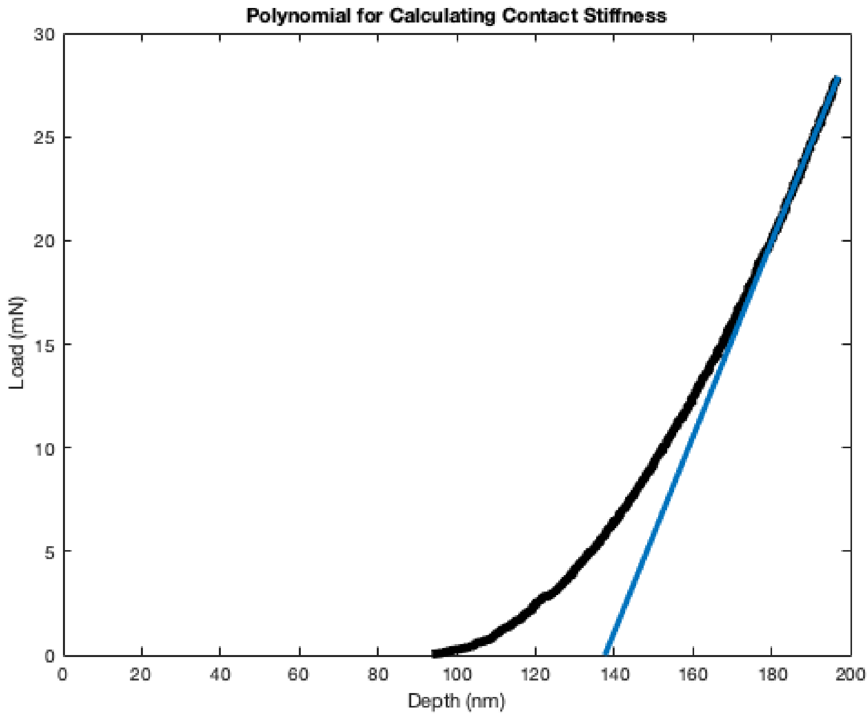
```
    count=1;
    polyOrder=1;                    % Define order of polynomial fit
    [NaNrow,NaNcol] = find(isnan(allDepth_select(:,i)));
    if isempty(NaNrow)
        testX_Depth=allDepth_select(:,i);
        testX_Load=allLoad_select(:,i);
    else
        testX_Depth=allDepth_select(1:(NaNrow-1),i);
        testX_Load=allLoad_select(1:(NaNrow-1),i);
    end
    [M,I]=max(testX_Depth); % Max value and index should be the same for both load and depth
    for j=(I+11):length(testX_Depth) % +11 for ignoring noise near Pmax
        UnloadingDepth=testX_Depth(I:end);
        UnloadingLoad=testX_Load(I:end);
        linearDepth=testX_Depth(I:j);
        linearLoad=testX_Load(I:j);
        p=polyfit(linearDepth,linearLoad,polyOrder);
        y1=polyval(p,UnloadingDepth);
        xintercept=interp1(y1,UnloadingDepth,0);
        xintercept_list(count,1)=xintercept;
%         figure(2)
%         plot(UnloadingDepth,UnloadingLoad,'o')
%         hold on
%         plot(UnloadingDepth,y1,'LineWidth',3)
%         hold off
%         title('Polynomial for Calculating Contact Stiffness')       % Table Formatting
%         xlabel('Depth (nm)')
%         ylabel('Load (mN)')
%         axis([0 200 -10 30]);
        count=count+1;
    end
    seeklinear=abs(diff(xintercept_list));  % Represents the differentiated unloading curve
    acceptDiff=0.01;                       % Indicate precision of line of best fit
    ii=1;                                   % Counter used for indexing later
    while seeklinear(ii)>=acceptDiff        % Finds the smallest difference
    %   type = sprintf('Nope! %f is larger than %f',seeklinear(ii),acceptDiff);
    %   disp(type)
        ii=ii+1;
    end
    bestfit=I+ii+1; % bestfit=StartingIndex+WhileLoopCounter+1(counting stops @seeklinear(ii)<=acceptDiff)
    p=polyfit(testX_Depth(I:bestfit),testX_Load(I:bestfit),polyOrder);
    y1=polyval(p,UnloadingDepth);
    figure(3)
    plot(UnloadingDepth,UnloadingLoad,'k','LineWidth',5)
    hold on
    plot(UnloadingDepth,y1,'LineWidth',3)
    hold off
    title('Polynomial for Calculating Contact Stiffness')      % Table Formatting
    xlabel('Depth (nm)')
    ylabel('Load (mN)')
    axis([0 200 0 30]);
```

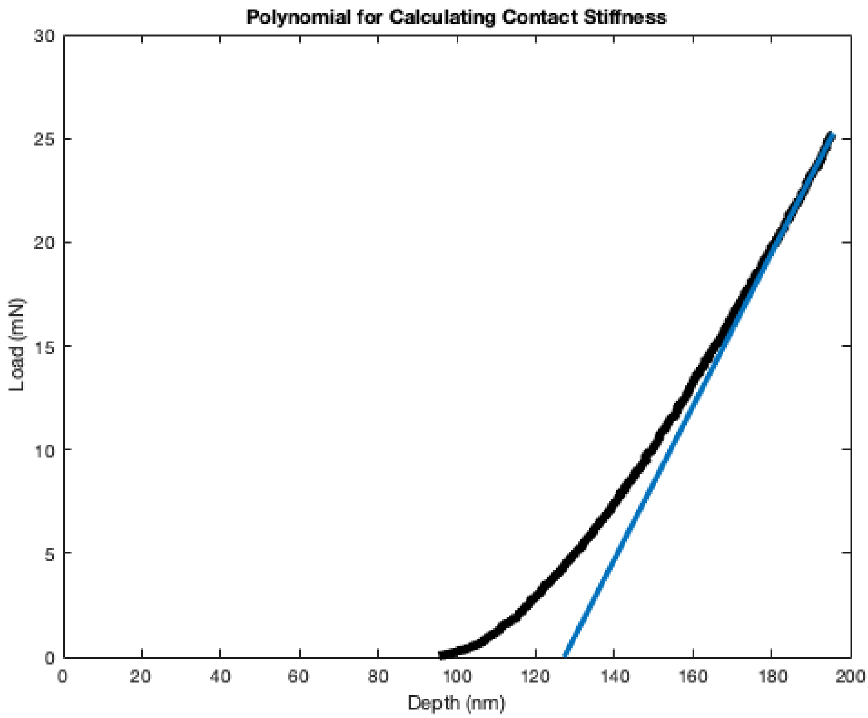


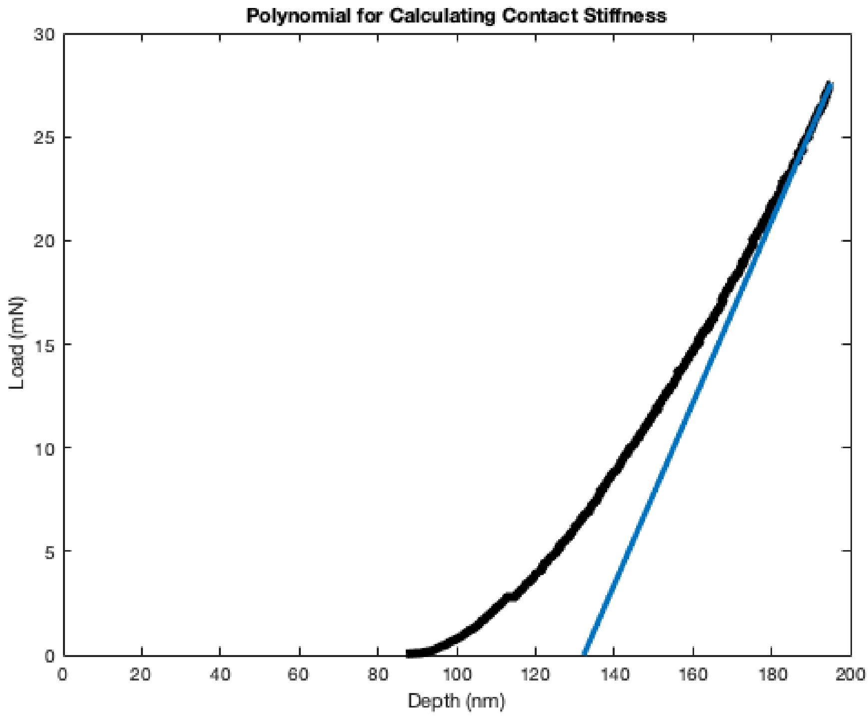
Polynomial for Calculating Contact Stiffness
Load (mN)
Depth (nm)
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140
160
180
200

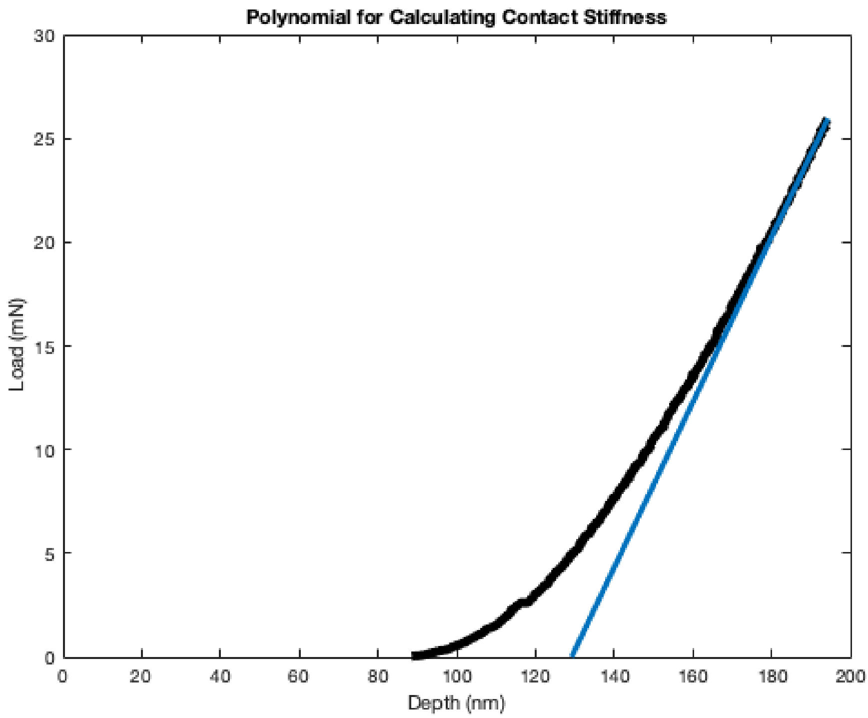



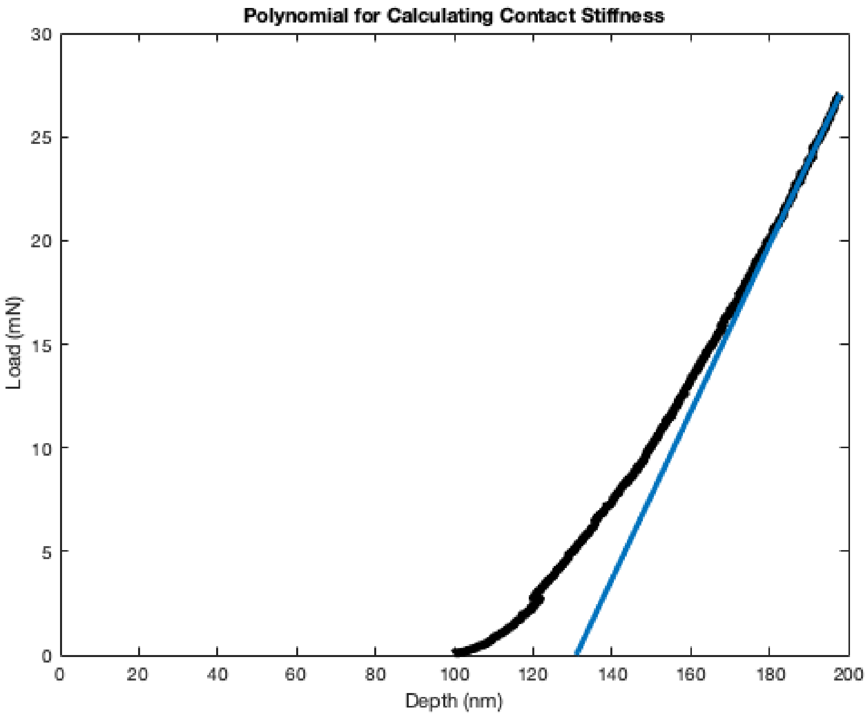
Polynomial for Calculating Contact Stiffness
Load (mN)
Depth (nm)
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140
160
180
200

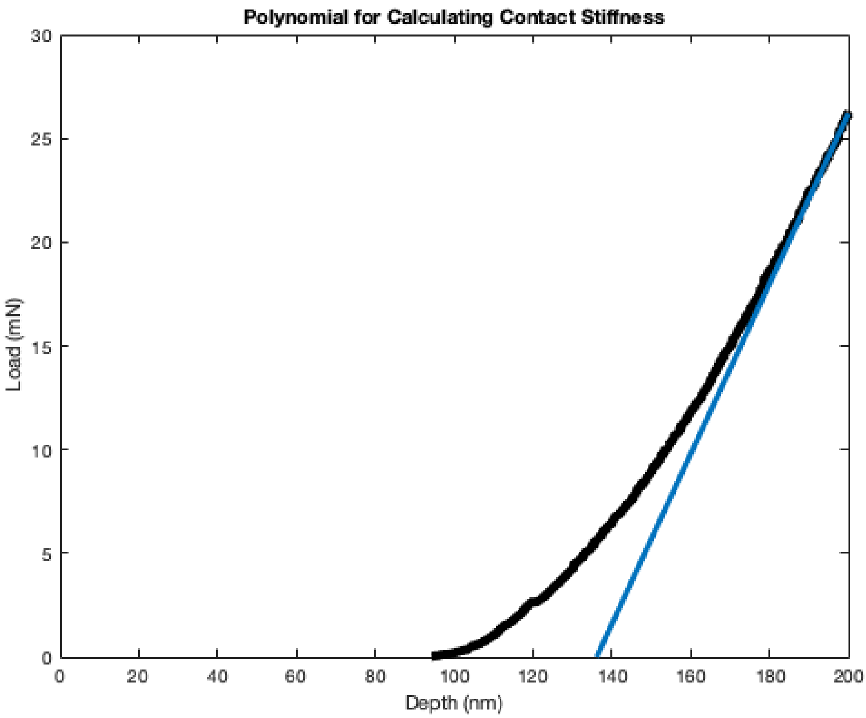

Polynomial for Calculating Contact Stiffness
Load (mN)
Depth (nm)
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140
160
180
200


Polynomial for Calculating Contact Stiffness
Load (mN)
Depth (nm)
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140
160
180
200


Polynomial for Calculating Contact Stiffness
Load (mN)
Depth (nm)
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140
160
180
200


Polynomial for Calculating Contact Stiffness
Load (mN)
Depth (nm)
0
5
10
15
20
25
30
0
20
40
60
80
100
120
140
160
180
200




## Calculating Mechanical Properties

Constant values are presented first with calculated values presented later.

```
theta=65.3;                     % For Berkovitch Indentor Tip (degrees)
v=0.3;                          % Poissons Ratio material
vi=0.0691;                      % Poissons Ratio indentor
Ei=1143*10^9;                   % Elastic Modulus indentor
beta=1.034;                     % Correction Factor for shape of Berkovitch indentor
epsilon=0.74;                   % Value can be calculated from 'm' (Oliver-Pharr)
Pmax=max(testX_Load)*(10^-3);   % Maximum Load
ht=max(testX_Depth)*10^-9;      % Maximum Depth
xintercept_opt=xintercept_list(ii)*10^-9; % Find x intercept for polynomial
dP=max(testX_Load)*(10^-3)-0;
dh=ht-(xintercept_opt);
S=dP/dh;                        % Contact stiffness
```

```
    hs=epsilon*(Pmax/S);
    hp=ht-hs;
    Ac=3*sqrt(3)*(hp^2)*tand(theta)^2;
    Er=(sqrt(pi)/(2*beta))*(S/sqrt(Ac));
    YM=(1-v^2)/((1/Er)-((1-vi^2)/Ei));
    H=Pmax/Ac;
    Elastic_Modulus(1,i)=YM;         %#ok<*SAGROW>
    Hardness(1,i)=H;
```

```
end
Average_Elastic_Modulus=mean(Elastic_Modulus)
Average_Hardness=mean(Hardness)
```

```
Average_Elastic_Modulus =

   7.4665e+11


Average_Hardness =

   5.0127e+10
```

## Finding Percentage Errors

Simply finding standard deviations for all relevant properties and converting them to percentages.

```
Depth_Perror=(100/mean(max(allDepth(:,[1:7,9]))))*std(max(allDepth(:,[1:7,9])))
Load_Perror=(100/mean(max(allLoad(:,[1:7,9]))))*std(max(allLoad(:,[1:7,9])))
Hardness_Perror=(100/Average_Hardness)*std(Hardness)
Elastic_Modulus_Perror=(100/Average_Elastic_Modulus)*std(Elastic_Modulus)
HE_ratio=Average_Hardness/Average_Elastic_Modulus
```

```
Depth_Perror =

    1.4446


Load_Perror =

    3.9480


Hardness_Perror =

    9.8460


Elastic_Modulus_Perror =

    9.6646


HE_ratio =

    0.0671
```

## Exporting to an Excel File

```
T = [max(averageDepth),Depth_Perror,...
     max(averageLoad),Load_Perror,...
     (Average_Hardness*10^-9),Hardness_Perror,...
     (Average_Elastic_Modulus*10^-9),Elastic_Modulus_Perror,HE_ratio];
```

```
Table=array2table(T,'VariableNames',{'Maximum_Penetration_Depth_nm',...
    'P_error1','Maximum_Load_mN','P_error2','Hardness_GPa',...
    'P_error3','Elastic_Modulus_GPa','P_error4','H_E_Ratio'});
filename = 'SESG6034_CW_Data.xlsx';
writetable(Table,filename,'Sheet',1,'Range','D1')

% Could be the following material:
% TiAlCrN (Hardness_max=53 GPa, elastic_modulus_max=887 GPa)
```

*Published with MATLAB® R2018a*