# Second Progress Report for

# Experimental Study of Crackling Noise as Micro-mechanics of Flow

Part of the LIGO SURF programme 2015

Kla Karava
Mentor: Xiaoyue Ni

## 1. Details of Work Completed

So far I have fabricated 250 nm, 500 nm and 1 μm copper nanopillars, and 500 nm fused silica nanopillars using the focused ion beam (FIB) technique[1,2] by *FEI® Versa 3D*. All of these pillars have an aspect ratio of approximately 3:1. Then these pillars were performed by either uniaxial compression tests or DMA tests (dynamic mechanical analysis tests) on copper and fused silica nanopillars using a commercialised nanoindentation machine *Hysitron® Triboindenter®* equipped with an 8 μm diamond flat tip.

### 1.1 500 nm Copper Nanopillars

There are 13 batches (each with 9 pillars) of 500 nm copper nanopillars. Batches 1 to 5 were used mostly for uniaxial compression tests for finding the Young's modulus and the yield strength of 500 nm copper nanopillars. In addition the first pillar of each of the following batches (that were used for DMA tests) was also used for these purposes. The Young's modulus and the yield strength were retrieved from the MatLab code developed by myself called *ss_con (stress-strain converter)* as mentioned and described in details in the first progress report[1]. The code is also given the appendix of this report for reference. Then the mean values were evaluated to represent the best estimates at such. An example of the stress-strain curve for a 500 nm copper nanopillar is given in figure 1. The values are given in table 1 below:

Table 1: summary of results for compression tests on 500 nm copper nanopillars

| | |
|---|---|
| Young's modulus/GPa | 150 ± 8 |
| Yield strength/MPa | 539 ± 25 |
| Diameter/nm | 442 ± 11 |
| Height/nm | 1580 ± 30 |

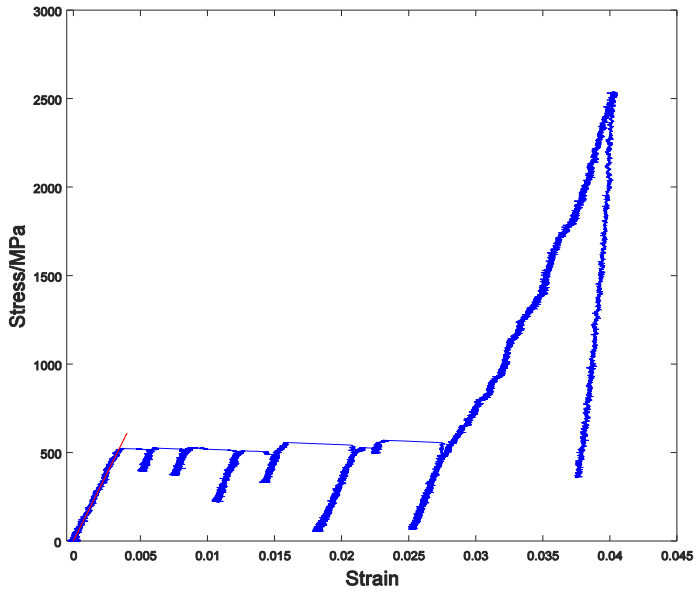



Figure 1: the stress-strain curve for a 500 nm copper nanopillar

The values of Young's modulus and yield strength found in the experiments are in good agreement with the literature values[3]. However it can be seen that the diameters and the heights of the pillars are below the nominal values of 500 nm and 1500 nm respectively. Moreover the pillars are never perfectly cylindrical as shown in figures 2 and 3. All pillars inevitably suffer from *tapering* at different degrees; hence their morphology is more like a lampshade than a cylinder. Tapering is mainly caused by imperfect focusing and poor adjustments of the stigmator. This effect can be eliminated by performing good focus and stigmation of the FIB.

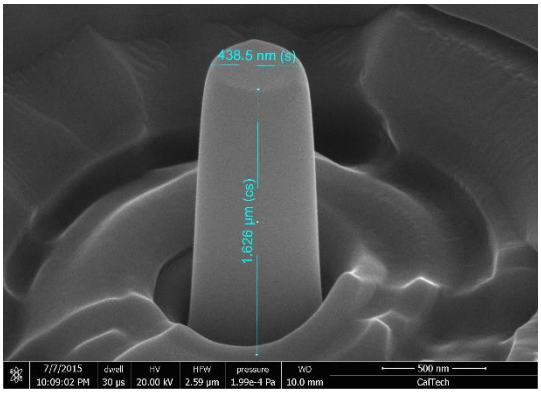



Figure 2: a lightly tapered 500 nm copper nanopillar

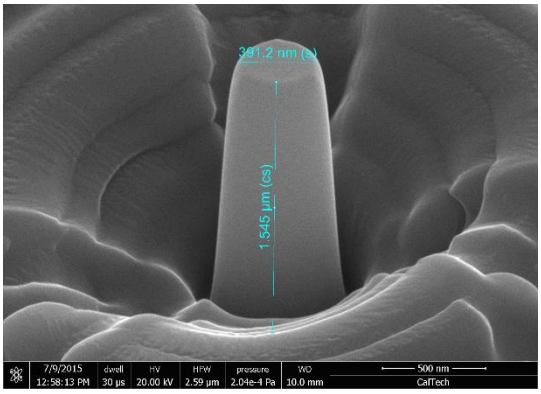



Figure 3: a severly tapered 500 nm copper nanopillar

After successfully obtaining the benchmark for 500 nm copper nanopillars, DMA tests were performed on 500 nm copper batches 6-13. There were 8 pillars available for each batch since the first pillar was always used for a compression test aforementioned. The

reason of doing this was to ensure that the tip was well-aligned and ready for more sensitive DMA tests afterwards. The parameters in a DMA test includes the number of static steps, maximum load, begin amplitude, and amplitude type. The parameters were adjusted for different pillars from different batches as to see if the results were different for different loading conditions or not. However the number of static steps merely determines the resolution of the moduli vs. static stress plots and does not affect the behaviour of the load data. The amplitude of the load should affect the dynamical behaviour of the pillars since the dislocations experience different degrees of stress for a given static stress step. There are two types of load amplitude including linearly variable and constant. The linearly variable amplitude type ensures that the amplitude increases proportionally with static stress whilst the constant amplitude type retains constant amplitude for all the static stress steps. The summary of the parameters are given in table A1 in the appendix.

For most of the pillars, the maximum load was taken to be 140 μN or equivalently stress of approximately 710 MPa for 500 nm pillars. This was to ensure that the yield point was surpassed and the pillars were ultimately plastically deformed. The number of static steps was begun from a low value of 11 as to see the general outline first before being continually increased to 24. However it was found that even at the value as high as 24 steps, the resolution did not improve significantly because almost half of the static steps were in the plastic regime, which we are not interested in for this project. So a totally new DMA load function was designed to achieve a very high resolution of 28 steps mostly within the elastic regime; the maximum load was reduced to 75 μN or stress of approximately 380 MPa for 500 nm pillars that was nearly the value of yield stress. The frequency of the oscillatory load was mostly 1 Hz as to ensure that any change in stress was not too rapid to activate the dislocations in the pillars. And at the end, after obtaining good results for 1 Hz, the frequency was then increased to 10 Hz as to see if there was any rate dependence. Then the DMA test data were analysed to find the behaviour loss and storage moduli according to different values of static stress using a MatLab code *dynamod*. The details for the development of *dynamod* are given in section 2 of this report.

### 1.2 500 nm Fused Silica Nanopillars

In addition to 500 nm copper nanopillars, compression tests and DMA tests were also performed on 500 nm fused silica nanopillars for calibration purpose, as we want to tell the effects due to dislocation dynamics from other anomalies such as machine effects. An example of a 500 nm fused silica nanopillar is showed in figure 4. The primary difference between copper and fused silica is that copper is crystalline whilst fused silica is amorphous which means that dislocations are not present in fused silica. This would give rise to a different behaviour of moduli vs. static stress than copper if our hypothesis of elastic dislocation mechanics were true[1]. Another difference is that fused silica, as being amorphous, is brittle whereas copper is ductile. This means the maximum load cannot

exceed the fracture stress for fused silica, otherwise pillars will fracture. The maximum load was taken to be 100 μN or stress of approximately 510 MPa for 500 nm pillars as to ensure that the loading will not reach the fracture point. The loading curve abd the Young's modulus of the 500 nm fused silica nanopillars are given in figure 5 and table 2 respectively. And the summary of the parameters for DMA tests are given in table A2 in the appendix.

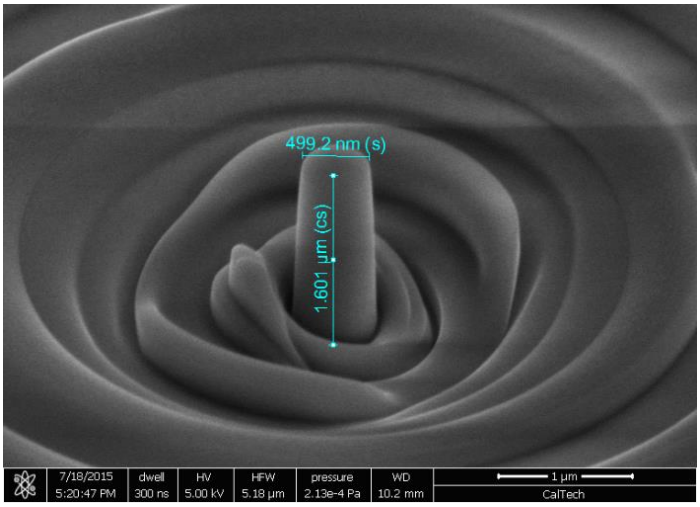



Figure 4: a 500 nm fused silica nanopillar

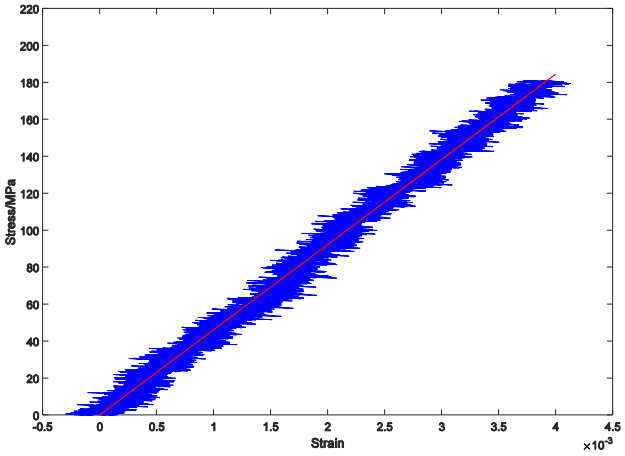



Figure 5: the stress-strain curve for a 500 nm fused silica nanopillar

Table 2: summary of results for compression tests on 500 nm fused silica nanopillars

| | |
|---|---|
| Young's Modulus/GPa | 55.4 ± 0.7 |
| Diameter/nm | 496 ± 3 |
| Height/nm | 1690 ± 80 |

### 1.3 1 μm Copper Nanopillars

According to the project proposal[1], the size effect is also to be studied in this project. We are going to see if there is any difference in the behaviour of DMA tests for different sizes of copper nanopillars. Since nanopillars with larger diameter are easier to fabricate using FIB, the next diameter to be studied first is 1 μm. An example of a 1 μm copper nanopillar is shown in figure 6.

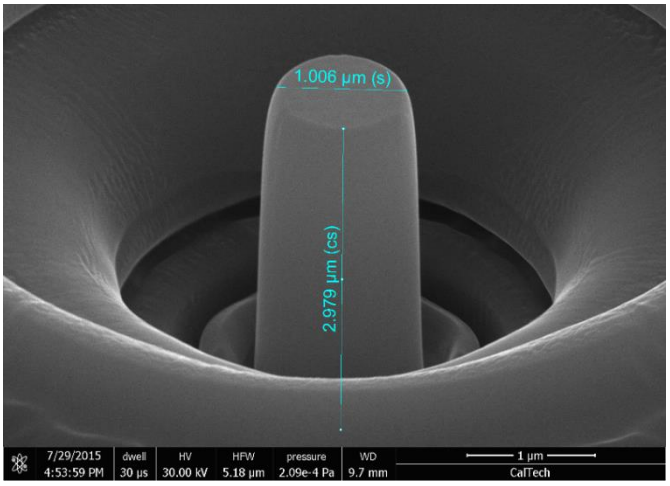



Figure 6: a 1 μm copper nanopillar

However the parameters for DMA tests are not identical to those of 500 nm nanopillars' because different dimensions give rise to different *stiffness* or generally known as *spring constant.* To achieve the same displacement amplitude, different load amplitude must be applied at a specific ratio of 3.23:1; or in other words, the load amplitude for 1 μm nanopillars must be 3.23 times of that for 500 nm nanopillars. The reason for retaining the same displacement amplitude for all pillar sizes is that we want to have the same dislocation dynamical behaviours. An example of the stress-strain curve for a 1 μm copper nanopillars is given in figure 7. The summary of the compression tests and the DMA parameters are given in tables 3 and A3 respectively.

Table 3: summary of results for compression tests on 1 μm copper nanopillars

| | |
|---|---|
| Young's modulus/GPa | 101 ± 5 |
| Yield strength/MPa | 339 ± 15 |
| Diameter/nm | 1024 ± 6 |
| Height/nm | 2093 ± 14 |

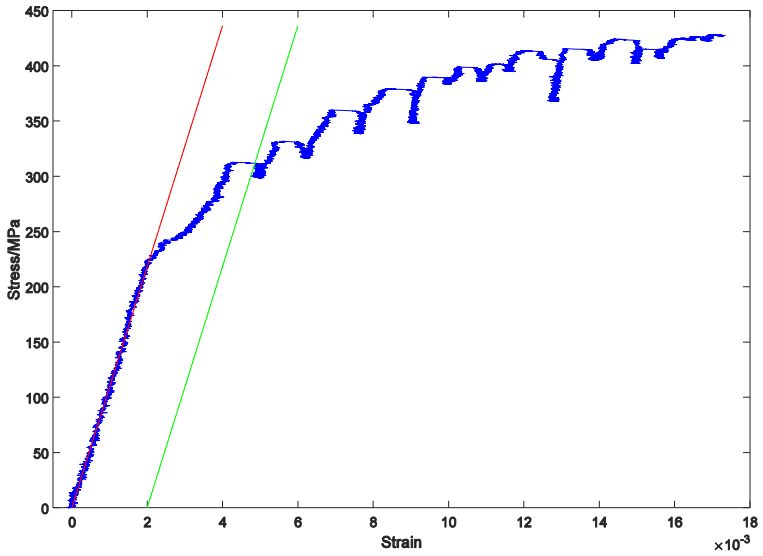


Figure 7: the stress-strain curve for a 1 μm copper nanopillar

## 2. Development of the Code for Analysing DMA Tests Data

The main concept is fitting raw load and displacement data to appropriate sloping sinusoidal functions of the form:

$$y = a_1 + a_2 x + a_3 \sin(a_4 x + a_5)$$

where $a_1 - a_5$ are parameters to be determined by chi-squared fitting.

The most crucial part that the user has to do manually is finding the begin and finish time for each load step. However this can be done very easily and quickly. The begin time should always be taken as the second crest or trough of a load step. And the finish time should be the penultimate crest or trough depending on what has been chosen for the begin time. The reason for consistently choosing either a crest or a trough as the starting point and the finishing point lies in the idea of estimating the initial, trial parameter $a_2$. As $a_2$ is the slope of the trend line, the two end points on the sloping sinusoid must be in such a way that a line joining these two points is parallel to the actual trend line. Therefore, these two points must be in-phase. And the reason for choosing the point on a crest or a trough is because firstly, they are the most obvious, well-defined point; and secondly, the rate of change of these points is the smallest so that any small difference in the phase of the two end points will not significantly affect the trial parameter $a_2$.

The algorithm automatically assigns the begin and finish time for each of the load steps and computes the corresponding static stress, Young's, storage, and loss moduli as well as the errors for each of the load steps. As I previously said, the final results are very sensitive to the initial, trial parameters, the errors in this data analysis potentially comes from the begin and finish time. So what I did to circumvent this and to ensure the reliability of the results is writing an algorithm to automatically shorten the begin and finish time by 1 second from the initial values assigned by the user. The number of times this is repeated is called

the number of samples, *n*, in my code. It is set to 4 by default. Then the resultant, final values are given by the means; and the errors are given by the standard errors.

Lastly the code automatically exports the load and displacement vs. time plot, the loss modulus vs. static stress plot, and the storage vs. static stress plot as well as the results table as a text (.txt) file. The full code is given in the appendix of this report.

## 3. Progress of the Project

So far I have obtained a large number of DMA results for 500 nm copper, 500 nm fused silica, and 1 μm copper nanopillars. All of the 500 nm copper nanopillars show the *non-constant* loss modulus vs. static stress behaviour. However there seems to be different types of this *non-constant* behaviour. Almost half of all the pillars that were analysed so far (probability = 0.48) show the *one-minimum* behaviour like what is shown in figure 8. However this behaviour might not be as obvious as what is shown in figure 5; a less obvious example is shown in figure 9. The minima are within the range of 200-300 MPa.

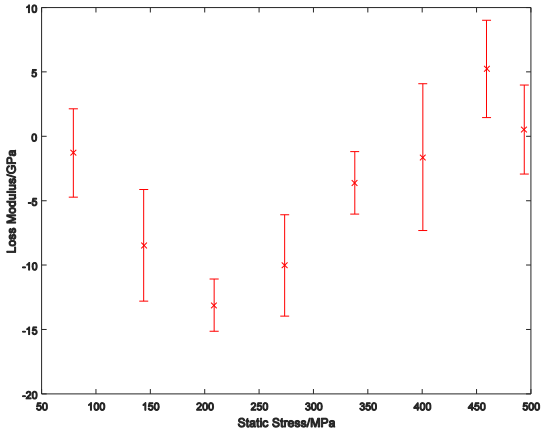



Figure 8: an example of the obvious *one-minimum* loss modulus vs. static stress behaviour

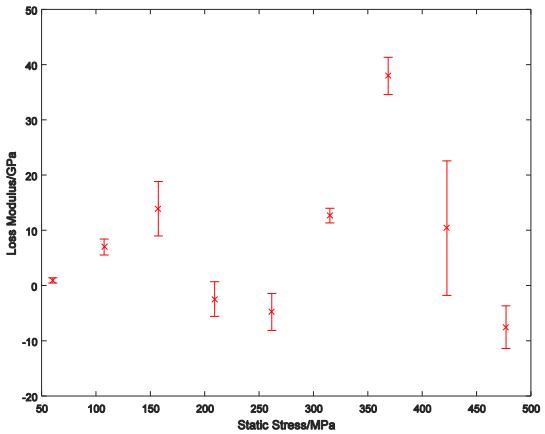
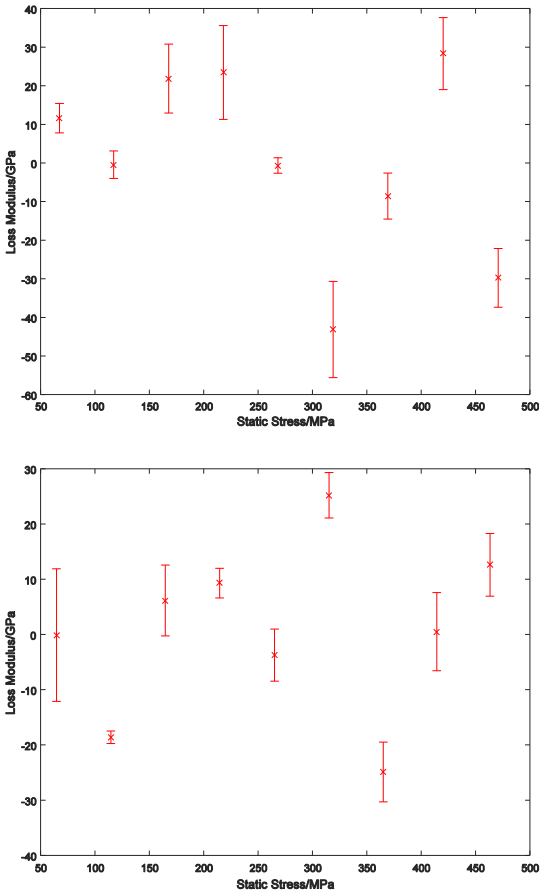



Figure 9: an example of the less obvious *one-minimum* loss modulus vs. static stress behaviour

And for the rest, there does not seem to be any obvious pattern. Examples are given in the following figures 10.




Figures 10: examples of the loss modulus vs. static stress plots that do not show the *one-minimum* behaviour

For the behaviour of storage modulus vs. static stress, almost all of the pillars show a common trend of consistently increasing storage moduli with static stress before saturating into a certain value at high static stress in the range 180-230 GPa. An example is shown in figure 11.

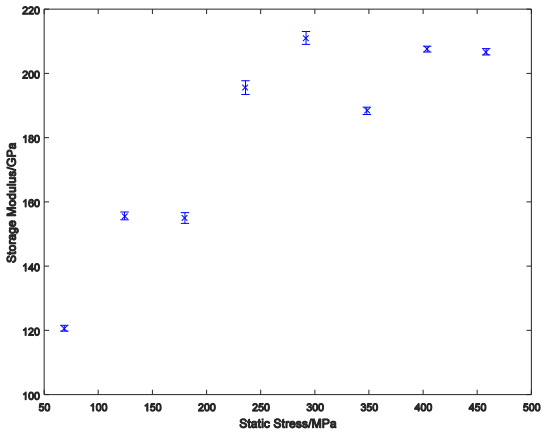



Figure 11: an example of the storage modulus vs. static stress plot

As discussed previously in section 1, the resolution of static stress was significantly improved for last batches as to scrutinise the *one-minimum* behaviour. An example of such high resolution plots is given in figures 12 and 13 for loss modulus and storage modulus respectively. The storage modulus vs. static stress plot follows the consistently increasing pattern before saturation as before. However, for the loss modulus vs. static stress plot, it can be seen that the data points lying between static stress of 200 and 300 MPa, the range where a minimum is expected to be observed, are not in accordance with the general trend of the other data points. The waveform of the data points that are expected to be around the minimum is quite bad; there are usually some discontinuities within the wave; this gives rise to large errors at the peaks. The effect is even more pronounced for the pillars with constant amplitude load which makes it very cumbersome for the code to fit the data efficiently and accurately.

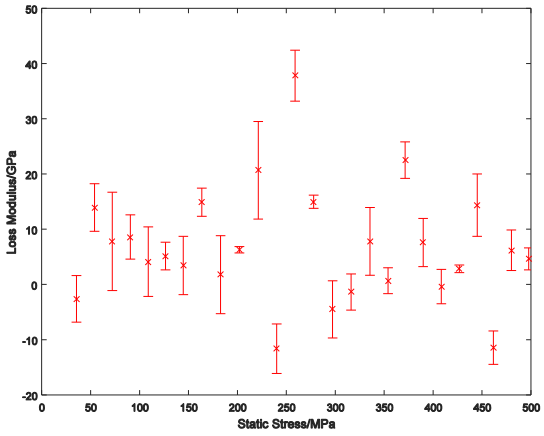



Figure 12: an examples of the high resolution loss modulus vs. static stress plot

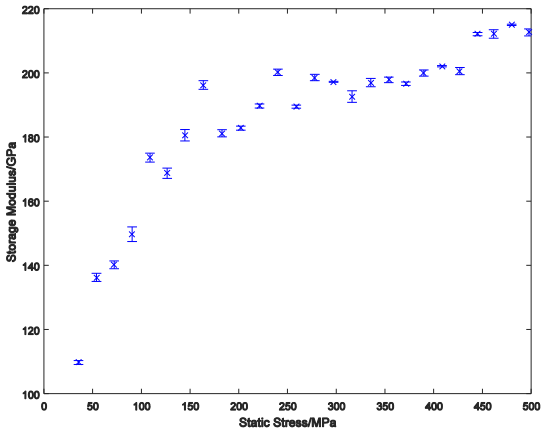



Figure 13: an examples of the high resolution storage modulus vs. static stress plot

## 4. Problems and Goals for the Rest of the Project

Since DMA tests are at the nanoscale, they are potentially prone to any floor vibration despite being done in an acoustic chamber that isolates the tests for the outside environment. And also the room where the Hysitron® Triboindenter® resides has a lot of activities going on and people all around during daytime, I solve the problem by doing experiments at night instead. The time I usually begin all necessary calibrations for the machine is 11pm and then start doing experiments after midnight when the room is totally quiet and is void of people.

Another problem is that the whole project is delayed by one week mainly due to the unexpected failure of the FEI® Versa® at the beginning of the SURF programme. By now, I should have finished performing compression tests and DMA tests for 250 μm copper nanopillars already. In fact I did already fabricate 4 batches of 250 μm copper nanopillars but the results of experiments were very bad; the Hysitron® Triboindenter® performed indentation on the pillars instead of uniaxial compression or DMA so that there was no dislocation burst seen. This problem is due to poor clearance of the pillars; the outer rings with diameters comparable to 8 μm (the diameter of the diamond tip) are too shallow so that when the tip moves down to compress the pillar, it touches the outer rings prematurely at small values of displacement. This problem is solely attributed to me because I did not design the FIB pattern very well and I had not been aware of the imminent poor clearance issue. I will re-design the FIB pattern next week.

Due to the delay, I do not think I will be able to do computer simulation for dislocations interactions as planned for week 8. Instead, I will be focusing on the size-effect experiments of copper nanopillars by performing compression and DMA tests for 250 nm, and more 1 μm copper nanopillars. And also I will further investigate the behaviour of 500 nm copper nanopillars with 10 Hz DMA tests. In addition, I will also introduce a new load function that does *load and re-load* within the elastic regime as to see if the two DMA tests performed consecutively will produce identical or comparable results or not.

## Reference


1. Karava K., 2015. *Project Proposal for Experimental Study of Crackling Noise as Micro-mechanics of Flow*. LIGO
2. Karava K., 2015. *First Progress Report for Experimental Study of Crackling Noise as Micro-mechanics of Flow*. LIGO
3. Greer J.R., De Hosson JTM., 2011. *Plasticity in Small-sized Metallic Systems: Intrinsic versus Extrinsic Size Effect.* Prog Mater Sci.
4. Ni X., et al., 2015. *What is Crackling Noise: a Study of Micro-mechanics of Flow in Metals (poster)*. LIGO

## Appendix

Table A1: summary of the parameters for DMA tests of 500 nm copper nanopillars

| Steps | Maximum Load/μN | Begin Amplitude/μN | Amplitude Type | Frequency/Hz | Number of pillars done |
|---|---|---|---|---|---|
| 11 | 140 | 3.416 | variable | 1 | 2 |
| 14 | 140 | 2 | variable | 1 | 2 |
| 14 | 140 | 4 | variable | 1 | 4 |
| 14 | 140 | 8 | variable | 1 | 2 |
| 16 | 140 | 2 | variable | 1 | 2 |
| 16 | 140 | 4 | variable | 1 | 2 |
| 16 | 140 | 8 | variable | 1 | 2 |
| 16 | 140 | 1 | variable | 1 | 3 |
| 20 | 140 | 2 | variable | 1 | 3 |
| 20 | 140 | 2 | constant | 1 | 2 |
| 20 | 140 | 4 | variable | 1 | 2 |
| 20 | 140 | 4 | constant | 1 | 2 |
| 24 | 140 | 2 | variable | 1 | 2 |
| 24 | 140 | 2 | constant | 1 | 2 |
| 24 | 140 | 4 | variable | 1 | 2 |
| 24 | 140 | 4 | constant | 1 | 2 |
| 28 | 75 | 2 | variable | 1 | 2 |
| 28 | 75 | 2 | constant | 1 | 2 |
| 28 | 75 | 2 | variable | 1 | 4 |
| 28 | 75 | 2 | constant | 1 | 4 |
| 25 | 97 | 2 | variable | 10 | 2 |
| 25 | 97 | 2 | constant | 10 | 2 |
| 25 | 97 | 3.6 | variable | 10 | 2 |
| 25 | 97 | 3.6 | constant | 10 | 2 |

Table A2: summary of the parameters for DMA tests of 500 nm fused silica nanopillars

| Steps | Maximum Load/μN | Begin Amplitude/μN | Amplitude Type | Frequency/Hz | Number of pillars done |
|---|---|---|---|---|---|
| 14 | 140 | 2 | variable | 1 | 2 |
| 14 | 140 | 4 | variable | 1 | 2 |
| 16 | 140 | 2 | variable | 1 | 2 |
| 16 | 140 | 4 | variable | 1 | 2 |

Table A3: summary of the parameters for DMA tests of 1 μm copper nanopillars

| Steps | Maximum Load/μN | Begin Amplitude/μN | Amplitude Type | Frequency/Hz | Number of pillars done |
|---|---|---|---|---|---|
| 24 | 604.8 | 6.46 | variable | 1 | 2 |
| 24 | 604.8 | 6.46 | constant | 1 | 2 |
| 24 | 604.8 | 12.92 | variable | 1 | 2 |
| 24 | 604.8 | 12.92 | constant | 1 | 2 |

## sscon.m : MatLab code for finding Young's modulus and yield strength

```
function Results = sscon(forceuN,dispnm,diameternm,heightnm)
% sscon converts force and displacement data into stress(MPa) and strain
% plots a stress-strain graph
% returns Young's modulus and yield strength

%Convert the raw data into stress and strain
stressori = 10^(-6).*(forceuN.*10^(-6))./(pi.*(10^(-9).*diameternm./2)^2);
strainori = dispnm./heightnm;

%Correct the offset
%Extract only the linear regime up to absolute strain

startmaxlinstrain = 0.001; %Initial value of linear regime strain
numtrial = 20; %Number of trial steps
step = 0.0005; %Increment

%Find r^2 for each step i
for (i=1:numtrial)
    dummy=startmaxlinstrain+(i-1)*step;
    indexlinstrain_i=find(strainori<=dummy); %Create a matrix for indices
    strainextr_i=strainori(indexlinstrain_i); %Extract the strain in the
linear regime
    stressextr_i=stressori(indexlinstrain_i); %Extract the stress in the
linear regime
    R_i=corrcoef(strainextr_i,stressextr_i); %Corr. coeff. matrix for this i
    rsq(i)=R_i(1,2).^2; %Matrix(1x10) containing corr. coeff. r^2 for i=1-10
end

%Find j for the greatest r^2
j=1;
while (rsq(j)~=max(rsq))&&(j<=numtrial)
    j=j+1;
end

%Assign j back to find the best fit linear regression
dummy_2=startmaxlinstrain+(j-1)*step;
indexlinstrain=find(strainori<=dummy_2); %Create a matrix for indices
strainextr=strainori(indexlinstrain); %Extract the strain in the linear
regime
stressextr=stressori(indexlinstrain); %Extract the stress in the linear
regime
```

```
p = polyfit(strainextr,stressextr,1); %Fit these points to a linear
regression y=p1*x+p2
yoffset = min(stressextr); %y-offset is the smallest stress
xoffset = (yoffset-p(2))./p(1); %x-offset can be found accordingly

%Point (x-offset,y-offset) defines the first point of elastic regime
%Now correct the stress and strain data
stress = stressori - yoffset;
strain = strainori - xoffset;

%Young's modulus is the slope of the linear regime
YoungGPa = p(1).*10^(-3);

%Plot the stress-strain graph
plot(strain,stress,'b','linewidth',0.001)
xlabel('Strain')
ylabel('Stress/MPa')
xlim([-0.0005,0.045])
ylim([0,3000])
hold on

%Plot the linear regression
xlr = [0:0.000001:0.004];
ylr = p(1).*xlr;
plot(xlr,ylr,'r','linewidth',0.001)

%Use the convention "strain 0.2% offset plastic strain" to determine the
yield strength
plasoffset = 0.002;
%Find and plot out the plastic offset line
xplaslr = [0:0.000001:0.006];
yplaslr = p(1).*(xplaslr-plasoffset);
plot(xplaslr,yplaslr,'g','linewidth',0.001)
hold off

%The yield stress is the value of y at the intersection point between the
%plastic offset line and the loading curve
%Truncate stress vector to start at strain of 0.2% (plasoffset)
indexstrainplas = find(strain>=plasoffset);
stressplas = stress(indexstrainplas);
%The number of entries in both vectors must agree.
si = size(stressplas,1);
terminus = 0.000001.*(si-1) + plasoffset; %Terminal value of the sequence
xplaslr2 = [plasoffset:0.000001:terminus];
yplaslr2 = p(1).*(xplaslr2-plasoffset);
stressplaslr = transpose(yplaslr2);

%The yield point is the point with the smallest difference between the stress
of the two
%lines.
err = abs(stressplas-stressplaslr);
%Find the index of such point
[index index] = min(err);
%Locate and return the yield strength in MPa
YieldMPa = stressplas(index);
Results = [YoungGPa,YieldMPa];
end
```

## dynamod.m : MatLab code for analysing DMA test data

```
function ResultsMatrix =
dynamod(pillar,diameter,height,Time,Load,Depth,BegFinTime)
% dynamod_7 returns:
% 1. E : the magnitude of the dynamic modulus
% 2. Estor : the storage modulus
% 3. Eloss : the loss modulus
% 4. Static Stress in MPa
% in GPa for oscillatory load and displacement in a given time interval
% as well as their error and the corresponding plots.
% The inputs are:
% 1. Pillar name (string)
% 1. Diameter (scalar)
% 2. Height (scalar)
% 3. Time (vector)
% 4. Load (vector)
% 5. Depth or displacement (vector)
% 6. Number of steps (scalar)
% 7. Beginning-Finishing time (matrix)

N = size(BegFinTime,1); %Number of steps
n = 4; %Number of samplings to consider for extracting the data (default=4)

for (j=1:N)
    BegTime_j = BegFinTime(j,1);   %Beginning Time for step j
    FinTime_j = BegFinTime(j,2);  %Finishing Time for step j

    for (i=1:n)
        BegTime_i=BegTime_j+(i-1);
        FinTime_i=FinTime_j-(i-1);
        %Extract the desired segment

[timeextr_i,forceextr_i,dispextr_i]=extract(Time,Load,Depth,BegTime_i,FinTime
_i);
        %Fitting
        [forceamp_i,forcephase_i,a_i]=fitforce(timeextr_i,forceextr_i);
        [dispamp_i,dispphase_i,b_i]=fitdisp(timeextr_i,dispextr_i);
        %Stress Amplitude
        stressamp_i=forceamp_i.*10^(-6)/(pi*(10^(-9).*diameter./2).^2);
        %Strain Amplitude
        strainamp_i=dispamp_i./height;
        %Phase difference
        PD_i=forcephase_i-dispphase_i;
        %Magnitude of Dynamic Modulus E in GPa
        E_i=10^(-9).*stressamp_i/strainamp_i;
        %Storage Modulus in GPa
        E_storage_i=E_i.*cos(PD_i);
        %Loss Modulus in GPa
        E_loss_i=E_i.*sin(PD_i);
        timeextr_i = timeextr_i - min(timeextr_i);
        modelload_i = a_i(1) + a_i(2).*timeextr_i +
a_i(3).*sin(a_i(4).*timeextr_i + a_i(5));
        %Static Stress in MPa for this step is the mean stress
        staticstress_i = mean(modelload_i)*10^(-12)/(pi*(10^(-
9).*diameter./2).^2);
```

```
        %Store the values in vectors
        E_j(i)=E_i;
        E_storage_j(i)=E_storage_i;
        E_loss_j(i)=E_loss_i;
        staticstress_j(i)=staticstress_i;
    end

%Store the results for step j
Step(j) = j;
YoungGPa(j) = mean(E_j);
errYoung(j) = std(E_j)./sqrt(n);
StorageGPa(j) = mean(E_storage_j);
errStorage(j) = std(E_storage_j)./sqrt(n);
LossGPa(j) = mean(E_loss_j);
errLoss(j) = std(E_loss_j)./sqrt(n);
StaticStressMPa(j) = mean(staticstress_j);
errStaticStress(j) = std(staticstress_j)./sqrt(n);

% Print out plots comparing the raw data and the model curves for only the
% specified BegTime and FinTime for each step j
[timeextr_j_ori,forceextr_j,dispextr_j]=extract(Time,Load,Depth,BegTime_j,Fin
Time_j);
timeextr_j_offset = timeextr_j_ori - min(timeextr_j_ori); %Offset time for
fitting
%Fitting
[forceamp_j,forcephase_j,a_j]=fitforce(timeextr_i,forceextr_i);
[dispamp_j,dispphase_j,b_j]=fitdisp(timeextr_i,dispextr_i);
modelload_j = a_j(1) + a_j(2).*timeextr_j_offset +
a_j(3).*sin(a_j(4).*timeextr_j_offset + a_j(5));
modeldisp_j = b_j(1) + b_j(2).*timeextr_j_offset +
b_j(3).*sin(b_j(4).*timeextr_j_offset + b_j(5));
%Plot
plot(timeextr_j_ori,forceextr_j,'b','linewidth',0.001)
hold on
plot(timeextr_j_ori,modelload_j,'r','linewidth',0.001)
plot(timeextr_j_ori,dispextr_j,'b','linewidth',0.001)
plot(timeextr_j_ori,modeldisp_j,'r','linewidth',0.001)

end

hold off
xlabel('Time/s')
ylabel('Displacement/nm OR Load/uN')
nameloaddisp=strcat(pillar,'_LoadvsDisp');
print(nameloaddisp,'-depsc2') %Save figure in the .eps format

% Plot static stress vs loss modulus
figure
p=errorbar(StaticStressMPa,LossGPa,errLoss,'.');
xlabel('Static Stress/MPa')
ylabel('Loss Modulus/GPa')
xlim([0,500])
set(p,'color','r')
set(p,'marker','x')
namep=strcat(pillar,'_LossModulus');
print(namep,'-depsc2') %Save figure in the .eps format
```

```
% Plot static stress vs storage modulus
figure
q=errorbar(StaticStressMPa,StorageGPa,errStorage,'.');
xlabel('Static Stress/MPa')
ylabel('Storage Modulus/GPa')
xlim([0,500])
set(q,'color','b')
set(q,'marker','x')
nameq=strcat(pillar,'_StorageModulus');
print(nameq,'-depsc2') %Save figure in the .eps format

%Generate the Results Matrix
BFT = transpose(BegFinTime);
ResultsMatrix_ori =
[Step;BFT;YoungGPa;errYoung;StorageGPa;errStorage;LossGPa;errLoss;StaticStres
sMPa;errStaticStress];
ResultsMatrix = transpose(ResultsMatrix_ori);

%Export .txt file for the results
TextFileName = strcat(pillar,'_Results.txt');
fileID = fopen(TextFileName,'w');
fprintf(fileID,'%s %s %s %s %s %s %s %s %s %s
%s\n','Step','BegTime','FinTime','Young','error','Storage','error','Loss','er
ror','Stress','error');
fprintf(fileID,'%4.3f %4.3f %4.3f %4.3f %4.3f %4.3f %4.3f %4.3f %4.3f %4.3f
%4.3f\n',ResultsMatrix_ori);
fclose(fileID);

end

function [timeextr,forceextr,dispextr] =
extract(Time,Load,Depth,BegTime,FinTime)
% Extract the desired sinusoidal segment by specifying the starting and
% finishing time.
i=find((Time>=BegTime)&(Time<=FinTime));
timeextr=Time(i);
forceextr=Load(i);
dispextr=Depth(i);
end

function [forceamp,forcephase,a] = fitforce(time,force)
%Fitting a sinusoid y = a1 + a2*x + a3*sin(a4*x + a5) using chisq
%Firstly, translate the time such that time starts from 0
time = time - min(time);
[i_mintime i_mintime] = min(time); %Index of min time
[i_maxtime i_maxtime] = max(time); %Index of max time
x0 = time(i_mintime);
x1 = time(i_maxtime);
force0 = force(i_mintime);
force1 = force(i_maxtime);
A2 = (force1-force0)./(x1-x0); %Slope
A1 = mean(force)-A2.*mean(time); %Y-intercept

% Trim out one period to determine the amplitude
[j_min j_min] = min(abs(time-0)); %Let's say starting from t=0s
[j_max j_max] = max(abs(time-1)); %Period=1s
forcetrim = force(j_min:j_max);
```

```
A3 = (max(forcetrim)-min(forcetrim))./2; %Amplitude

A4 = 2.*pi.*1; %Angular frequency
A5 = asin((force0-A1)/A3);

forcetrial = [A1 A2 A3 A4 A5];
a = fminsearch(@(a)sinefit(a,time,force),forcetrial); %Fitting chisq
forceamp = a(3); %Load Amplitude in uN
forcephase = a(5); %Initial Phase of force;
end

function [dispamp,dispphase,b] = fitdisp(time,disp)
%Fitting a sinusoid y = b1 + b2*x + b3*sin(b4*x + b5) using chisq
%Firstly, translate the time such that time starts from 0
time = time - min(time);
[i_mintime i_mintime] = min(time); %Index of min time
[i_maxtime i_maxtime] = max(time); %Index of max time
x0 = time(i_mintime);
x1 = time(i_maxtime);
disp0 = disp(i_mintime);
disp1 = disp(i_maxtime);
B2 = (disp1-disp0)./(x1-x0); %Slope
B1 = mean(disp)-B2.*mean(time); %Y-intercept

% Trim out one period to determine the amplitude
[j_min j_min] = min(abs(time-0)); %Let's say starting from t=0s
[j_max j_max] = max(abs(time-1)); %Period=1s
disptrim = disp(j_min:j_max);
B3 = (max(disptrim)-min(disptrim))./2; %Amplitude

B4 = 2.*pi.*1; %Angular frequency (f=1Hz)
B5 = asin((disp0-B1)/B3);

disptrial = [B1 B2 B3 B4 B5];
b = fminsearch(@(b)sinefit(b,time,disp),disptrial); %Fitting chisq
dispamp = b(3); %Displacement Amplitude in nm
dispphase = b(5); %Initial Phase of displacment
end

function chisq = sinefit(k,x,y)
% Computing a chi-squared test statistic comparing the raw data
% with a model function
% k is an array of the model function's coefficients.
% The model function takes the form: y = k1 + k2*x + k3*sin(k4*x+k5)
chi = y - (k(1) + k(2).*x + k(3).*sin(k(4).*x + k(5)));
chisq = sum(chi.^2);
end
```