Adhesive Wear as Ductile Fracture

This notebook contains some examples leading to results displayed in the article of the same title.
The data necessary to run it is a small portion of a much larger volume generated during the research process. The rest of the data can be obtained via direct correspondence with the first author.
This notebook is divided into two sections:
1.- Computing the debris particle volume from atoms' positions.
2.- Computing work and other derived variables.

This notebook illustrates calculations using only 2D data.

---

1) Assessing debris volume

This first part is concerned with the computation of debris volume (area in 2D) given the position of the atomic ensemble once the debris particle has formed.
The input is a "structure file" that contains the atomic positions computed via LAMMPS.

Initialization of some variables

In[ ]:=
```
d = 60;(*we are gonna use data for the case d=40 for illustration purposes*)
Lx = 5 * d;
Ly = 2 (Floor[8 * d / √3] + 8);
```

Insert here the path of the file "structure.dump.0000600000":

```
rootName = "";
```

Count atoms

In[ ]:=
```
(*go to the directory*)
SetDirectory[rootName];
(*name of the file*)
fileName = "structure.dump.0000600000";
(*import data*)
structure = Import[fileName, "Table"];
(*extract atom positions*)
finalPos = structure[[10 ;; -1, 3 ;; 5]];
(*select data around the file*)
coords2 = Select[finalPos,
   ((Lx/2 - 0.3 * Lx < #[[1]] < Lx/2 + 0.3 * Lx) && (Ly/2 - 0.4 * Ly < #[[2]] < Ly/2 + 0.4 * Ly)) &];
(*be even more precise: select the atoms that lie
   between the two surfaces (third body)*)
debris = Select[coords2, ((Lx/2 - 0.15 * Lx < #[[1]] < Lx/2 + 0.28 * Lx) &&
      (Ly/2 - 0.08 * Ly < #[[2]] < Ly/2 + 0.085 * Ly)) &];
(*create image to visualize the debris in green*)
auxImage = Rasterize@Show[
     ListPlot[{#[[1]], #[[2]]} & /@ coords2,
      Axes → False, AspectRatio → 1/2, PlotStyle → Gray],
     ListPlot[{#[[1]], #[[2]]} & /@ debris, Axes → False,
      AspectRatio → 1/2, PlotStyle → Darker@Green]
   ];
(*the number of atoms in the particle =
 length of the vector containing the debris atomic positions*)
auxCount = Length@debris;
```

In[ ]:=
```
auxImage
```

Out[ ]=

And the #atoms is

In[ ]:=
```
auxCount
```
Out[ ]=
```
7322
```

2) Compute maximum force, effective sliding and tangential work

Preliminaries

Set location where data is and some other parameters

```
rootName = "";
```

In[ ]:=
```
vTop = 0.05;(*sliding velocity*)
deltaT = 0.005;(*timestep*)
Navg = 5;(*size of the averaging window = 25t0*)
Nsampling = 1000;(*frequency --# timesteps -- at which the
 results in LAMMPS are sampled to be written in the log file*)
```

In[ ]:=
```
junction = 60;
Nfirst = 280;
Nsteps = 3 * junction / vTop / deltaT;
```

Retrieve forces from LAMMPS log file

In[ ]:=
```
SetDirectory[rootName];
data = Import["log.lammps", "Table"];
forceX = data[[Nfirst ;; Nfirst + Nsteps/Nsampling, -6]];
```

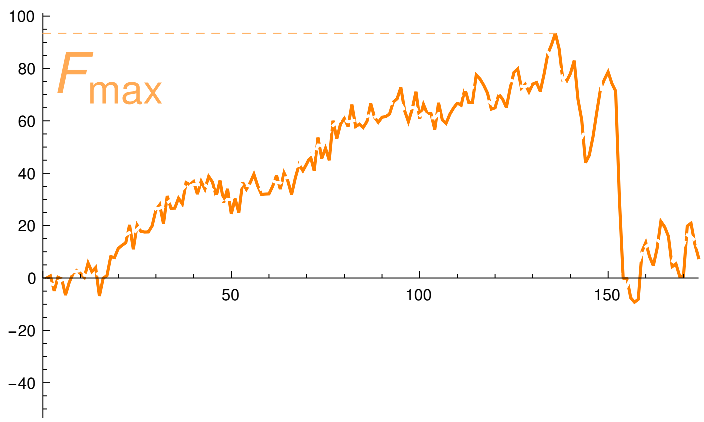
We need to perform further post-processing:
we consider that the debris creation process ends when the tangential force becomes zero after the drop,
so we gotta find the instant at which precisely that happens:

In[ ]:=
```
posNeg = 100 + LengthWhile[-1 * forceX[[100 ;; -1]], # > 0 &];
(*the 100 is added to avoid the initial
 oscillations when the force starts to ramp up*)
```

Take a look at the force profile after averaging atomic oscillations, highlighting the maximum force attained:

In[ ]:=
```
peakF = {#, -1. * forceX[[#]]} &@First[Ordering[forceX]];
```

In[ ]:=
```
ListPlot[{-1. * forceX[[1 ;; posNeg + 20]],
   -1. * MovingAverage[forceX, 5]},
  PlotRange → {{0, posNeg + 20}, Full},
  PlotStyle → {Orange, {White, Dashed}},
  Joined → True,
  Epilog → {{Dashed, Lighter@Orange, Line[{{0, peakF[[2]]}, {peakF[[1]], peakF[[2]]}}]},
    {Text[Style["Fmax", Lighter@Orange, 30], Scaled[{0.1, 0.85}]]}
   }
]
```

Out[ ]=


Compute the tangential work

Since the sliding velocity is known, the integration simplifies considerably:

In[ ]:=
```
auxWork =
   Table[-1. * deltaT * Nsampling * vTop * Total[forceX[[1 ;; ii]]], {ii, 1, Length@forceX}];
```

In similar fashion, the effective sliding S_{eff} corresponds to the sliding at the time the process ends (when the force becomes zero), thus, given that the process happens at constant imposed velocity:

In[ ]:=
```
Seff = posNeg * (Nsampling * deltaT) * vTop;
```

In[ ]:=
```
sliding = Table[ii * (deltaT * Nsampling) * vTop, {ii, 1, Length @ forceX}];
ListPlot[{Transpose @{sliding , auxWork},
   Transpose @{sliding[[1 ;; posNeg]] , auxWork[[1 ;; posNeg]]}
  },
  PlotRange → {{0, sliding[[posNeg + 200]]}, Automatic},
  Joined → True, PlotStyle → {Red, Blue},
  Epilog → {{Dashed, Lighter @Blue,
      Line[{{0, auxWork[[posNeg]]}, {sliding[[posNeg]], auxWork[[posNeg]]}}]},
    {Dashed, Lighter @Blue, Line[{{sliding[[posNeg]], auxWork[[posNeg]]},
        {sliding[[posNeg]], 0}}]},
    {Text[Style["W_t", Lighter @Blue, 30], Scaled[{0.1, 0.92}]]},
    {Text[Style["S_eff", Lighter @Blue, 30], Scaled[{0.525, 0.15}]]}
   }]
```

Out[ ]=