Glycerin,Water,Sand_enhanced

February 19, 2020

[1]:
```
%matplotlib notebook
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import glob
```

Glycerin and water

[3]:
```
glycerine_files = glob.glob('../data/glycerin*')
water_files = glob.glob('../data/water*')
sand_files = glob.glob('../data/sand*Nov__4*')
modsand_files = glob.glob('../data/sand*Feb__9*') # sand experiments
```

Make data frames:

[4]:
```
gframes = []
for f in glycerine_files:
    gframes.append(pd.DataFrame((np.loadtxt(f)).
 ↪T,columns=["time","volume","pressure"]))
wframes = []
for f in water_files:
    wframes.append(pd.DataFrame((np.loadtxt(f)).
 ↪T,columns=["time","volume","pressure"]))
sframes = []
for f in sand_files:
    sframes.append(pd.DataFrame((np.loadtxt(f)).
 ↪T,columns=["time","volume","pressure"]))
msframes = []
for f in modsand_files:
    msframes.append(pd.DataFrame((np.loadtxt(f)).
 ↪T,columns=["time","volume","pressure"]))
```

[5]:
```
print(len(gframes),len(wframes),len(sframes),len(msframes))
```

```
2 2 9 3
```

[6]:
```
gframes[0].head()
```

[6]:

	time	volume	pressure
0	53.218854	252.0	937.62
1	53.378559	251.0	937.65
2	53.570494	252.0	937.65
3	53.762241	248.0	937.63
4	53.954100	249.0	937.65

[7]:
```
wframes[0].head()
```

[7]:

	time	volume	pressure
0	154.273348	257.0	946.06
1	154.433098	252.0	946.18
2	154.592845	253.0	946.15
3	154.752598	252.0	946.11
4	154.912189	253.0	946.12

[8]:
```
sframes[0].head()
```

[8]:

	time	volume	pressure
0	3629.318228	251.0	950.74
1	3629.510149	253.0	950.78
2	3629.701823	254.0	950.75
3	3629.861731	250.0	950.83
4	3630.053497	252.0	950.87

[9]:
```
msframes[0].head()
```

[9]:

	time	volume	pressure
0	2249.242982	257.0	955.36
1	2249.402563	253.0	955.43
2	2249.562317	255.0	955.42
3	2249.722458	252.0	955.38
4	2249.881918	255.0	955.24

Smooth the data sets to find cut-points:

[10]:
```
gsmooth = []
wsmooth = []
ssmooth = []
mssmooth = []
gspan = 100
wspan = 100
sspan = 100
msspan = 100
for fr in gframes:
    gsmooth.append(fr['volume'].ewm(span=gspan).mean())
for fr in wframes:
```

```
    wsmooth.append(fr['volume'].ewm(span=wspan).mean())
for fr in sframes:
    ssmooth.append(fr['volume'].ewm(span=sspan).mean())
for fr in msframes:
    mssmooth.append(fr['volume'].ewm(span=sspan).mean())
```

[11]:
```
for fr in gsmooth:
    plt.plot(fr)
for fr in wsmooth:
    plt.plot(fr)
for fr in ssmooth:
    plt.plot(fr)
for fr in mssmooth:
    plt.plot(fr)
plt.xlim(0,400);
plt.ylim(240,270);
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Cut the noise at the beginning of each trajectory.

[12]:
```
gl = []
wa = []
sa = []
msa = []
for ii,fr in enumerate(gsmooth):
    tmp = np.max(np.where(fr>=fr[0]))
    print(tmp)
    gl.append(gframes[ii]['volume'].to_numpy()[tmp:])
for ii,fr in enumerate(wsmooth):
    tmp = np.max(np.where(fr>=fr[0]))
    print(tmp)
    wa.append(wframes[ii]['volume'].to_numpy()[tmp:])
for ii,fr in enumerate(ssmooth):
    tmp = np.max(np.where(fr>=fr[0]))
    print(tmp)
    sa.append(sframes[ii]['volume'].to_numpy()[tmp:])
for ii,fr in enumerate(mssmooth):
    tmp = np.max(np.where(fr>=fr[0]))
    print(tmp)
    msa.append(msframes[ii]['volume'].to_numpy()[tmp:])
```

```
0
289
0
```

```
45
105
154
100
213
1
159
97
116
0
0
156
145
```

[13]:
```
plt.figure();
for fr in gl:
    plt.plot(fr)
for fr in wa:
    plt.plot(fr)
for fr in sa:
    plt.plot(fr)
for fr in msa:
    plt.plot(fr)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Convert to volume.

[14]:
```
tmp_g = []
tmp_w = []
tmp_s = []
tmp_ms = []
for tmp in gl:
    tmp = (tmp[0] - tmp) / 10. * np.pi * (2.125 * 2.54)**2
    tmp_g.append(pd.Series(tmp,name='volume'))
for tmp in wa:
    tmp = (tmp[0] - tmp) / 10. * np.pi * (2.125 * 2.54)**2
    tmp_w.append(pd.Series(tmp,name='volume'))
for tmp in sa:
    tmp = (tmp[0] - tmp) / 10. * np.pi * (2.125 * 2.54)**2
    tmp_s.append(pd.Series(tmp,name='volume'))
for tmp in msa:
    tmp = (tmp[0] - tmp) / 10. * np.pi * (2.125 * 2.54)**2
    tmp_ms.append(pd.Series(tmp,name='volume'))
gl = tmp_g
```

```
wa = tmp_w
sa = tmp_s
msa = tmp_ms
```

Plot the volume data:

[15]:
```
plt.figure()
print('glycerin')
for fr in gl:
    plt.plot(fr)
print(max(fr))
print('water')
for fr in wa:
    plt.plot(fr)
print(max(fr))
print('sand')
for fr in sa:
    plt.plot(fr)
print(max(fr))
print('modified sand')
for fr in msa:
    plt.plot(fr)
print(max(fr))
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


glycerin
494.22980742472186
water
1144.0504801498191
sand
1244.7269224030033
modified sand
1244.7269224030033
```

Smooth the data sets to find cut-points:

[16]:
```
gsmooth = []
wsmooth = []
ssmooth = []
mssmooth = []
for fr in gl:
    gsmooth.append(fr.ewm(span=5).mean().to_numpy())
for fr in wa:
    wsmooth.append(fr.ewm(span=5).mean().to_numpy())
```

```
for fr in sa:
    ssmooth.append(fr.ewm(span=5).mean().to_numpy())
for fr in msa:
    mssmooth.append(fr.ewm(span=5).mean().to_numpy())
```

[17]:
```
print(gsmooth[0])
print(wsmooth[0])
print(ssmooth[0])
print(mssmooth[0])
```

```
[  0.           5.4914423    2.89023279 … 436.07408188 434.10371477
 438.89173925]
[   0.           27.45721152   31.79256071   37.59525886   37.21688387
   30.30615528   39.01884454   38.18316108   47.04147475   52.81674414
   50.43766457   48.86701054   50.89322587   49.17694455   38.862678
   41.16596502   33.537837     34.56245642   41.34950689   55.02766286
   58.04132173   56.99888284   62.40614728   62.95974004   63.3287836
   51.37127721   43.39978168   41.13636635   36.5766124    48.79088221
   53.88288147   48.12511276   50.38821969   48.84615127   53.91971329
   51.20048069   40.235253     39.02670688   44.32261299   44.8024151
   90.88430512  118.55476219   97.34131524   83.19901759  116.48203773
   99.01030064  108.71808507  118.24074262  142.89398869  168.4818899
  185.54049068  193.86208991  214.66382913  237.68405912  259.13248167
  279.5330326   302.28580372  317.4543178   345.87146821  367.86703642
  385.58154985  409.59443059  428.65381902  435.25847542  464.06798993
  492.42673678  520.48497185  533.08885934  556.7454574   581.66892661
  601.33537404  611.39553771  621.15311476  639.86137126  646.2319397
  665.73299172  678.73369307  696.55323115  717.58532704  731.6067243
  750.10672631  756.33845843  769.64535035  784.61821419  815.95573238
  836.84741117  853.82599831  871.24665897  892.01283658  911.95855755
  934.4081087   949.37447614  962.40285571  974.13924337  991.11590565
 1005.48448179 1018.11433382 1026.53423518 1041.29990659 1051.14368754
 1060.75700944 1058.01348688 1071.43847823 1089.54087631 1104.65994298
 1117.79012203 1129.59437602 1140.51467996 1147.79488258 1146.54674844
 1164.01946669 1172.61714425 1172.24732672 1184.20398683 1195.22589484
 1208.67543608 1208.48939307 1214.46696695 1233.70602261 1228.22725203
 1239.82874472 1250.61387445 1263.90556351 1263.6142857  1278.6741069
 1285.66318641 1293.37337404 1298.51349912 1295.83864662 1297.1062129
 1297.95125708 1304.61622243 1306.00873138 1309.98787197 1318.74223492
 1318.47687432 1327.4523711  1330.38523434 1335.39127778 1341.77944135
 1349.08901835 1357.01287096 1368.39704193 1369.88488668 1367.8259819
 1360.35177615 1355.36897232 1364.25030822 1370.17119882 1377.1692605
 1372.68223111 1369.69087819 1376.84904675 1375.51955656 1374.63322976
 1370.99154396 1362.46215086 1365.92829263 1356.03584869 1367.74569375
 1375.55225712 1371.60422886 1365.92140874 1371.28526583 1368.759568
 1367.07576945 1369.00403836 1373.34035225 1373.18042689 1379.17541255
 1374.01966581 1355.32849492 1352.02011816 1355.91613622 1361.56428287]
```

```
[   0.           10.98288461    5.78046558 … 1171.3002135  1159.16616777
 1169.38161163]
[   0.           21.96576922   20.23162954 … 1203.65180011 1195.98789858
 1203.08183601]
```

[18]:
```
plt.figure()
for fr in gsmooth:
    plt.plot(fr)
    print(max(fr))
for fr in wsmooth:
    plt.plot(fr)
    print(max(fr))
for fr in ssmooth:
    plt.plot(fr)
    print(max(fr))
for fr in mssmooth:
    plt.plot(fr)
    print(max(fr))
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.HTML object>


451.33870923026
469.7579601717694
1379.1754125476466
1140.7146127497083
1175.8217364287557
1101.545570213762
1103.5031782358576
1152.607444399522
1189.3737108826247
1130.4650392152378
1202.6487107705818
1086.600466264279
1221.6785344668372
1221.8947784387713
1212.3818800053414
1208.6195932476953
```

[ ]:
```
plt.figure()
for ii,fr in enumerate(gl):
    plt.plot(fr,'g.')
    plt.plot(gsmooth[ii],'k-')
    print(max(fr))
for ii,fr in enumerate(wa):
```

```
    plt.plot(fr,'b.')
    plt.plot(wsmooth[ii],'k-')
    print(max(fr))
for ii,fr in enumerate(sa):
    plt.plot(fr,'r.')
    plt.plot(ssmooth[ii],'k-')
    print(max(fr))
for ii,fr in enumerate(msa):
    plt.plot(fr,'m.')
    plt.plot(mssmooth[ii],'k-')
    print(max(fr))
```

[19]:
```
gclips = []
wclips = []
sclips = []
msclips = []
for fr in gsmooth:
    v = fr
    high = np.max(np.where(v<400.))
    low = np.max(np.where(v<100.))
    mid = np.max(np.where(v<250.))
    gclips.append([low,mid,high])
for fr in wsmooth:
    v = fr
    high = np.max(np.where(v<400.))
    low = np.max(np.where(v<100.))
    mid = np.max(np.where(v<250.))
    wclips.append([low,mid,high])
for fr in ssmooth:
    v = fr
    high = np.max(np.where(v<400.))
    low = np.max(np.where(v<100.))
    mid = np.max(np.where(v<250.))
    sclips.append([low,mid,high])
for fr in mssmooth:
    v = fr
    high = np.max(np.where(v<400.))
    low = np.max(np.where(v<100.))
    mid = np.max(np.where(v<250.))
    msclips.append([low,mid,high])
```

Convert the data from series back to numpy arrays.

[20]:
```
tmp=[]
for fr in gl:
    tmp.append(fr.to_numpy())
gl = tmp
```

```
tmp=[]
for fr in wa:
    tmp.append(fr.to_numpy())
wa = tmp
tmp=[]
for fr in sa:
    tmp.append(fr.to_numpy())
sa = tmp
tmp=[]
for fr in msa:
    tmp.append(fr.to_numpy())
msa = tmp
```

Clip the raw data and plot:

[23]:
```
gsyms = []
for i,fr in enumerate(gl):
    lo = gclips[i][0]
    mid = gclips[i][1]
    hi = gclips[i][2]
    untrans = fr[lo:mid]
    trans = fr[mid:hi]
    if len(untrans)>len(trans):
        untrans = untrans[:len(trans)]
    elif len(trans)>len(untrans):
        trans = trans[:len(untrans)]
    gsyms.append([untrans,trans])
wsyms = []
for i,fr in enumerate(wa):
    lo = wclips[i][0]
    mid = wclips[i][1]
    hi = wclips[i][2]
    untrans = fr[lo:mid]
    trans = fr[mid:hi]
    if len(untrans)>len(trans):
        untrans = untrans[:len(trans)]
    elif len(trans)>len(untrans):
        trans = trans[:len(untrans)]
    wsyms.append([untrans,trans])
ssyms = []
for i,fr in enumerate(sa):
    lo = sclips[i][0]
    mid = sclips[i][1]
    hi = sclips[i][2]
    untrans = fr[lo:mid]
    trans = fr[mid:hi]
    if len(untrans)>len(trans):
```

```
        untrans = untrans[:len(trans)]
    elif len(trans)>len(untrans):
        trans = trans[:len(untrans)]
    ssyms.append([untrans,trans])
mssyms = []
for i,fr in enumerate(msa):
    lo = msclips[i][0]
    mid = msclips[i][1]
    hi = msclips[i][2]
    untrans = fr[lo:mid]
    trans = fr[mid:hi]
    if len(untrans)>len(trans):
        untrans = untrans[:len(trans)]
    elif len(trans)>len(untrans):
        trans = trans[:len(untrans)]
    mssyms.append([untrans,trans])
```

[26]:
```
#print(gclips)
#print(wclips)
#print(wa[0][48])
#print(wsmooth[0][48])
#print(wa[0][59])
#print(wsmooth[0][59])
#print(len(wa[0]),len(wsmooth[0]))
plt.figure()
plt.plot(wa[0],'b.')
plt.plot(wsmooth[0],'r-')
plt.figure()
plt.plot(gl[0],'b.')
plt.plot(gsmooth[0],'r-')
plt.figure()
plt.plot(sa[0],'b.')
plt.plot(ssmooth[0],'r-')
plt.figure()
plt.plot(msa[0],'b.')
plt.plot(mssmooth[0],'r-')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[26]:
```
[<matplotlib.lines.Line2D at 0x7f306ec06d68>]
```

[28]:
```
plt.figure()
legend_text = []
for ii,ss in enumerate(ssyms):
    plt.plot(ss[0],ss[1],linestyle='none',marker='.')
    legend_text.append(sand_files[ii])
for ii,ss in enumerate(mssyms):
    plt.plot(ss[0],ss[1],linestyle='none',marker='.')
    legend_text.append(modsand_files[ii])
plt.legend(legend_text)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[28]:
```
<matplotlib.legend.Legend at 0x7f306eafe710>
```

[30]:
```
plt.figure()
for gs in gsyms:
    plt.plot(gs[0],'g.')
    plt.plot(gs[1],'g.')
for ws in wsyms:
    plt.plot(ws[0],'b.')
    plt.plot(ws[1],'b.')
for ss in ssyms:
    plt.plot(ss[0],'r.')
    plt.plot(ss[1],'r.')
for ss in mssyms:
    plt.plot(ss[0],'m.')
    plt.plot(ss[1],'m.')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[32]:
```
plt.figure()
for ss in ssyms:
    plt.plot(ss[0],ss[1],'r.')
for ss in mssyms:
    plt.plot(ss[0],ss[1],'m.')
for gs in gsyms:
    plt.plot(gs[0],gs[1],'g.')
for ws in wsyms:
    plt.plot(ws[0],ws[1],'b.')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[35]:
```
gfits=[]
wfits=[]
sfits=[]
msfits=[]
for gs in gsyms:
    f = np.polyfit(gs[0],gs[1],1)
    gfits.append(np.array(f).reshape(1,-1))
gfits = np.concatenate(gfits,axis=0)
for ws in wsyms:
    f = np.polyfit(ws[0],ws[1],1)
    wfits.append(np.array(f).reshape(1,-1))
wfits = np.concatenate(wfits,axis=0)
for ss in ssyms:
    f = np.polyfit(ss[0],ss[1],1)
    sfits.append(np.array(f).reshape(1,-1))
sfits = np.concatenate(sfits,axis=0)
for mss in mssyms:
    f = np.polyfit(mss[0],mss[1],1)
    msfits.append(np.array(f).reshape(1,-1))
msfits = np.concatenate(msfits,axis=0)
print('glycerin fits:\n {}'.format(gfits))
print('water fits:\n {}'.format(wfits))
print('sand fits:\n {}'.format(sfits))
print('modified sand fits:\n {}'.format(msfits))
```

```
glycerin fits:
 [[  0.71335198 194.64162503]
 [  0.70732473 182.51011289]]
water fits:
 [[  0.70625    226.97961526]
 [  0.62555066 200.48602687]]
sand fits:
```

```
 [[  0.66860552 202.76684806]
 [  0.63323844 210.57999013]
 [  0.69132742 207.77760746]
 [  0.687028   192.23046451]
 [  0.76767719 184.37224836]
 [  0.69659256 193.71621437]
 [  0.61542107 201.93657277]
 [  0.72516612 195.70915548]
 [  0.73566177 206.54453109]]
modified sand fits:
 [[  0.74870694 181.32272679]
 [  0.7398903  189.97434419]
 [  0.74041052 182.88534022]]
```

[ ]: