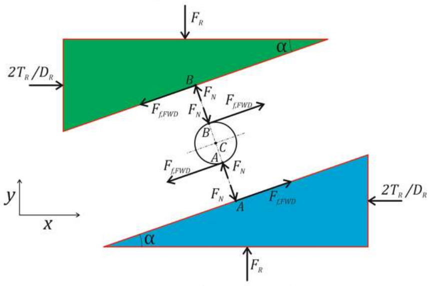
Ball Ramp Explenation

Dienstag, 3. Mai 2022 10:22

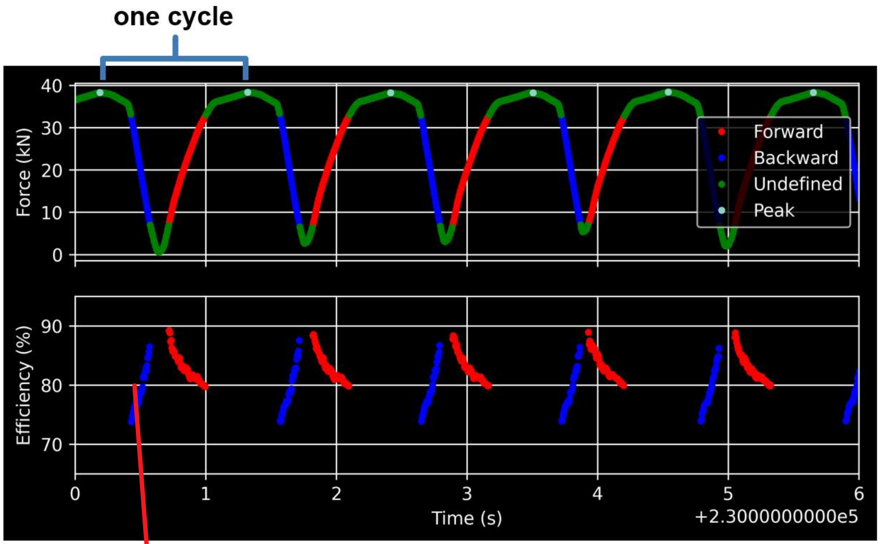
0_ReadRaw_to_explain

- Ball Ramp Friction Calculation

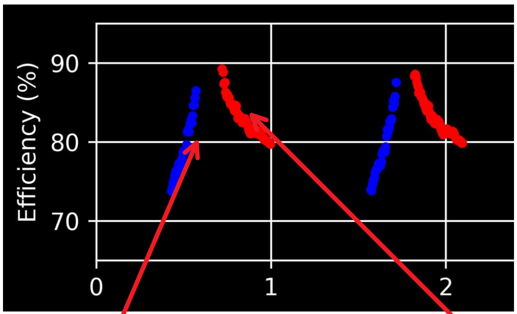


$$F_{R,FWD} = \frac{2\pi}{L} T_{R,FWD} \underbrace{\frac{(1-\mu\tan\alpha)}{(1+\mu/\tan\alpha)}}_{\eta_{FWD}}$$

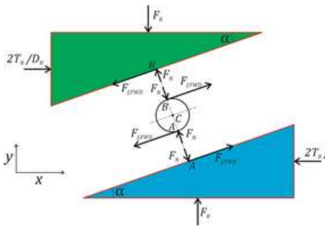
$$F_{R,BWD} = \frac{2\pi}{L} T_{R,BWD} \underbrace{\frac{(1-\mu\ /\tan\alpha)}{(1+\mu\tan\alpha)}}_{1/\eta_{BWD}}$$



Eff. In backward normally shouldn't rise. But we have unstable mü?
In Backward or Forward efficiency will differs but in terms of direction mü doesn't change.
If we look the formulas in the left bottom, we see only one mü and 2 different efficiency. Ni_fwd, Ni_bwd
This different efficiency calls in literature "self locking"
2- 1/Ni_fwd = Ni_bwd
For example If we have 50% efficiency at forward,
2-1/0.50=0
then we have zero efficiency at backward. It means it will be "selflocking"
If we rise the efficiency at forward more then 50%
2-1/0.51=0.0392
2-1/0.7=0.5714
2-1/0.60 = 0.3333
It is not going to lock itself.
If we put %90 percent both they get close
2-1/0.9 = 0.8889
Our Ball ramp more less 80% fwd
2-1/0.8=0.75



As we caluculated in the right, our fwd efficiency should be average 80% and bwd 75%. But here could be different because fwd seems 85%
And bwd = 2-1/0.85=0.8235.



In the modelling we can calculate the efficiency and the mü. And the use the mü in the simulation.

$$\underbrace{F_{R,FWD}}_{\text{measured}} = \frac{2\pi}{L} \underbrace{T_{R,FWD}}_{\text{measured}} \underbrace{\frac{(1-\mu\tan\alpha)}{(1+\mu/\tan\alpha)}}_{\eta_{FWD}}$$

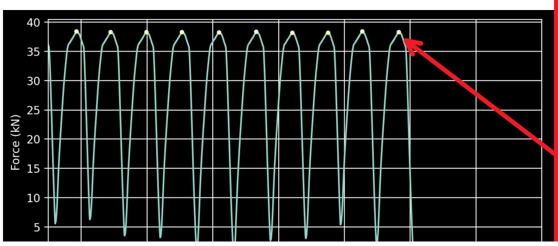
$$F_{R,BWD} = \frac{2\pi}{L} T_{R,BWD} \underbrace{\frac{(1-\mu\ /\tan\alpha)}{(1+\mu\tan\alpha)}}_{1/\eta_{BWD}}$$

We can calculate the mü but there is big problem,
Alpha is pressure angle and it not constant due to design
Even it can change over the cycles and effect the efficiency.
Due to unknown pressure angle, we can directly calculate the Efficiency from the measured F_R,FWD and T_R,FWD and According to given formula we can calculate the backward efficiency.

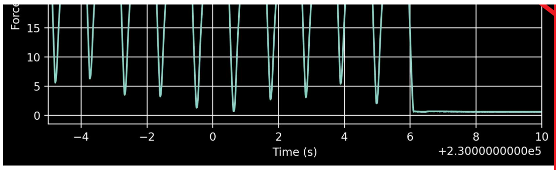
Efficiency will decrease the over life and we need to more motor Torque to apply same amont force. We need to adapt this one To matlab simulation code.

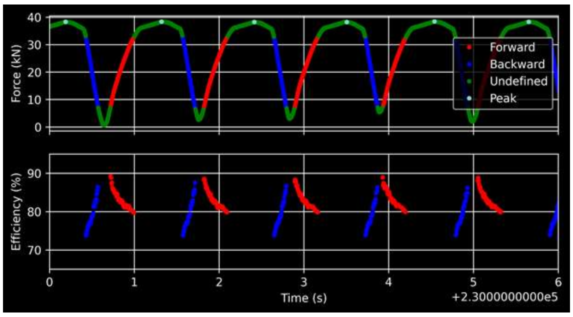
But for the more simple simulation you can calculate the directly Mü and the use direct in the simulation. But for me efficiency is more logic.

```
#%% Find force peaks and Full Closes
# Force cycles
Dt = t[1]-t[0]
#Fc_peak_idx, _ = scipy.signal.find_peaks(Fc, prominence=10000, width=30)
#F_a_peak_idx, _ = scipy.signal.find_peaks(F_a, prominence=15000
# F_a_peak_idx, _ = scipy.signal.find_peaks(F_a, prominence=15000, width=[round(1/Dt*0.5)
F_br_peak_idx, _ = scipy.signal.find_peaks(F_br,
                                           prominence=15000,
                                           distance=round(1/Dt*0.5),
                                           width=[round(1/Dt*0.5),round(1/Dt*2)])
```



This plots for peaks.

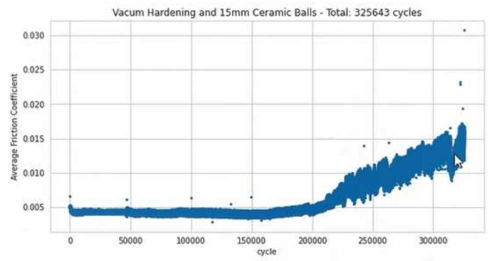


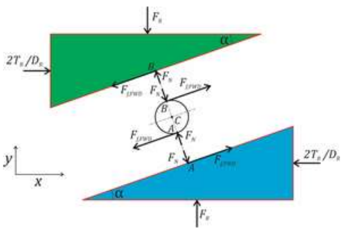



```
#convert Force and Torque to np arrays to apply it afterwards in the scipy func
F_br = np.array(sdf_date.select("Kraft_Kugelrampe_Real_kN").rdd.flatMap(lambda x: x).collect())
F_br *= 1e3
T_br = np.array(sdf_date.select("Drehmoment_Nm").rdd.flatMap(lambda x: x).collect())
Dt = 0.005
Time_br = np.array(sdf_date.select("Time_s").rdd.flatMap(lambda x: x).collect())

# get peak indexes for each Force and Torque (negative peaks)
F_br_peak_idx, _ = scipy.signal.find_peaks(F_br,
                                            prominence=10000,
                                            distance=np.round(1/Dt*0.5),
                                            width=[np.round(1/Dt*0.5),np.round(1/Dt*2)])
T_br_peak_idx, _ = scipy.signal.find_peaks(-1*T_br, height=2,prominence=2, distance=np.round(1/Dt*0.5))

# extract the relevant peak samples
F_br_peaks = F_br[F_br_peak_idx]
T_br_peaks = T_br[T_br_peak_idx]
Time_br_peaks = Time_br[F_br_peak_idx]
```

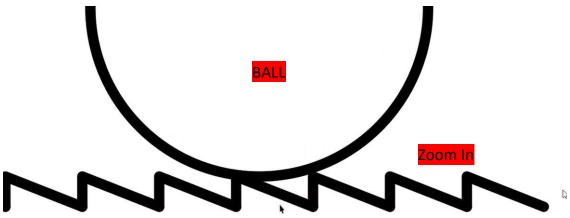




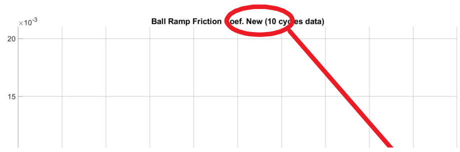
Mü looks like here same for the both direction fwd, bwd but in the reality actually not!

IF you work with "ni" (efficiency) again like in the beginning we don't have any problem.

Why is not the same?



Mü will not going to same for the both direction.



This plots for peaks.

Then we classify FWD , BWD NWD

In green areas force cannot be distinguish in terms of direction. Because of this reason it is removed.

```
#%% Classify  fwd (1) /bwd (-1) / neutral(0)
upper_F_br = max(F_br_peaks)*0.85 #85% to maximum force (upper is masked)
lower_F_br = 7e3 #For the mask lower force limitation 7kN (remove green area)
pwr_mode = np.zeros(len(F_br))
for i1 in range(1,len(F_br)):
    if (F_br[i1]-F_br[i1-1])>1e1:
        if (F_br[i1]>lower_F_br)&(F_br[i1]<upper_F_br):
            pwr_mode[i1]=1 #fwd
    elif (F_br[i1]-F_br[i1-1])<-1e1:
        if (F_br[i1]>lower_F_br)&(F_br[i1]<upper_F_br):
            pwr_mode[i1]=-1 #bwd
F_br_fwd = np.ma.masked_where(~(pwr_mode==1), F_br) #forward
F_br_bwd = np.ma.masked_where(~(pwr_mode==-1), F_br) #backward
F_br_nwd = np.ma.masked_where(~(pwr_mode==0), F_br)  #unknown
# Tests
```

In the azure,
If you want to find the different peaks? You can change the parameter here.
Choose the negative peaks…

Try to understand what are the meaning of prominance, distance, width and then you will be better understand to choose right value for parameters.

What kind of parameter?
How effects to find the peak
Is finding peaks important because we need to distinguish fwd bwd motion then efficiency?

This test could be good example for local.
It is quite similar what we have but they used ceramic balls, which deteriorate early.

We want to do cbm, also because of the factory fails. Those ball sometimes are unique and we have different behaviour. Also one day the manufactuere could say we are not producing this balls anymore. And we have different characterstics. Our CBM algorithm in this circumstances able understand what is wrong.

In the industry it is called " Quality Obsolence "

If u desing the algorthm train with lab data and detect the quality difference then it is really useful.

On the other hand we have a lot test with the different balls and those are could be Useful to undersant quality of the ball.

Mü Calculation and comments about singe mü for fwd and bwd;

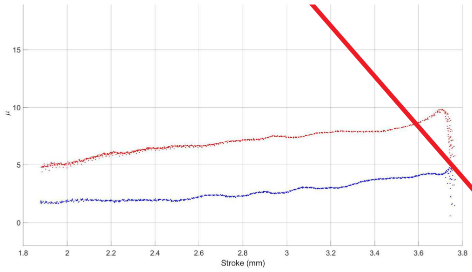
```
#In here we calculate the mü
#Mü is only one because when the ball rolling bwd or fwd is always same.
#Perhaps it is not a good assumption...
#Mü also differ when rolling fwd and bwd.
mu = np.empty(len(t))
mu[:] = np.NaN

for i1 in range(len(t)):
    if not np.ma.isMaskedArray(F_br_bwd[i1]):
        ni_bwd[i1] = (2*pi*T_br[i1])/(F_br_bwd[i1]*L)
        ni = ni_bwd[i1]; #efficiency its fwd and bwd
        mu[i1] = (1-ni)*ta/(1+ni*ta**2)# - mu_b
    if not np.ma.isMaskedArray(F_br_fwd[i1]):
        ni_fwd[i1] = F_br_fwd[i1]*L/(2*pi*T_br[i1])
        ni = ni_fwd[i1]
        mu[i1] = (1-ni)*ta/(ni+ta**2)# - mu_b
ni_bwd = np.ma.masked_invalid(ni_bwd)
ni_fwd = np.ma.masked_invalid(ni_fwd)
mu = np.ma.masked_invalid(mu)
```

RED = FWD
BLUE = BWD

Over the stroke ball ramp

There is difference in mü, whenthe ball ramp goes fwd and bwd

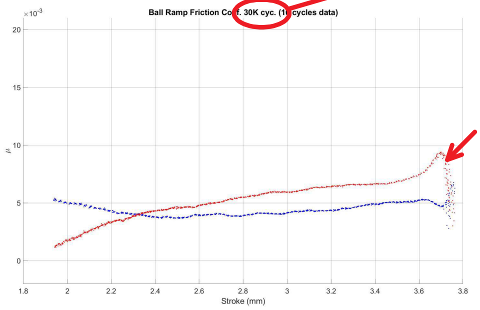


BLUE = BWD

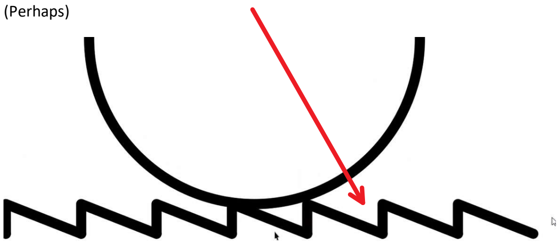
Over the stroke ball ramp

There is difference in mü, whenthe ball ramp goes fwd and bwd.

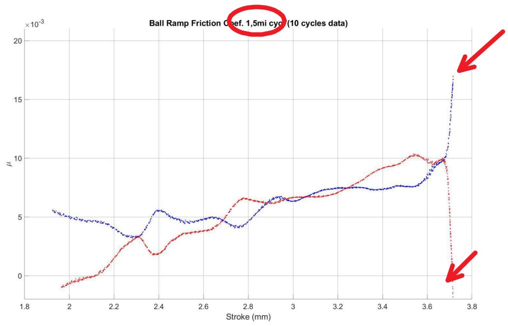
And after 30k cycles, get closer….



Because, after 3000K is getting more flat
(Perhaps)

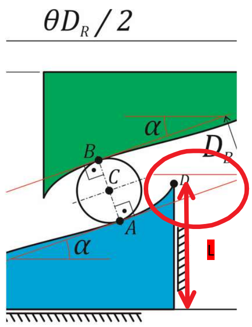


After 1.5m Cycles…..



End of the life it get higher more the other cycles and it reach the peak.
This could be damage is really concentrated in the end.

If we think simple, both mü coefficient should rise but one of them drops..
Perhaps the damage is something like this…



$$F_{R,FWD} = \frac{2\pi}{L} T_{R,FWD} \frac{\frac{(1-\mu\tan\alpha)}{(1+\mu/\tan\alpha)}}{\eta_{FWD}}$$

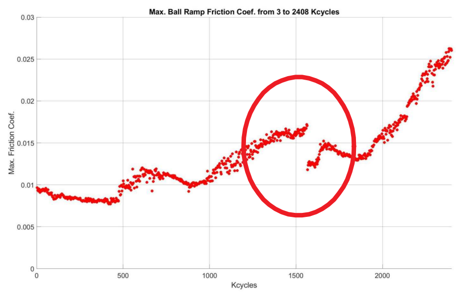
$$F_{R,BWD} = \frac{2\pi}{L} T_{R,BWD} \frac{\frac{(1-\mu/\tan\alpha)}{(1+\mu\tan\alpha)}}{1/\eta_{BWD}}$$

Different peaks for fwd and bwd reason.
In the end we got rounded stuff, It changes the pitch (L) , in the beginnig we started constat L (pitch but in the end it changes..

Its really complicated, for the simulation….
We don't want to reall ballramp model.
We need additionaly damage model! And this is creazy

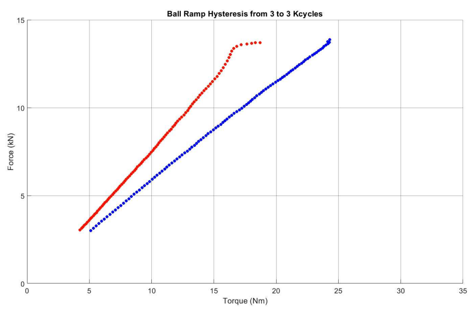
Also in the test bench we have different resaluts ->
Mu ecoefficiency grows and suddenly drops.

….

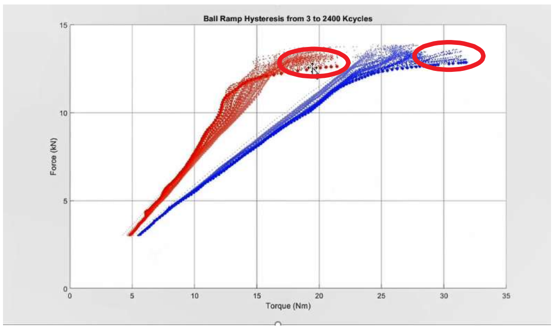


We remember our Torque versus force graphs, and we esaly see the deteroriation on high force demeands. Look carefully charesteristics of the linie.

Starting (less torque needed on high force demand)



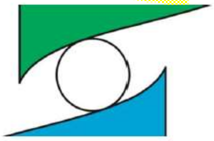
Deteriorated (more torque needed on high force demand)

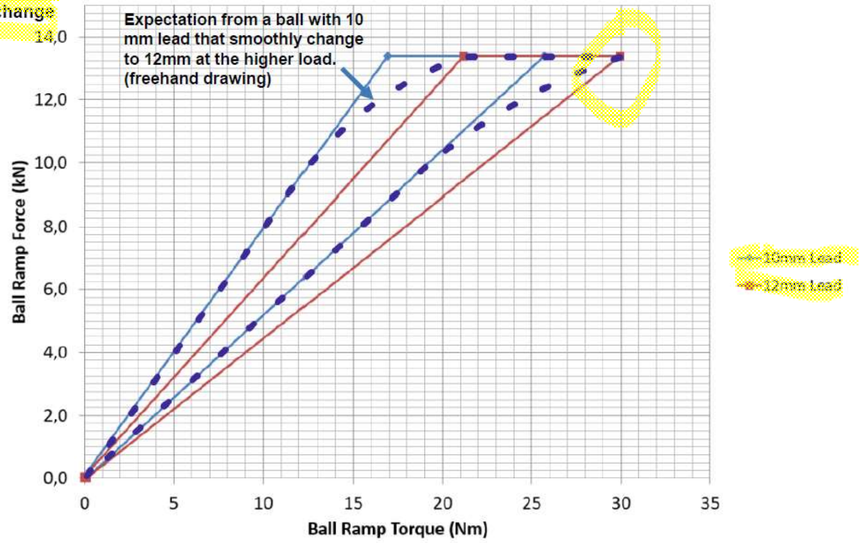


Understanding the Damage

- The ball ramp shows a micrometric damage.

If a damage only increases the lead the highest load, and don't change the friction coefficient.





Sometimes also ball destructed also sometimes we have small fatigue.

In the end he showed the mü calculation for every cycle.. But the mü if applying and reliasing will be constant but in the realty is different. But We can do average of mü fwd and bwd as a one value!! (One apprach)
Otherone with the ni... (calculate the ni for every cycle for bwd and fwd)

The general idea mü (average) ni fwd, bwd take those then we have for the every clcle of the test. -> This is deterministic simulation.

If you consider some randomness in the mü, take the std deviation, generate the random samples on the simulation.