Modelling Viscoelastic Impacts

January 10, 2021

1 Bachelor Thesis - Impacts in viscoelastic foams

2 Modelling of the problem

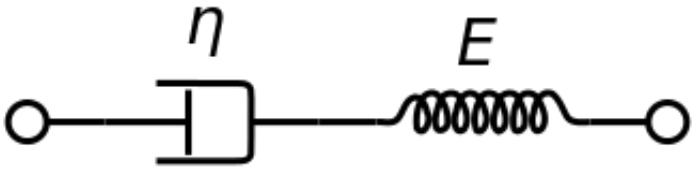
We will consider different models for impacts in viscoelastic foams. We will compare their reactions to impacts against rigid solids, and then against a similar viscoelastic solid.

2.1 Part 1: Considering Different Models

We will consider at first simple models for impacts in visocelastic foams. We can consider 4 different models, and analyse their responses to an impact. The first two will be the Kelvin-Voigt, and Maxwell Model, while the latter two will be the Maxwell and Kelvin-Voigt representations of the standard linear model.

2.1.1 Maxwell Model

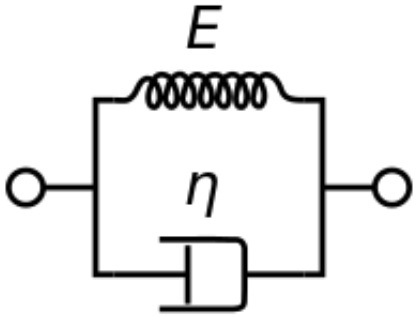
The Maxwell model is described by a damper and sprring mounted in series, as seen in the diagram below



We can see that following a deformation, the damper element will not return to it's initial length, and therre is therefore an element of permanent deformation. We can therefore discount this model, as it clearly does not represent the situation wee are trying to model, that of a boxing glove.

2.1.2 Kelvin-Voigt Model

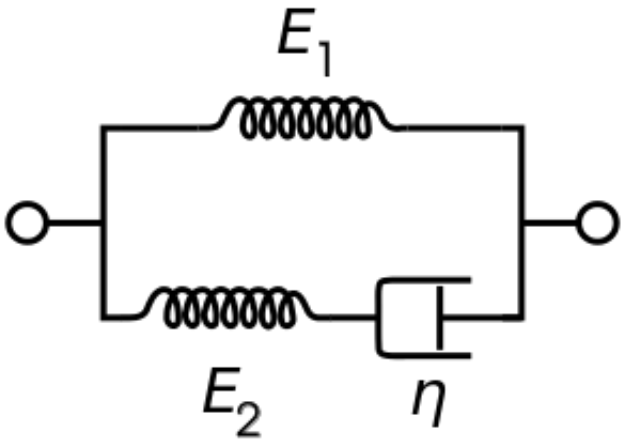
The Kelvin-Voigt model is described by a damper and spring mounted in parallel to one another, as seen in the diagram below.



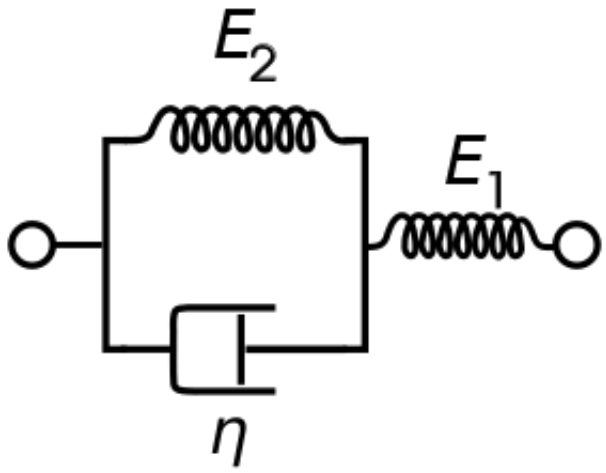
We can see that unlike the Maxwell model, when the Kelvin-Voigt system is compressed and released, it will return to its initial size. This is representative of the system we are seeking to model, and can therefore be explored further.

2.1.3 Standard Linear Solid Model

The standard linear solid model (SLSM) can be represented in one of two ways, however both of these ways overcome the inability of the basic Maxwell model to return to its initial size when unloaded. We will consider both models below. #### Maxwell representation of the SLSM The maxwell representation of the standard linear solid model is described as a spring in parallel with a basic Maxwell model (i.e. a spring and damper in series) as described in the diagram below.



Kelvin-Voigt representation of the SLSM The Kelvin-Voigt representation of the standard linear solid model is described as a spring in series with a basic Kelvin-Voigt model (i.e. a spring and damper in parallel), as described in the diagram below.



We will therefore consider the latter three models, ignoring only the basic maxwell model.

2.2 Part 2: Equations behind the Models

We will now seek to write the equations for each of these three models. We will write out the differential equations in full, and we will write them iteratively for a given time step up to a first order approximation, in order to implement them programatically. We will write as k_i the spring constants, and η_i the dmaping constants, with $\sigma = \frac{F}{A}$ the stress, and $\varepsilon = \frac{\Delta L}{L}$ the strain, where F is the force, A the surface area, and L the length. ### Kelvin-Voigt

Differential Equation: $\sigma = k\varepsilon + \eta\dot{\varepsilon}$

Iterative equation: $\sigma_{t+\delta t} = k\varepsilon_t + \eta\dot{\varepsilon}_t$

2.2.1 Maxwell representation of the standard linear solid model

Differential Equation: $\sigma + \frac{\eta}{k_2}\dot{\sigma} = k_1\varepsilon + \frac{\eta(k_1+k_2)}{k_2}\dot{\varepsilon}$

Iterative equation: $\sigma_{t+\delta t} = k_1\varepsilon_t + \frac{\eta(k_1+k_2)}{k_2}\dot{\varepsilon}_t - \frac{\eta}{k_2}\dot{\sigma}_t$

$\dot{\sigma}_{t+\delta t} = \frac{k_2}{\eta}\left(k_1\varepsilon_t + \frac{\eta(k_1+k_2)}{k_2}\dot{\varepsilon}_t - \sigma_t\right)$

2.2.2 Kelvin-Voigt representation of the standard linear solid model

Differential Equation: $\sigma + \frac{\eta}{k_1+k_2}\dot{\sigma} = \frac{k_1k_2}{K-1+k_2}\varepsilon + \frac{k_1\eta}{k_1+k2}\dot{\varepsilon}$

Iterative equation: $\sigma_{t+\delta t} = \frac{k_1k_2}{k_1+k_2}\varepsilon_t + \frac{k_1\eta_t}{k_1+k_2}\dot{\varepsilon}_t - \frac{\eta}{k_1+k_2}\dot{\sigma}_t$

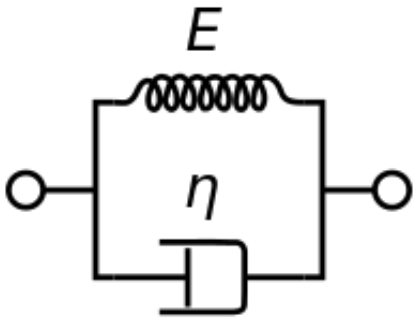
$\dot{\sigma}_{t+\delta t} = \frac{k_1+k_2}{\eta}\left(\frac{k_1k_2}{k_1+k_2}\varepsilon_t + \frac{k_1\eta_t}{k_1+k_2}\dot{\varepsilon}_t - \sigma_t\right)$

General remarks on iterative modelling For all three cases, when treating the model iteratively, we will have

$\varepsilon_{t+\delta t} = \varepsilon_t + \delta t\dot{\varepsilon}_t$

$\dot{\varepsilon}_{t+\delta t} = \dot{\varepsilon}_t + \delta t\ddot{\varepsilon}_t$

$\ddot{\varepsilon}_{t+\delta t} = \frac{A}{m}\sigma_{t+\delta t}$



title

2.3 Part 3: Implementing the models - Impact case:

We are interested in the behaviour of the model immediately after an impact. We will consider the basic case of a visco-eelastic solid impacting a rigid surface with a velocity v_0 downwards, and driven by a force F_0. We therefore initialise $\dot{\varepsilon}(t=0) = v_0$, and all other parameters are initialised to 0. The value used for F_0 is characteristic for a punch, which is around $80g \approx 800\,N$

```
In [30]: ### Import Necessary packages
         import numpy as np
         %matplotlib notebook
         import matplotlib.pyplot as plt

         ### Initialise model parameters:
         m = 0.4 # mass (approx. mass of a boxing glove)
         k = 22000 # Single-spring constant (approx. for rubber foam)
         eta = 800# single-damper constant
         A = 0.01 # Surface
         F_0 = 800 # Constant force applied

         ### Variable Initialisation
         e = 0 # Initial strain
         ev = 1 # Initial strain speed
         ea = 0 # Initial strain acceleration
         sig = 0 # Initial stress
         sigv = ev*eta # Initial stress speed
```

2.3.1 Kelvin-Voigt

$\sigma_{t+\delta t} = k\varepsilon_t + \eta\dot{\varepsilon}_t$

```
In [31]: def kelvin_voigt_iter(e,ev,ea,sig,dt):
             # Define iterative function for defininning t+dt from point t according to kelvin-vo
             signew = (k*e + eta*ev) if e>=0 else 0 # define force at t+dt from constitutive equ
             eanew = (A/m)*(F_0 - signew) # Define acceleration from force
             evnew = ev + dt*eanew # Define velocity from old velocity and acceleration
             enew = e + dt*evnew #Define strain from old strain and velocity
```

```
            return enew, evnew, eanew, signew
```

```
In [32]: def kelvin_voigt(e_init,ev_init,ea_init,sig_init,dt,N):
             #Defines the full loop of kelvin voigt model, performing iteeratve step N times, wi
             (e,ev,ea,sig) = (e_init,ev_init,ea_init,sig_init)
             (elist,evlist,ealist,siglist) = ([e],[ev],[ea],[sig])
             for i in range(N):
                 (enew,evnew,eanew,signew) = kelvin_voigt_iter(e,ev,ea,sig,dt)
                 elist.append(enew)
                 evlist.append(evnew)
                 ealist.append(eanew)
                 siglist.append(signew)
                 (e,ev,ea,sig) = (enew,evnew,eanew,signew)
             return elist,evlist,ealist,siglist
```

```
In [33]: (el,evl,eal,sigl) = kelvin_voigt(e,ev,ea,sig,0.001,2000)
```

```
In [34]: plt.figure(figsize=(9,6))
         plt.xlabel('Time')
         plt.subplot(221)
         plt.title('stress')
         plt.plot(sigl)
         plt.subplot(222)
         plt.title('strain')
         plt.plot(el)
         plt.subplot(223)
         plt.title('speed')
         plt.plot(evl)
         plt.show()
```

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

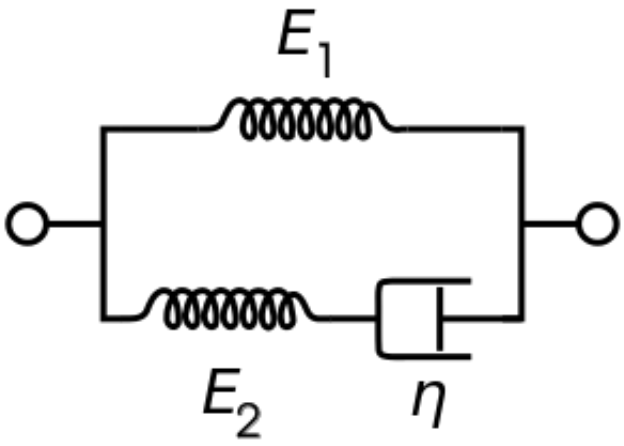
```
<IPython.core.display.HTML object>
```

2.3.2 Standard Linear Solid Model - Maxwell Representation

$$\sigma_{t+\delta t} = k_1\varepsilon_t + \frac{\eta(k_1+k_2)}{k_2}\dot{\varepsilon}_t - \frac{\eta}{k_2}\dot{\sigma}_t$$
$$\dot{\sigma}_{t+\delta t} = \frac{k_2}{\eta}\left(k_1\varepsilon_t + \frac{\eta(k_1+k_2)}{k_2}\dot{\varepsilon}_t - \sigma_t\right)$$

```
In [35]: ### Initialise Parameters
         k1 = 12000 # Double-spring constant 1
         k2 = 20000 # Double-spring constant 2
         eta = 800 # Damping constant
```

```
In [36]: def SLSM_maxwell_iter(e,ev,ea,sig,sigv,dt):
             # Define iterative function for defininin t+dt from point t according to SLSM-Maxu
```



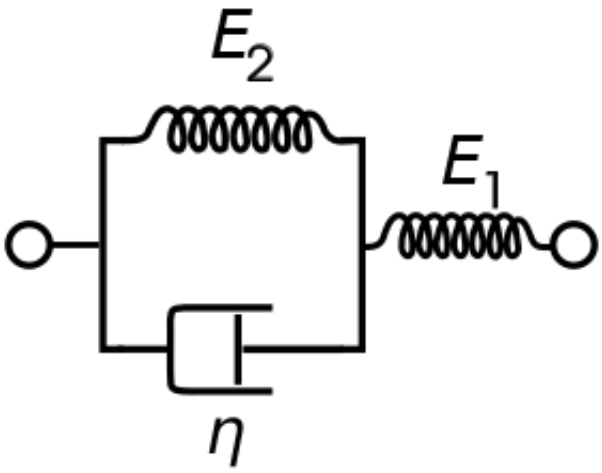
title

```
        sigvnew = (k2/eta)*(k1*e + ev*(eta*(k1+k2))/k2 - sig) if e>=0 else 0 # define force
        signew =  sig + dt*sigvnew if e>=0 else 0 # define force at t+dt from constitutive
        eanew = (A/m)*(F_0 - signew) # Define acceleration from force
        evnew = ev + dt*eanew # Define velocity from old velocity and acceleration
        enew = e + dt*evnew #Define strain from old strain and velocity
        return enew, evnew, eanew, signew, sigvnew
```

In [37]:
```
def SLSM_maxwell(e_init,ev_init,ea_init,sig_init,sigv_init,dt,N):
    #Defines the full loop of SLSM-maxwell model, performing iteeratve step N times, wi
    (e,ev,ea,sig,sigv) = (e_init,ev_init,ea_init,sig_init,sigv_init)
    (elist,evlist,ealist,siglist,sigvlist) = ([e],[ev],[ea],[sig],[sigv])
    for i in range(N):
        (enew,evnew,eanew,signew,sigvnew) = SLSM_maxwell_iter(e,ev,ea,sig,sigv,dt)
        elist.append(enew)
        evlist.append(evnew)
        ealist.append(eanew)
        siglist.append(signew)
        sigvlist.append(sigvnew)
        (e,ev,ea,sig,sigv) = (enew,evnew,eanew,signew,sigvnew)
    return elist,evlist,ealist,siglist,sigvlist
```

In [38]:
```
(el,evl,eal,sigl,sigvl) = SLSM_maxwell(e,ev,ea,sig,sigv,0.001,2000)
```

In [39]:
```
plt.figure(figsize=(9,6))
plt.xlabel('Time')
plt.subplot(221)
plt.title('stress')
plt.plot(sigl)
plt.subplot(222)
plt.title('strain')
plt.plot(el)
```



title

```
        plt.subplot(223)
        plt.title('speed')
        plt.plot(evl)
        plt.subplot(224)
        plt.title('stress speed')
        plt.plot(sigvl)
        plt.show()
```

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

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

2.3.3 Standard Linear Solid Model - Kelvin-Voigt Representation

$$\sigma_{t+\delta t} = \frac{k_1 k_2}{k_1+k_2}\varepsilon_t + \frac{k_1\eta_t}{k_1+k_2}\dot{\varepsilon}_t - \frac{\eta}{k_1+k_2}\dot{\sigma}_t$$
$$\dot{\sigma}_{t+\delta t} = \frac{k_1+k_2}{\eta}\left(\frac{k_1 k_2}{k_1+k_2}\varepsilon_t + \frac{k_1\eta_t}{k_1+k_2}\dot{\varepsilon}_t - \sigma_t\right)$$

```
In [40]: ### Initialise Parameters
         k1 = 60000 # Double-spring constant 1
         k2 = 30000 # Double-spring constant 2
         eta = 1000 # Damping constant
         sig = 0 #e*k1*k2/(k1+k2) + ev*(eta*k1)/(k1+k2)
```

```
In [41]: def SLSM_kelvin_voigt_iter(e,ev,ea,sig,sigv,dt):
             # Define iterative function for defininingt+dt from point t according to SLSM-kelv
             sigvnew = (k1+k2/eta)*(e*k1*k2/(k1+k2) + ev*(eta*k1)/(k1+k2) - sig) if e>=0 else 0
             signew =  sig + dt*sigvnew if e>=0 else 0 # define force at t+dt from constitutive
             # signew = e*k1*k2/(k1+k2) + ev*k1*eta/(k1+k2) -  eta*sigu/(k1+k2) if e>=0 else 0
             # sigvnew = (k1+k2/eta)*(e*k1*k2/(k1+k2) + ev*(eta*k1)/(k1+k2) - sig)
```

```
        eanew = (A/m)*(F_0 - signew) # Define acceleration from force
        evnew = ev + dt*eanew # Define velocity from old velocity and acceleration
        enew = e + dt*evnew #Define strain from old strain and velocity
        return enew, evnew, eanew, signew, sigvnew
```

```
In [42]: def SLSM_kelvin_voigt(e_init,ev_init,ea_init,sig_init,sigv_init,dt,N):
             #Defines the full loop of SLSM-kelvin-voigt model, performing iteratve step N times
             (e,ev,ea,sig,sigv) = (e_init,ev_init,ea_init,sig_init,sigv_init)
             (elist,evlist,ealist,siglist,sigvlist) = ([e],[ev],[ea],[sig],[sigv])
             for i in range(N):
                 (enew,evnew,eanew,signew,sigvnew) = SLSM_kelvin_voigt_iter(e,ev,ea,sig,sigv,dt)
                 elist.append(enew)
                 evlist.append(evnew)
                 ealist.append(eanew)
                 siglist.append(signew)
                 sigvlist.append(sigvnew)
                 (e,ev,ea,sig,sigv) = (enew,evnew,eanew,signew,sigvnew)
             return elist,evlist,ealist,siglist,sigvlist
```

```
In [43]: (el,evl,eal,sigl,sigvl) = SLSM_kelvin_voigt(e,ev,ea,sig,sigv,0.00001,100000)
```

```
In [44]: plt.figure(figsize=(9,6))
         plt.xlabel('Time')
         plt.subplot(221)
         plt.title('stress')
         plt.plot(sigl)
         plt.subplot(222)
         plt.title('strain')
         plt.plot(el)
         plt.subplot(223)
         plt.title('speed')
         plt.plot(evl)
         plt.subplot(224)
         plt.title('stress speed')
         i=15
         sigvl[0:i]=[sigvl[i]]*i
         plt.plot(sigvl)
         plt.show()
```

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

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

2.4 Part 4: Analysing and Comparing the models

Due to the fact that we have different input parameters for each model, it is very difficult to compare them *ceteris paribus*, however, we can consider each of them individually, and understand their dependence on theeir various inputs.

2.4.1 Kelvin - Voigt

We will start from an arbitrary *reference* configuration, ($k = 22000\,N/m$, $\eta = 800\,Ns/m$), and vary both parameters independently. Finally, we will also vary the force, to understand how this affects the early part of the motion.

```
In [45]: ## Comparison at constant eta = 800
         eta = 800
         ### Variable Initialisation
         e = 0 # Initial strain
         ev = 1 # Initial strain speed
         ea = 0 # Initial strain acceleration
         sig = 0 # Initial stress
         k = 10000
         (el1,evl1,eal1,sigl1) = kelvin_voigt(e,ev,ea,sig,0.0001,10000)
         k = 16000
         (el2,evl2,eal2,sigl2) = kelvin_voigt(e,ev,ea,sig,0.0001,10000)
         k = 22000
         (el3,evl3,eal3,sigl3) = kelvin_voigt(e,ev,ea,sig,0.0001,10000)
         k = 28000
         (el4,evl4,eal4,sigl4) = kelvin_voigt(e,ev,ea,sig,0.0001,10000)
         k = 34000
         (el5,evl5,eal5,sigl5) = kelvin_voigt(e,ev,ea,sig,0.0001,10000)
```

```
In [46]: plt.figure(figsize=(9,6))
         plt.title('stress')
         plt.plot(sigl1 , label = 'k = 10000')
         plt.plot(sigl2 , label = 'k = 16000')
         plt.plot(sigl3 , label = 'k = 22000')
         plt.plot(sigl4 , label = 'k = 28000')
         plt.plot(sigl5 , label = 'k = 32000')
         plt.xlabel('time')
         a=plt.legend()
```

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

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

Analysis We can see that a higher spring constant corresponds to a higher peak force, a shorter impact time, and more oscillations.

```
In [47]: ## Comparison at constant k = 22000
         k = 22000
         ### Variable Initialisation
         e = 0 # Initial strain
         ev = 1 # Initial strain speed
         ea = 0 # Initial strain acceleration
```

```
sig = 0 # Initial stress
eta =200
(el1,evl1,eal1,sigl1) = kelvin_voigt(e,ev,ea,sig,0.001,1000)
eta = 400
(el2,evl2,eal2,sigl2) = kelvin_voigt(e,ev,ea,sig,0.001,1000)
eta = 800
(el3,evl3,eal3,sigl3) = kelvin_voigt(e,ev,ea,sig,0.001,1000)
eta = 1200
(el4,evl4,eal4,sigl4) = kelvin_voigt(e,ev,ea,sig,0.001,1000)
eta = 1600
(el5,evl5,eal5,sigl5) = kelvin_voigt(e,ev,ea,sig,0.001,1000)
```

In [48]:
```
plt.figure(figsize=(9,6))
plt.title('stress')
plt.plot(sigl1 , label = 'eta = 200')
plt.plot(sigl2 , label = 'eta = 400')
plt.plot(sigl3 , label = 'eta = 800')
plt.plot(sigl4 , label = 'eta = 1200')
plt.plot(sigl5 , label = 'eta = 1600')
plt.xlabel('time (ms)')
b = plt.legend()
```

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

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

Analysis We can see that an increasing damping factor corresponds to a reduction of oscillations, a reduction in peak force, and a reduction in impact time.

In [49]:
```
## Comparison at constant k = 22000, eta = 800
k = 22000
eta = 800
### Variable Initialisation
e = 0 # Initial strain
ev = 1 # Initial strain speed
ea = 0 # Initial strain acceleration
sig = 0 # Initial stress
F_0 = 0
(el1,evl1,eal1,sigl1) = kelvin_voigt(e,ev,ea,sig,0.0001,10000)
F_0 = 400
(el2,evl2,eal2,sigl2) = kelvin_voigt(e,ev,ea,sig,0.0001,10000)
F_0 = 800
(el3,evl3,eal3,sigl3) = kelvin_voigt(e,ev,ea,sig,0.0001,10000)
F_0 = 1200
(el4,evl4,eal4,sigl4) = kelvin_voigt(e,ev,ea,sig,0.0001,10000)
F_0 = 1600
(el5,evl5,eal5,sigl5) = kelvin_voigt(e,ev,ea,sig,0.0001,10000)
```

```
In [50]: plt.figure(figsize=(9,6))
         plt.title('stress')
         plt.plot(sigl1 , label = 'F = 0')
         plt.plot(sigl2 , label = 'F = 400')
         plt.plot(sigl3 , label = 'F = 800')
         plt.plot(sigl4 , label = 'F = 1200')
         plt.plot(sigl5 , label = 'F = 1600')
         b = plt.legend()
```

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

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

Analysis We can see that an increased driving force naturally leads to a higher peak stress, as well as a higher plateau stress, however, we also see that it leads to aa lnger impact time.

2.4.2 Standard Linear Solid Model - Maxwell Representation

In the standard linear model, we have 4 input parameters $\eta, k_1, k_2,$ and F_0. We have a link between the two spring constants here, and the spring constant we have in the Kelvin-Voigt model, in particular the sum of the two constants in the standard linear models maxwell representation corresponds to the constant in the Kelvin-Voigt model. We want to study the relation between the ratio of the spring constants, and the ratio between the max force and the plateau force.

```
In [51]: ### Analysis of ratios at constant eta, K_tot = k1 + k2, F_0
         F_0 = 800
         eta = 800
         k_tot = 22000
         n=80
         ratiolist = [0]*n
         x_axis = [0.01*i for i in range(n)]

         for i in range(0,n):
             ratio = i/100
             k1 = ratio*k_tot
             k2 = (1-ratio)*k_tot
             (el,evl,eal,sigl,sigvl) = SLSM_maxwell(e,ev,ea,sig,sigv,0.0001,30000)
             ratiolist[i] = max(sigl)/max(sigl[-1],0.01)
```

```
In [52]: plt.figure(figsize = (9,6))
         a=plt.plot(x_axis,ratiolist)
         a=plt.xlabel('k1/k_tot')
         a=plt.ylabel('Stress_max/Stress_plateau')
```

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

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

Analysis What we see from this, is that as the proportion of k_1 increases, the ratio of the max stress over the final stress increases, relatively linearly. This relation breaks down for values greater than 0.8, as the model is no longer working properly. We can use this graph to infer the ratio of the spring constants, given the total constant, and the max-plateau ratio from real data.

2.4.3 Standard Linear Solid Model - Kelvin-Voigt Representation

We can study the same relation between the ratio of the spring constants and the max-plateau ratio for the kelvin voigt model, once again varying the ratio between k_1 and k_2, and analysing the resultant ratio between the maximum strain observed and the final strain observed. Since the total spring constant is no longer the sum of the constants, we will vary k_1 in a large interval, and calculate k_2 such that the total spring constant is equal to a given k_{tot}.

```
In [53]: ### Analysis of ratios at constant eta, K_tot = k1 + k2, F_0
         F_0 = 800
         eta = 800
         k_tot = 22000
         k_max = 200000
         n=100
         start=20
         end = 2
         ratiolist = [0]*(n-start - end)
         x_axis = [0.01*i for i in range(start,n-end)]

         for i in range(start,n-end):
             ratio = i/n
             k1 = ratio*k_max
             k2 = 1/ (1/k_tot - 1/k1)
             (el,evl,eal,sigl,sigvl) = SLSM_kelvin_voigt(e,ev,ea,sig,sigv,0.00001,100000)
             ratiolist[i-start] = max(sigl)/max(sigl[-1],0.01)

In [54]: plt.figure(figsize = (9,6))
         a=plt.plot(x_axis,ratiolist)
         a=plt.xlabel('k1/k_tot')
         a=plt.ylabel('Stress_max/Stress_plateau')

<IPython.core.display.Javascript object>


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

Analysis What we see is essentially a reciprocal curve tending to the solution of the simple Kelvin-Voigt system. This is coherent, as it corresponds to k_1 being almost rigid, and k_2 almost equal to k_{tot}.

2.5 Part 5: Conclusions

The availability of three different models gives us the flexibility to adapt to the experimental data that is generated. The main shortcoming of the basic kelvin-voigt model is the fact that it is dfficult to decouple the max/plateau ratio and the impact time, therefore, for a given impact, it may be possible to find a solution that fits the max and plateau perfectly, but does not fit the impact time. In this case, we would have to turn to our other two models to find a more adequate fit. In order to choose between them, it would likely be necessary to test both, and see which is able to to give the most reliable result.

2.5.1 Use

In order to apply these models, we would need to determine the various input parameters. The easiest to determine are the initial speed and thee constant force. The initial speed can be recovered from video, and the constant force is equal to the plateau force from real data. we would then have to recover the damping constant, and one or two spring coefficients. Recovering a total spring coefficient can be done relatively easily, by loading the glove and determining deformation. The ratio and damping factor could then be determined either by a simple grid search, or by gradient descent.

2.5.2 Further ideas

Another potential method for determining the spring constants and damping factors is the direct rheological one. In particular, submittng the material to a sinsuoidal forcing, and anaysing its response on a force gauge.

2.5.3 References

https://en.wikipedia.org/wiki/Maxwell_material
https://en.wikipedia.org/wiki/Kelvin-Voigt_material
https://en.wikipedia.org/wiki/Standard_linear_solid_model