Fatigue Homework 3

March 3, 2023

Gabe Morris

[1]:
```
# Notebook Preamble
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.optimize import fsolve, curve_fit
from scipy.interpolate import interp1d
from IPython.display import display, Markdown, Latex

plt.style.use('maroon_ipynb.mplstyle')
```

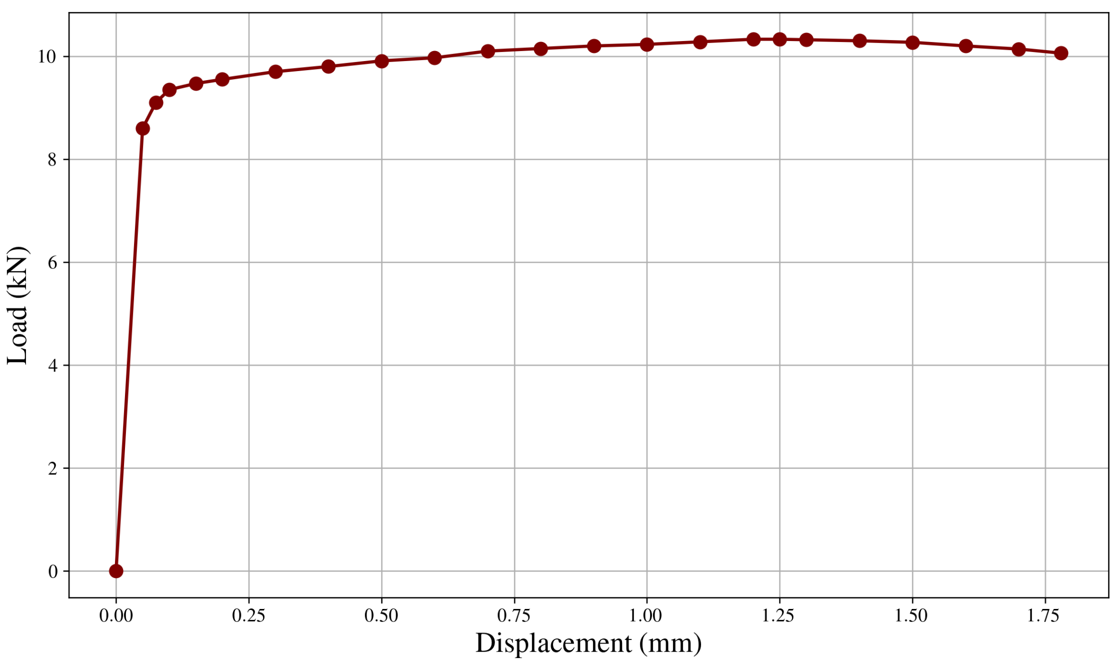
Contents

1

1.1 Given

The initial part of the load-displacement curve from the tension test of a 6061-T6 aluminum alloy is shown below. A cylindrical specimen with an initial gage section diameter of 6.3 mm and an initial uniform gage section length of 12.7 mm was used. After fracture, which occurred at a load of 7.2 kN, the minimum diameter in the neck region, D_{min}, and the neck radius, R, were measured to be 4.2 mm and 3.3 mm, respectfully.

[2]:
```python
# Measurements
D = 6.3  # mm
L = 12.7  # mm
D_min = 4.2  # mm
R = 3.3  # mm

# Plotting Load vs Displacement
df = pd.read_csv('data.csv')  # data

fig, ax = plt.subplots()
ax.set_xlabel('Displacement (mm)')
ax.set_ylabel('Load (kN)')

ax.plot(df['Displacement'], df['Load']/1000, marker='o')
plt.show()
```



1.2 Find

A) Obtain and superimpose plots of engineering and true stress-strain curves.
B) Determine the following monotonic tensile properties:

- Module of elasticity (E)
- Yield strength (S_y) at 0.2% offset
- Ultimate tensile strength (S_u)
- True fracture strength (σ_f)
- True fracture strain or ductility (ϵ_f)
- Percent reduction in area ($\%RA$)

C) Determine the strength coefficient, K, and the strain hardening exponent, n.

1.3 Solution

1.3.1 Parts A and B

The relationship for engineering stress and strain is,

$$\sigma = \frac{F}{A}$$
$$\epsilon = \frac{\Delta L}{L}$$

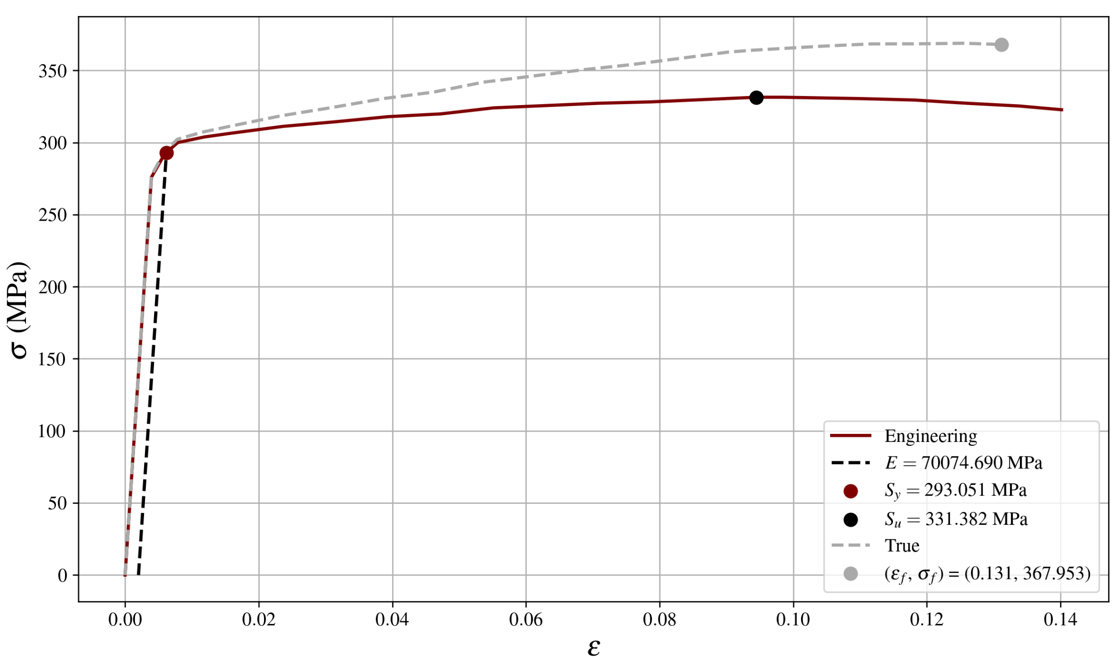
The diameter and original length were determined to fully define the above relationships. True stress and strain are as follows:

$$\sigma_{true} = \sigma(\epsilon + 1)$$
$$\epsilon_{true} = \ln(\epsilon + 1)$$

The code cell below will generate a plot that highlights the key points from Part B.

[3]:
```python
# Get stress and strain
A = np.pi/4*D**2
stress = np.array(df['Load']/A)
strain = np.array(df['Displacement']/L)

fig, ax = plt.subplots()
ax.set_xlabel(r'$\epsilon$')
ax.set_ylabel(r'$\sigma$ (MPa)')

ax.plot(strain, stress, label='Engineering', zorder=2)

# Get elastic modulus
# There appears to only be two points in the elastic region
```

```
E = (stress[1] - stress[0])/(strain[1] - strain[0])

# Find yield strength
f = interp1d(strain, stress, fill_value='extrapolate')
eps_y = fsolve(lambda x_: E*(x_ - 0.002) - f(x_), np.array([0.02, ]))[0]
Sy = f(eps_y)
ax.plot([0.002, eps_y], [0, Sy], ls='--', label=f'$E={E:.3f}$ MPa', zorder=2)
ax.scatter(eps_y, Sy, zorder=3, label=rf'$S_y={Sy:.3f}$ MPa')

# Find tensile strength
Su = np.max(stress)
ax.scatter(strain[stress == Su][0], Su, label=rf'$S_u={Su:.3f}$ MPa', zorder=3)

# Plot true stress and strain
true_stress = stress*(strain + 1)
true_strain = np.log(strain + 1)
ax.plot(true_strain, true_stress, label='True', zorder=2, ls='--')
sigma_f, epsilon_f = true_stress[-1], true_strain[-1]
ax.scatter(epsilon_f, sigma_f, label=rf'($\epsilon_f$, $\sigma_f$) =␣
 ↪({epsilon_f:.3f}, {sigma_f:.3f})', zorder=3)

ax.legend()
plt.show()
```



The true fracture stress and strain from the plot above does not apply to local fracture. The true fracture strain can be checked with the reduction of area calculation.

$$\epsilon_f = \ln\left(\frac{100}{100 - \%RA}\right)$$

[4]:
```
# Reduction of Area
Af = np.pi/4*D_min**2
RA = 100*(A - Af)/A
RA  # %
```

[4]: 55.55555555555555

[5]:
```
# Ductility
np.log(100/(100 - RA))
```

[5]: 0.8109302162163285

The true fracture stress should be computed using the bridgman correction factor to compensate for the tri-axial state of stress, which applies to cylindrical specimens. The fracture force was reported to be 7.2 kN.

$$\sigma_f = \frac{P_f/A_f}{(1 + 4R/D_{min})\ln(1 + D_{min}/4R)}$$

[6]:
```
# True fracture stress
Pf = 7.2e3
Pf/Af/((1+4*R/D_min)*np.log(1+D_min/(4*R)))  # MPa
```

[6]: 454.0842946770154

1.3.2 Part C

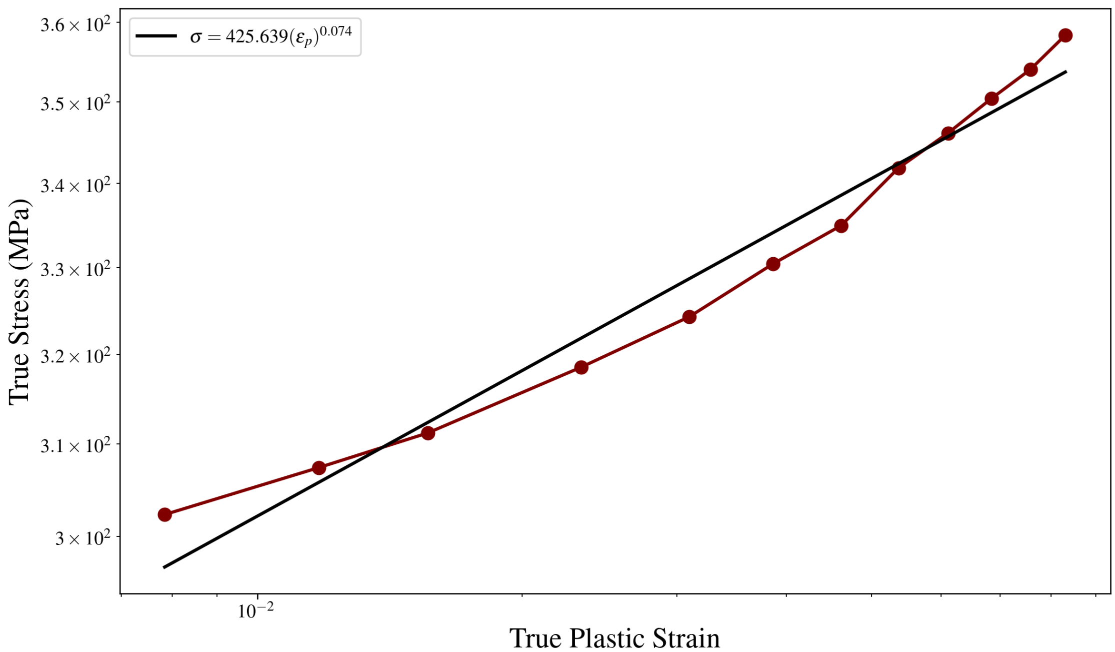
The strength coefficient and hardening exponent is determined by curve fitting a straight line for a log-log plot of the plastic true stress and strain.

$$\sigma = K(\epsilon_p)^n$$

Only the data between the yield strength and ultimate strength is used to generate the curve below.

[7]:
```
plastic = np.logical_and(stress > Sy, strain < strain[15])

fig, ax = plt.subplots()
ax.set_xscale('log')
ax.set_yscale('log')
ax.set_xlabel('True Plastic Strain')
ax.set_ylabel('True Stress (MPa)')
ax.grid(visible=False, which='both')

ax.plot(true_strain[plastic], true_stress[plastic], marker='o')
```

```python
power_func = lambda x_, K_, n_: K_*x_**n_
popt, _ = curve_fit(power_func, true_strain[plastic], true_stress[plastic])
ax.plot(true_strain[plastic], power_func(true_strain[plastic], *popt),
 label=fr'$\sigma={popt[0]:.3f}(\epsilon_p)^{{{popt[1]:.3f}}}$')

ax.legend()
plt.show()
```



From the above plot, $K = 426$ MPa and $n = 0.074$.

ME 8253 Homework 3 Gabe Morris
gnm54

2

2.1 Given

TABLE A.2 Monotonic, Cyclic, and Strain–Life Properties of Selected Engineering Alloys[a–c]

Material	Process Description	S_u MPa (ksi)	HB	E GPa (ksi $\cdot 10^3$)	%RA	S_y/S_y' MPa (ksi)	K/K' MPa (ksi)	n/n'	$\varepsilon_f/\varepsilon_f'$	σ_f/σ_f' MPa (ksi)	b	c
						Steel						
1010	HR sheet	331 (48)	—	203 (29.5)	80	200/— (29)/—	534/867 (78)/(126)	0.185/0.244	1.63/0.104	—/499 —/(72)	−0.100	−0.408
1020	HR sheet	441 (64)	109	203 (29.5)	62	262/— (38)/—	738/1962 (107)/(284)	0.190/0.321	0.96/0.377	—/1384 —/(201)	−0.156	−0.485
1038[c]	Normalized	582 (84)	163	201 (29.5)	54	331/342 (48)/(50)	1106/1340 (160)/(195)	0.259/0.220	0.77/0.309	898/1043 (130)/(151)	−0.107	−0.481
1038[c]	Q&T	649 (94)	195	219 (31.5)	67	410/364 (60)/(53)	1183/1330 (172)/(193)	0.221/0.208	1.10/0.255	1197/1009 (174)/(146)	−0.097	−0.460

[8]:
```python
sig_prime = 1043  # MPa
E = 201e3  # MPa
eps_prime = 0.309
b, c = -0.107, -0.481
```

2.2 Find

A) Superimpose plots of elastic, plastic, and total strain versus life curves

B) Determine the total, elastic, and plastic strain amplitudes for smooth uni-axial test specimens for life to:

- 10^3 cycles ($2 \cdot 10^3$ reversals)
- 10^5 cycles ($2 \cdot 10^5$ reversals)

C) Repeat Part B using the equation of the Method of Universal Slopes below and comment on any differences from Part B.

$$\frac{\Delta\epsilon}{2} = \frac{\Delta\epsilon_e}{2} + \frac{\Delta\epsilon_p}{2} = 0.623\left(\frac{S_u}{E}\right)^{0.832}\left(2N_f\right)^{-0.09} + 0.0196\left(\epsilon_f\right)^{0.155}\left(\frac{S_u}{E}\right)^{-0.53}\left(2N_f\right)^{-0.56}$$

D) If the mean stress is $\sigma_m = 0.2\sigma_f'$, what approximate effect does this have on 1038 steel strain amplitude for 10^3 and 10^5 cycles using the three mean stress equations:

- Morrow's parameter:

$$\frac{\Delta\epsilon}{2} = \epsilon_a = \frac{\sigma_f' - \sigma_m}{E}\left(2N_f\right)^b + \epsilon_f'\left(2N_f\right)^c$$

- Alternative version of Morrow's mean stress parameter:

$$\frac{\Delta\epsilon}{2} = \epsilon_a = \frac{\sigma_f' - \sigma_m}{E}\left(2N_f\right)^b + \epsilon_f'\left(\frac{\sigma_f' - \sigma_m}{\sigma_f'}\right)^{\frac{c}{b}}\left(2N_f\right)^c$$

- Smith, Watson, and Topper (SWT parameter):

$$\sigma_{\max}\epsilon_a E = \left(\sigma_f'\right)^2 \left(2N_f\right)^{2b} + \sigma_f'\epsilon_f' E \left(2N_f\right)^{b+c}$$

where $\sigma_{max} = \sigma_m + \sigma_a$ and ϵ_a is the alternating strain (strain amplitude)

Note: In the SWT parameter case, the stress amplitude σ_a and the strain amplitude ϵ_a are found solving both the SWT nonlinear equation and the following Ramberg-Osgood nonlinear equation:

$$\epsilon_a = \frac{\sigma_a}{E} + \left(\frac{\sigma_a}{K'}\right)^{1/n'}$$

E) Compare the results of Part D with those of Part B. Where does σ_m have its greatest influence?

2.3 Solution

For elastic behavior, the Basquin equation is used:

$$\frac{\Delta\sigma}{2} = \sigma_f'(2N_f)^b \rightarrow \frac{\Delta\epsilon_e}{2} = \frac{\sigma_f'}{E}(2N_f)^b$$

For plastic behavior, the Manson-Coffin relationship is used:

$$\frac{\Delta\epsilon_p}{2} = \epsilon_f'(2N_f)^c$$

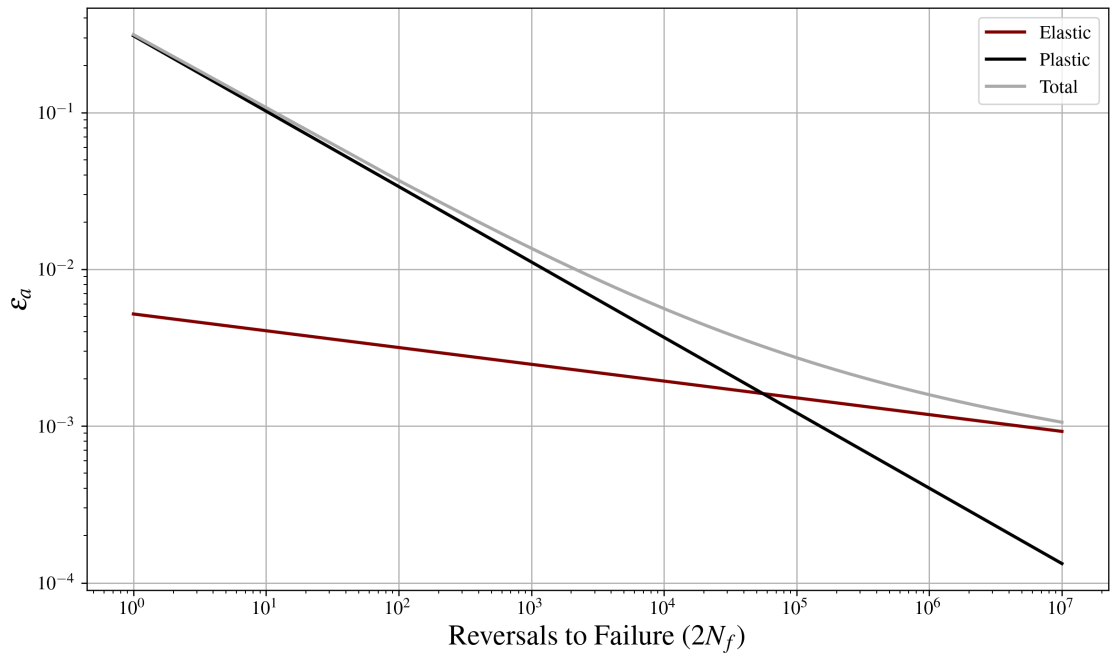
The total strain amplitude is the sum of the above relationships:

$$\epsilon_a = \frac{\Delta\epsilon_e}{2} + \frac{\Delta\epsilon_p}{2} = \frac{\sigma_f'}{E}(2N_f)^b + \epsilon_f'(2N_f)^c$$

2.3.1 Part A

[9]:
```
# Elastic function
eps_elastic = lambda Nt_: sig_prime/E*Nt_**b

# Plastic function
eps_plastic = lambda Nt_: eps_prime*Nt_**c

# Total
eps_total = lambda Nt_: eps_elastic(Nt_) + eps_plastic(Nt_)

# Making plot
Nt = np.linspace(1, 1e7, 100_000)  # number of reversals
fig, ax = plt.subplots()
ax.set_xlabel('Reversals to Failure ($2N_f$)')
ax.set_ylabel(r'$\epsilon_a$')
ax.set_xscale('log')
ax.set_yscale('log')
```

```
ax.plot(Nt, eps_elastic(Nt), label='Elastic')
ax.plot(Nt, eps_plastic(Nt), label='Plastic')
ax.plot(Nt, eps_total(Nt), label='Total')

ax.legend()
plt.show()
```



2.3.2 Part B

The functions created above can be simply reused. The number of reversals is used for the strain life models.

1000 Cycles

[10]:
```
# Elastic strain
eps_elastic(2e3)
```

[10]: 0.0023007956175003924

[11]:
```
# Plastic strain
eps_plastic(2e3)
```

[11]: 0.007982943426230897

[12]:
```
# Total strain
B_3 = eps_total(2e3)
```

```
B_3
```

[12]: 0.01028373904373129

100,000 Cycles

[13]:
```
# Elastic strain
eps_elastic(2e5)
```

[13]: 0.001405652733449556

[14]:
```
# Plastic strain
eps_plastic(2e5)
```

[14]: 0.0008712906458976844

[15]:
```
# Total strain
B_5 = eps_total(2e5)
B_5
```

[15]: 0.002276943379347244

2.3.3 Part C

The Universal Slopes method can be used to compare the total strain from Part A. The only additional parameter is the ductility ($\epsilon_f = 0.77$) and the tensile strength ($S_u = 582$ MPa).

1000 Cycles

[16]:
```
Su = 582   # MPa
eps_f = 0.77
0.623*(Su/E)**0.832*2e3**-0.09 + 0.0196*eps_f**0.155*(Su/E)**-0.53*2e3**-0.56
```

[16]: 0.008336739266295948

100,000 Cycles

[17]:
```
0.623*(Su/E)**0.832*2e5**-0.09 + 0.0196*eps_f**0.155*(Su/E)**-0.53*2e5**-0.56
```

[17]: 0.002053379976480394

For 1,000 cycles, the value from Part B is 23.4% greater. For 100,000 cycles, the value from Part B is 10.9% greater; therefore, the Universal Slopes method can be considered as a very conservative approximation, since it calls for a smaller strain amplitude for the same number of cycles.

2.3.4 Part D

Morrow's Parameter

$$\frac{\Delta\epsilon}{2} = \epsilon_a = \frac{\sigma_f' - \sigma_m}{E}\left(2N_f\right)^b + \epsilon_f'\left(2N_f\right)^c$$

1,000 Cycles

[18]:
```
sig_mean = 0.2*sig_prime

D_M3 = (sig_prime - sig_mean)/E*2e3**b + eps_prime*2e3**c
D_M3
```

[18]: 0.009823579920231211

100,000 Cycles

[19]:
```
D_M5 = (sig_prime - sig_mean)/E*2e5**b + eps_prime*2e5**c
D_M5
```

[19]: 0.0019958128326573285

Alternative Morrow's Parameter

$$\frac{\Delta\epsilon}{2} = \epsilon_a = \frac{\sigma_f' - \sigma_m}{E}\left(2N_f\right)^b + \epsilon_f'\left(\frac{\sigma_f' - \sigma_m}{\sigma_f'}\right)^{\frac{c}{b}}\left(2N_f\right)^c$$

1,000 Cycles

[20]:
```
D_AM3 = (sig_prime - sig_mean)/E*2e3**b + eps_prime*((sig_prime - sig_mean)/
 ↪sig_prime)**(c/b)*2e3**c
D_AM3
```

[20]: 0.004768297871995594

100,000 Cycles

[21]:
```
D_AM5 = (sig_prime - sig_mean)/E*2e5**b + eps_prime*((sig_prime - sig_mean)/
 ↪sig_prime)**(c/b)*2e5**c
D_AM5
```

[21]: 0.0014440589587003646

Smith, Watson, and Topper Parameter

$$\sigma_{\max}\epsilon_a E = \left(\sigma_f'\right)^2\left(2N_f\right)^{2b} + \sigma_f'\epsilon_f' E\left(2N_f\right)^{b+c}$$

$$\epsilon_a = \frac{\sigma_a}{E} + \left(\frac{\sigma_a}{K'}\right)^{1/n'}$$

$$\sigma_{max} = \sigma_m + \sigma_a$$

Although `fsolve` may be used as a numerical solver, `sympy` also has a numerical solver and is able to clearly show the equations through LaTeX. There are three equations and three unknowns (σ_{max}, σ_a, and ϵ_a).

1,000 Cycles

[22]:
```
# Define symbols
sig_max_, eps_a_, E_ = sp.symbols(r'\sigma_{max} \epsilon_a E')
sig_prime_, Nr_, b_, c_, eps_prime_ = sp.symbols(r'\sigma^`_f N_t b c␣
↪\epsilon^`_f')
K_prime_, n_prime_, sig_m_, sig_a_ = sp.symbols(r'K^` n^` \sigma_m \sigma_a')

eq1 = sp.Eq(sig_max_*eps_a_*E_, sig_prime_**2*Nr_**(2*b_) +␣
↪sig_prime_*eps_prime_*E_*Nr_**(b_ + c_))
eq2 = sp.Eq(eps_a_, sig_a_/E_ + (sig_a_/K_prime_)**(1/n_prime_))
eq3 = sp.Eq(sig_max_, sig_m_ + sig_a_)
display(eq1, eq2, eq3)

display(Markdown('---'))
subs3 = [(sig_prime_, sig_prime), (sig_m_, sig_mean), (eps_prime_, eps_prime),␣
↪(E_, E), (Nr_, 2e3), (b_, b), (c_, c), (K_prime_, 1340), (n_prime_, sp.S('0.
↪22'))]

with sp.evaluate(False):
    eq1_ = sp.Eq(eq1.lhs.subs(subs3), eq1.rhs.subs(subs3))
    eq2_ = sp.Eq(eq2.lhs.subs(subs3), eq2.rhs.subs(subs3))
    eq3_ = sp.Eq(eq3.lhs.subs(subs3), eq3.rhs.subs(subs3))
display(eq1_, eq2_, eq3_)
```

$$E\epsilon_a\sigma_{max} = EN_t^{b+c}\epsilon^{`}_f\sigma^{`}_f + N_t^{2b}\left(\sigma^{`}_f\right)^2$$

$$\epsilon_a = \left(\frac{\sigma_a}{K^{`}}\right)^{\frac{1}{n^{`}}} + \frac{\sigma_a}{E}$$

$$\sigma_{max} = \sigma_a + \sigma_m$$

$$201000.0\epsilon_a\sigma_{max} = \frac{1043^2}{5.08651604839233} + 201000.0 \cdot 0.309 \cdot 1043 \cdot 2000.0^{-0.481-0.107}$$

$$\epsilon_a = \frac{\sigma_a}{201000.0} + \left(\frac{\sigma_a}{1340}\right)^{\frac{1}{0.22}}$$

$$\sigma_{max} = \sigma_a + 208.6$$

[23]:
```
# Solve the system
sol = sp.nsolve([eq1_, eq2_, eq3_], [eps_a_, sig_a_, sig_max_], [0.1, 100,␣
↪100], dict=True)[0]
for key, value in sol.items():
    display(sp.Eq(key, sp.re(value)))
```

```
D_SMT3 = float(sp.re(sol[eps_a_]))
```

$\epsilon_a = 0.00750877406155887$

$\sigma_a = 424.767988888844$

$\sigma_{max} = 633.367988888844$

100,000 Cycles

[24]:
```
subs5 = [(sig_prime_, sig_prime), (sig_m_, sig_mean), (eps_prime_, eps_prime),␣
 ↪(E_, E), (Nr_, 2e5), (b_, b), (c_, c), (K_prime_, 1340), (n_prime_, sp.S('0.
 ↪22'))]

with sp.evaluate(False):
    eq1_ = sp.Eq(eq1.lhs.subs(subs5), eq1.rhs.subs(subs5))
    eq2_ = sp.Eq(eq2.lhs.subs(subs5), eq2.rhs.subs(subs5))
    eq3_ = sp.Eq(eq3.lhs.subs(subs5), eq3.rhs.subs(subs5))
display(eq1_, eq2_, eq3_)
```

$$201000.0\epsilon_a\sigma_{max} = 201000.0 \cdot 0.309 \cdot 1043 \cdot 200000.0^{-0.481-0.107} + \frac{1043^2}{13.6276326805364}$$

$$\epsilon_a = \frac{\sigma_a}{201000.0} + \left(\frac{\sigma_a}{1340}\right)^{\frac{1}{0.22}}$$

$$\sigma_{max} = \sigma_a + 208.6$$

[25]:
```
# Solve the system
sol = sp.nsolve([eq1_, eq2_, eq3_], [eps_a_, sig_a_, sig_max_], [0.01, 200,␣
 ↪300], dict=True)[0]
for key, value in sol.items():
    display(sp.Eq(key, sp.re(value)))

D_SMT5 = float(sp.re(sol[eps_a_]))
```

$\epsilon_a = 0.00146888148497975$

$\sigma_a = 229.365169607826$

$\sigma_{max} = 437.965169607826$

2.3.5 Part E

The table below shows a summary of the different methods for obtaining the stress amplitude. The percent difference is relative to the reversible case in which the mean stress is zero.

	1,000 Cycles	% Difference at 1,000	100,000 Cycles	% Difference at 100,000
Reversible	0.010284	0.000000	0.002277	0.000000
Morrow	0.009824	4.684230	0.001996	14.086018
Alternative Morrow	0.004768	115.668973	0.001444	57.676622
SWT	0.007509	36.956299	0.001469	55.012055

The overall effect of having a mean stress other than zero is that the strain amplitude must be decreased to have the same number of cycles as a fully reversible case. The Alternative Morrow method, however, is the most conservative of the three options above with having lower values of the strain amplitude compared to the others. The original Morrow method is the least conservative with changes that are less than the other two.