Machine Design Final Exam

August 1, 2022

Gabe Morris

[1]:
```
# Notebook Preamble
import sympy as sp
import matplotlib.pyplot as plt
from IPython.display import display

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

def sub_return(expression, substitution, sym=None):
    """
    Displays a substituted expression and returns the value.

    :param expression: Symbolic expression
    :param substitution: A list of tuples to be passed into the .sub method
    :param sym: An optional symbol to display prior to operations
    :return: A simplified value of what is substituted.
    """
    with sp.evaluate(False):
        expr_sub = expression.subs(substitution)
    simp = expr_sub.doit().n()
    if sym is None:
        lev1 = sp.Eq(expression, expr_sub)
        lev2 = sp.Eq(lev1, simp, evaluate=False)
        display(lev2)
    else:
        lev1 = sp.Eq(sym, expression)
        lev2 = sp.Eq(lev1, expr_sub, evaluate=False)
        lev3 = sp.Eq(lev2, simp, evaluate=False)
        display(lev3)
    return simp

def get_principal(s_x, s_y, t_xy):
    """
    Shows the principal stress calculations for a 2D element.

    :param s_x: normal stress along the x-axis
    :param s_y: normal stress along the y-axis
```

```
    :param t_xy: shear stress on face x in the direction of y
    :return: principal stresses
    """
    s_x_, s_y_, t_xy_ = sp.symbols(r'\sigma_x \sigma_y \tau_{xy}')
    symbols_ = sp.symbols(r'\sigma_1 \sigma_2 \tau_1 \tau_2')
    expr1_ = (s_x_ + s_y_)/2 + sp.sqrt(((s_x_ - s_y_)/2)**2 + t_xy_**2)
    expr2_ = (s_x_ + s_y_)/2 - sp.sqrt(((s_x_ - s_y_)/2)**2 + t_xy_**2)
    expr3_ = sp.sqrt(((s_x_ - s_y_)/2)**2 + t_xy_**2)
    expr4_ = -sp.sqrt(((s_x_ - s_y_)/2)**2 + t_xy_**2)
    sub_list = [(s_x_, s_x), (s_y_, s_y), (t_xy_, t_xy)]
    ans = []
    for sym_, expr_ in zip(symbols_, [expr1_, expr2_, expr3_, expr4_]):
        ans.append(sub_return(expr_, sub_list, sym=sym_))
    return ans

def von_mises(s_x, s_y, s_z, t_xy, t_yz, t_zx):
    return 1/sp.sqrt(2)*sp.sqrt((s_x - s_y)**2 + (s_y - s_z)**2 + (s_z -␣
 ↪s_x)**2 + 6*(t_xy**2 + t_yz**2 + t_zx**2))

def log10(x_):
    return sp.log(x_)/sp.log(10)
```

Contents

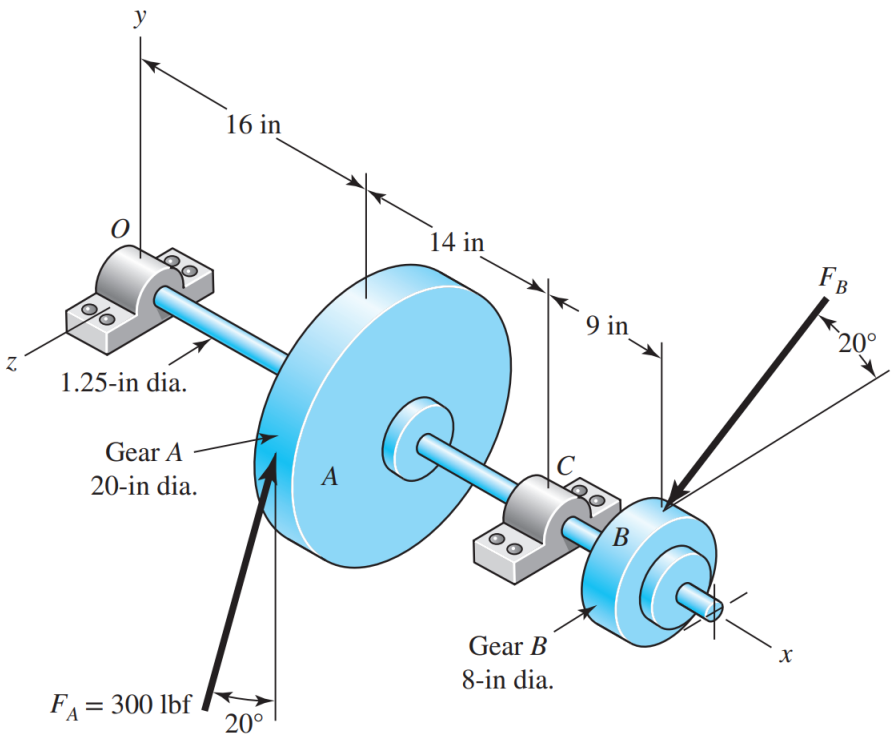
1 Problem 3-83 **4**
- 1.1 Given . . . 4
- 1.2 Find . . . 4
- 1.3 Solution . . . 5
  - 1.3.1 Part A and Part B . . . 5
  - 1.3.2 Part C . . . 6
  - 1.3.3 Part D . . . 6
  - 1.3.4 Part E . . . 7

2 Problem 5-36 **8**
- 2.1 Given . . . 8
- 2.2 Find . . . 8
- 2.3 Solution . . . 8
  - 2.3.1 Element A . . . 8
  - 2.3.2 Element B . . . 9

3 Problem 6-62 **10**
- 3.1 Given . . . 10
- 3.2 Find . . . 10
- 3.3 Solution . . . 10
  - 3.3.1 Part A . . . 10
  - 3.3.2 Part B . . . 11

1 Problem 3-83

1.1 Given



A gear reduction unit uses the countershaft shown in the figure above. Gear A receives power from another gear with a transmitted force F_A applied at the 20° pressure angle as shown. The power is transmitted through the shaft and delivered through gear B through a transmitted force F_B at the pressure angle shown.

1.2 Find

a. Determine the force F_B, assuming the shaft is running at a constant speed.
b. Find the bearing reaction forces, assuming the bearings act as simple supports.
c. Draw a shear-force and bending moment diagrams for the shaft. If needed, make one set for the horizontal plane and another set for the vertical plane.
d. At the point of maximum bending moment, determine the bending stress and the torsional shear stress.
e. At the point of maximum bending moment, determine the principal stresses and the maximum shear stress.

1.3 Solution

1.3.1 Part A and Part B

F_B and the reaction forces may be found by summing the forces and taking the moments about the point O.

[2]:
```
# Using matrices takes less thinking and more letting python do the work.
Oy, Oz, Cy, Cz, F_B = sp.symbols('O_y O_z C_y C_z F_B')
zero = sp.ZeroMatrix(3, 1)
F_A_mat = sp.Matrix([0, 300*sp.cos(sp.rad(20)), -300*sp.sin(sp.rad(20))])
F_B_mat = sp.Matrix([0, -F_B*sp.sin(sp.rad(20)), F_B*sp.cos(sp.rad(20))])
O = sp.Matrix([0, Oy, Oz])
C = sp.Matrix([0, Cy, Cz])

# Sum forces
sys1 = sp.Eq(O + F_A_mat + F_B_mat + C, zero)

# Sum moments about O
rA = sp.Matrix([16, 0, 10])
rC = sp.Matrix([16 + 14, 0, 0])
rB = sp.Matrix([16 + 14 + 9, 4, 0])
sys2 = sp.Eq(rA.cross(F_A_mat) + rC.cross(C) + rB.cross(F_B_mat), zero)

sol = sp.solve([sys1, sys2], dict=True)[0]
for key, value in sol.items():
    display(sp.Eq(key, value.n()))  # lbf
```

$C_y = 183.118820416782$

$C_z = -861.477082334154$

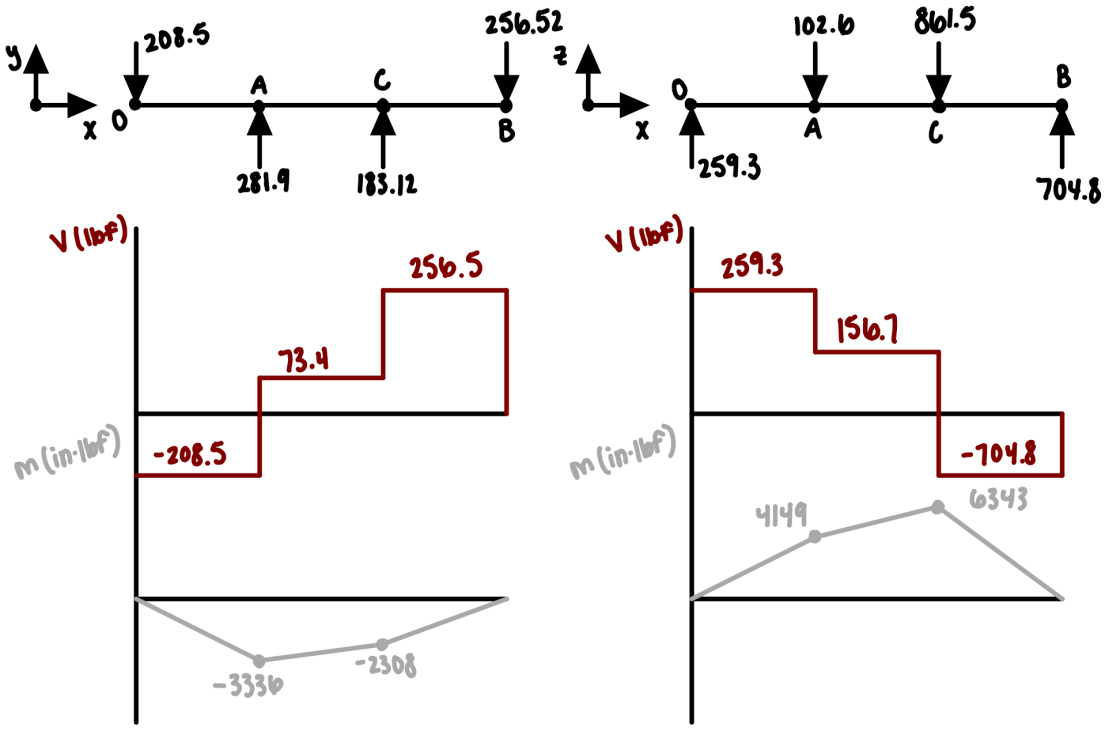
$F_B = 750.0$

$O_y = -208.511499158303$

$O_z = 259.313659742423$

The reaction forces above assume all positive directions.

1.3.2 Part C



1.3.3 Part D

The magnitudes of the moment at section A and section C must be compared.

[3]:
```
M_A = sp.sqrt(sp.S('3336.18399')**2 + sp.S('4149.01856')**2)
M_C = sp.sqrt(sp.S('6342.92519')**2 + sp.S('2308.63597')**2)
display(M_A, M_C)
```

5323.95328927262

6750.00000058625

[4]:
```
# M_C is the greater value
M_C = sp.S(6750)  # in*lbf
d = sp.S('1.25')
sig = M_C*32/(sp.pi*d**3)  # psi
T = 300*sp.cos(sp.rad(20))*10  # static torque
tau = T*16/(sp.pi*d**3)  # psi
display(sig.n(), tau.n())
```

35202.5269328378

7351.01217595663

1.3.4 Part E

$$\sigma_1 = \frac{\sigma_x}{2} + \frac{\sigma_y}{2} + \sqrt{\tau_{xy}^2 + \left(\frac{\sigma_x}{2} - \frac{\sigma_y}{2}\right)^2} = \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 35203.0 + \sqrt{7351.0^2 + \left(\left(-\frac{1}{2}\right) 0 + \frac{1}{2} \cdot 35203.0\right)^2} = 36675.875$$

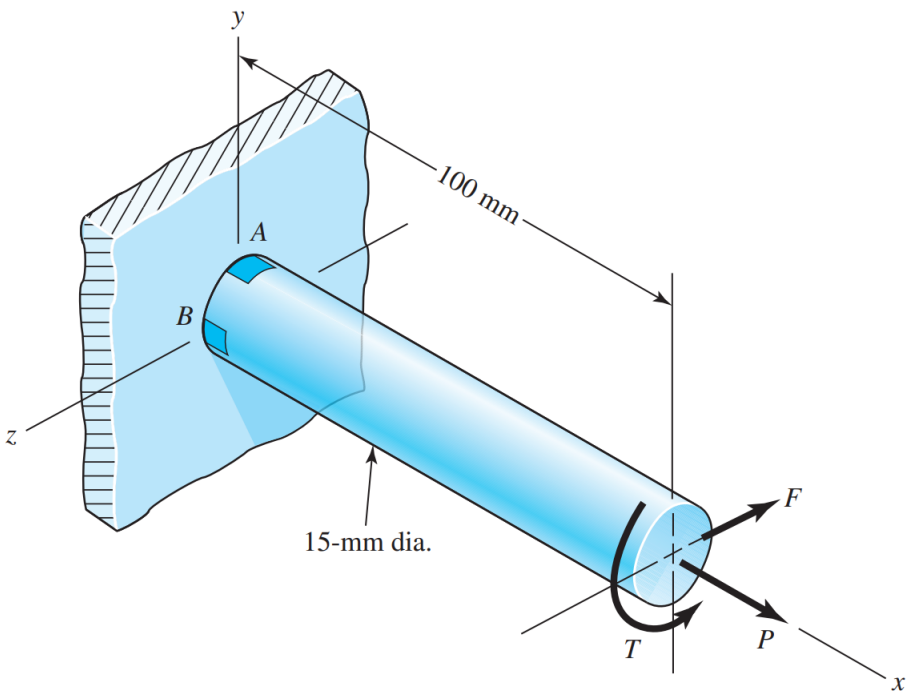
$$\sigma_2 = \frac{\sigma_x}{2} + \frac{\sigma_y}{2} - \sqrt{\tau_{xy}^2 + \left(\frac{\sigma_x}{2} - \frac{\sigma_y}{2}\right)^2} = -\sqrt{7351.0^2 + \left(\left(-\frac{1}{2}\right) 0 + \frac{1}{2} \cdot 35203.0\right)^2} + \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 35203.0 = -1473.375$$

$$\tau_1 = \sqrt{\tau_{xy}^2 + \left(\frac{\sigma_x}{2} - \frac{\sigma_y}{2}\right)^2} = \sqrt{7351.0^2 + \left(\left(-\frac{1}{2}\right) 0 + \frac{1}{2} \cdot 35203.0\right)^2} = 19074.625$$

$$\tau_2 = -\sqrt{\tau_{xy}^2 + \left(\frac{\sigma_x}{2} - \frac{\sigma_y}{2}\right)^2} = -\sqrt{7351.0^2 + \left(\left(-\frac{1}{2}\right) 0 + \frac{1}{2} \cdot 35203.0\right)^2} = -19074.625$$

2 Problem 5-36

2.1 Given



The figure above illustrates that the factor of safety for a machine element depends on the particular point selected for analysis. The bar is made of AISI 1006 cold-drawn steel and is loaded by the forces $F = 0.55\,kN$, $P = 4.0\,kN$, and $T = 25\,N \cdot m$.

2.2 Find

Find the factor of safety for both elements using the distortion energy method.

2.3 Solution

The yield strength is $S_y = 280\,MPa$.

2.3.1 Element A

[6]:
```
P = sp.S(4_000)
Sy = sp.S(280e6)
T = sp.S(25)
d = sp.S('0.015')
F = sp.S('0.55e3')
l = sp.S('0.1')
```

```
sig_x = 4*P/(sp.pi*d**2)
tau_xy = 16*T/(sp.pi*d**3) - 4*sp.S('0.55e3')/(3*(sp.pi/4)*(sp.S('0.015')**2))
display(sig_x.n(), tau_xy.n())  # Pa
```

22635369.6841807

33575798.364868

[7]:
```
sig_prime = von_mises(sig_x, 0, 0, tau_xy, 0, 0)
sig_prime.n()  # Pa
```

[7]:
62404828.8857068

[8]:
```
(Sy/sig_prime).n()
```

[8]:
4.48683226922735

2.3.2 Element B

[9]:
```
sig_x = 32*F*l/(sp.pi*d**3) + 4*P/(sp.pi*d**2)
tau_xy = 16*T/(sp.pi*d**3)
display(sig_x.n(), tau_xy.n())  # Pa
```

188628080.701506

37725616.1403011

[10]:
```
sig_prime = von_mises(sig_x, 0, 0, tau_xy, 0, 0)
sig_prime.n()  # Pa
```

[10]:
199625196.727842

[11]:
```
(Sy/sig_prime).n()
```

[11]:
1.40262854884865

3 Problem 6-62

3.1 Given

The material properties of a machine part are $S_{ut} = 85\,ksi$, $f = 0.86$, and a fully corrected endurance limit of $S_e = 45\,ksi$. The part is to be cycled at $\sigma_a = 35\,ksi$ and $\sigma_m = 30\,ksi$ for $12 \cdot 10^3$ cycles.

3.2 Find

Using the Goodman criterion, estimate the new endurance limit after cycling.

a. Use Miner's method.
b. Use Manson's method.

3.3 Solution

3.3.1 Part A

[12]:
```
Se = sp.S(45)
Sut = sp.S(85)
sig_a = sp.S(35)
sig_m = sp.S(30)
N = sp.S(12_000)
# f = sp.S('1.06') - sp.S('2.8e-3')*Sut + sp.S('6.9e-6')*Sut**2
f = sp.S('0.86')

sig_ar = sig_a/(1 - (sig_m/Sut))  # 6-59
sig_ar.n()  # ksi
```

[12]:
54.0909090909091

[13]:
```
a = (f*Sut)**2/Se
b = -sp.Rational(1, 3)*log10(f*Sut/Se)
display(a, b.n())
```

118.746888888889

-0.0702349543941723

The total number of cycles needs to be calculated, then the remaining number of cycles may be found.

[14]:
```
N1 = (sig_ar/a)**(1/b)
N1.n()
```

[14]:
72815.2378713139

[15]:
```
N_remaining = N1 - N
N_remaining.n()
```

[15]:
60815.2378713139

[16]:
```
a_prime = sig_ar/(N_remaining**b)
Se_prime = a_prime*1e6**b
Se_prime.n()  # ksi
```

[16]:
44.4344130331323

3.3.2 Part B

See p. 355.

[17]:
```
Sf = f*Sut
b_prime = log10(Sf/sig_ar)/log10(1e3/N_remaining)
a_prime = sig_ar/(N_remaining**b_prime)
Se_prime = a_prime*1e6**b_prime
Se_prime.n()  # ksi
```

[17]:
44.0529868187135