

Shaft Design Project Code

Machine Design MAE 4300

Professor Larry Gardner

Tyler Black

Daniel Estes

Jonah Frederick

Mason Frederick

Michael Fretz

Travis Slade

Due: 28 January 2025

1 Code

```
1 from math import sqrt, pi
2
3
4 def get_inputs():
5     """
6     Prompts the user to input the moment, torque, and diameter of the shaft.
7     Also allows selection of a stress concentration case.
8
9     Returns:
10        m (float): Bending moment in kip-in (kilo pound-inch).
11        t (float): Torque in kip-in.
12        d (float): Shaft diameter in inches.
13        case (int): Selected stress concentration case (1-4).
14    """
15    m = float(input("Input moment (lbf*in): ")) / 1000 # Convert to kip-in
16    t = float(input("Input torque (lbf*in): ")) / 1000 # Convert to kip-in
17    d = float(input("Input diameter (in): ")) # Input shaft diameter in
18    inches
19
20    def select_case():
21        """
22        Displays the available stress concentration cases and gets user input.
23
24        Returns:
25            int: Selected case number (1-4).
26        """
```

```
26     print(""" Stress Concentration Cases:
27     1. Wide Radius
28     2. Sharp Radius
29     3. Keyway
30     4. R.R.G""")
31
32     case = int(input("Select Case: "))
33     return case if case in {1, 2, 3, 4} else select_case()
34
35     case = select_case()
36     return m, t, d, case
37
38
39 def get_kf_kfs(d, case):
40     """
41     Computes the fatigue stress concentration factors (kf and kfs) based on
42     the selected case.
43
44     Args:
45         d (float): Shaft diameter in inches.
46         case (int): Selected stress concentration case.
47
48     Returns:
49         kf (float): Fatigue stress concentration factor for bending.
50         kfs (float): Fatigue stress concentration factor for torsion.
51     """
52     a = 0.009601465 # Notch sensitivity coefficient for bending
53     a_s = 0.00538003 # Notch sensitivity coefficient for shear
54
55     # Stress concentration factors (k, ks) and notch radii (r) based on case
56     scf = {
57         1: (1.7, 1.5, 0.1 * d),
58         2: (2.7, 2.2, 0.02 * d),
59         3: (2.14, 3, 0.02 * d),
60         4: (5, 3, 0.01 * d)
61     }
62
63     k, ks, r = scf[case]
64     r = max(r, 1e-9) # Prevent division by zero
65
66     # Calculate fatigue stress concentration factors
67     kf = 1 + (k - 1) / (1 + sqrt(a) / sqrt(r))
68     kfs = 1 + (ks - 1) / (1 + sqrt(a_s) / sqrt(r))
69
70     return kf, kfs
71
72 def fatigue_sf(m, t, d, case):
73     """
74     Computes the fatigue safety factor for the shaft.
75
76     Args:
77         m (float): Bending moment in kip-in.
78         t (float): Torque in kip-in.
```

```
79         d (float): Shaft diameter in inches.
80         case (int): Selected stress concentration case.
81
82     Returns:
83         n_f (float): Fatigue safety factor.
84     """
85     uts = 68 # Ultimate tensile strength in ksi (kilo pounds per square inch)
86     kf, kfs = get_kf_kfs(d, case)
87
88     # Calculate bending and torsional stress components
89     sigma_b = 2 * kf * m # Bending stress component
90     tau = sqrt(3) * kfs * t # Torsional stress component
91
92     # Compute endurance limit (Se) using size factor correction
93     Se_prime = uts / 2 # Initial endurance limit assumption
94     Se = 2 * uts ** (-0.217) * 0.879 * d ** (-0.107) * Se_prime # Modified
    endurance limit
95
96     # Compute fatigue safety factor
97     n_f = (pi * d ** 3 / 16) / ((sigma_b / Se) + (tau / uts))
98
99     return n_f
100
101
102 def yield_sf(m, t, d, case):
103     """
104     Computes the yield safety factor using von Mises stress.
105
106     Args:
107         m (float): Bending moment in kip-in.
108         t (float): Torque in kip-in.
109         d (float): Shaft diameter in inches.
110         case (int): Selected stress concentration case.
111
112     Returns:
113         n_y (float): Yield safety factor.
114     """
115     sy = 37.5 # Yield strength in ksi
116     kf, kfs = get_kf_kfs(d, case)
117
118     # Compute bending and torsional stresses
119     sigma = 32 * m * kf / (pi * d ** 3) # Bending stress in ksi
120     tau = 16 * t * kfs / (pi * d ** 3) # Torsional stress in ksi
121
122     # Compute von Mises equivalent stress
123     sigma_vm = sqrt(sigma ** 2 + 3 * tau ** 2)
124
125     # Compute yield safety factor
126     n_y = sy / sigma_vm
127
128     return n_y
129
130 def con_yield_sf(m, t, d, case):
131     """
```

```
132     Computes the yield safety factor a quick conservative check.
133
134     Args:
135         m (float): Bending moment in kip-in.
136         t (float): Torque in kip-in.
137         d (float): Shaft diameter in inches.
138         case (int): Selected stress concentration case.
139
140     Returns:
141         n_y (float): Yield safety factor.
142     """
143     sy = 37.5 # Yield strength in ksi
144     kf, kfs = get_kf_kfs(d, case)
145
146     # Compute bending and torsional stresses
147     sigma_a = 32 * m * kf / (pi * d ** 3)
148     sigma_m = sqrt(3) * 16 * t * kfs / (pi * d ** 3)
149
150     # Compute yield safety factor
151     n_y_c = sy / (sigma_a + sigma_m)
152
153     return n_y_c
154
155
156 def calc_diameter(m, t, case):
157     """
158     Computes the required shaft diameter to meet a target safety factor of
159     1.5.
160
161     Args:
162         m (float): Bending moment in kip-in.
163         t (float): Torque in kip-in.
164         case (int): Selected stress concentration case.
165
166     Returns:
167         d (float): Recommended shaft diameter in inches.
168     """
169
170     def f(d):
171         """Returns the fatigue safety factor for a given diameter."""
172         return fatigue_sf(m, t, d, case)
173
174     def g(d):
175         """Returns the yield safety factor for a given diameter."""
176         return con_yield_sf(m, t, d, case)
177
178     d = 0.01 # Initial guess for shaft diameter in inches
179     n = 1.5 # Target safety factor
180
181     # Iteratively increase diameter until safety factor conditions are met
182     while f(d) < n or g(d) < n:
183         d += 0.001 # Increment diameter in small steps
184
185     return d
```

```
185
186
187 if __name__ == '__main__':
188     while True:
189         # Get user input
190         m, t, d, case = get_inputs()
191
192         # Compute safety factors
193         n_f = fatigue_sf(m, t, d, case)
194         n_y = yield_sf(m, t, d, case)
195         n_y_c = con_yield_sf(m, t, d, case)
196         d_rec = calc_diameter(m, t, case)
197
198         # Print results
199         print(f"Fatigue Safety Factor: {n_f:.3f}")
200         print(f"Yield Safety Factor: {n_y:.3f}")
201         print(f"Conservative Yield Safety Factor: {n_y_c:.3f}")
202         print(f"Recommended Shaft Diameter (for SF=1.5): {d_rec:.3f} in")
```

shaft_design.py