I. Shafts

A) Input Shaft:

The first step involves identifying reaction forces using free body diagram analysis as shown in Figure 1 below. Note the figure assumes all reaction forces to be in the same direction as the datum co-ordinates

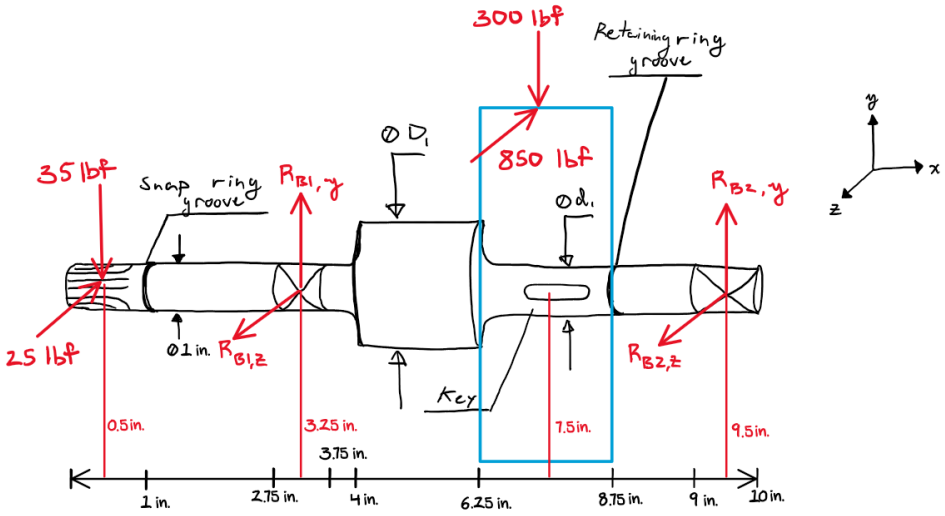

Fig 1. Free Body Diagram of Input shaft

Now we use equations of motion to satisfy the conditions for static equilibrium

$$(1)\ \sum F_y = -35 + R_{B1,y} - 300 + R_{B2,y} = 0$$

$$(2)\ \sum F_z = -25 + R_{B1,z} - 850 + R_{B2,z} = 0$$

$$(3)\ \sum \vec{M}_{k,x=0.5in} = R_{B1,y}(2.75) - 300(7.00) + R_{B2,y}(9.00) = 0$$

$$(4)\ \sum \vec{M}_{x,x=0.5in} = -R_{B1,z}(2.75) + 850(7.00) - R_{B2,z}(9.00) = 0$$

Now, we have four equations of motions and four unknowns (use linear algebra to solve)

$$\begin{matrix} R_{B1,y} & R_{B2,y} & R_{B1,z} & R_{B2,y} \end{matrix}$$

$$\begin{pmatrix} 1.00 & 1.00 & 0.00 & 0.00 & 335 \\ 0.00 & 0.00 & 1.00 & 1.00 & 875 \\ 2.75 & 9.00 & 0.00 & 0.00 & 2100 \\ 0.00 & 0.00 & 2.75 & 9.00 & 5950 \end{pmatrix} \sim \begin{pmatrix} 1.00 & 0.00 & 0.00 & 0.00 & 146.40 \\ 0.00 & 1.00 & 0.00 & 0.00 & 188.60 \\ 0.0 & 0.0 & 1.00 & 0.00 & 324.00 \\ 0.00 & 0.00 & 0.0 & 1.00 & 551.00 \end{pmatrix}$$

$$\vec{R}_{B1} = 146.40\hat{j} + 308.00\hat{k}\ lbf$$

$$\vec{R}_{B2} = 188.60\hat{j} + 567.00\hat{k}\ lbf$$

Now, using these reaction forces, we need to draw the shear diagrams in both y and z directions to determine the bending moment maxima and minima. This can be seen in Figures 2,3, and 4

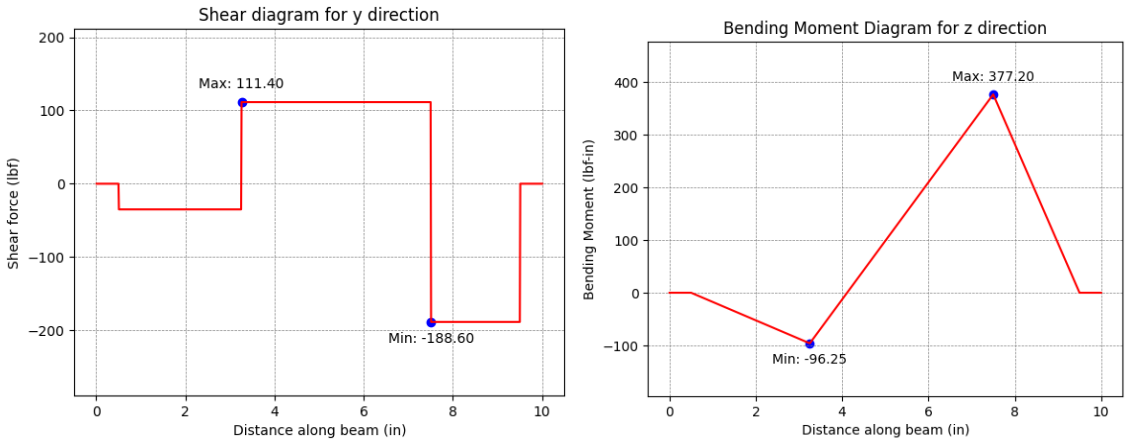

Fig 2. Shear Force in Y direction (V_y) and associated Bending Moment (M_z)

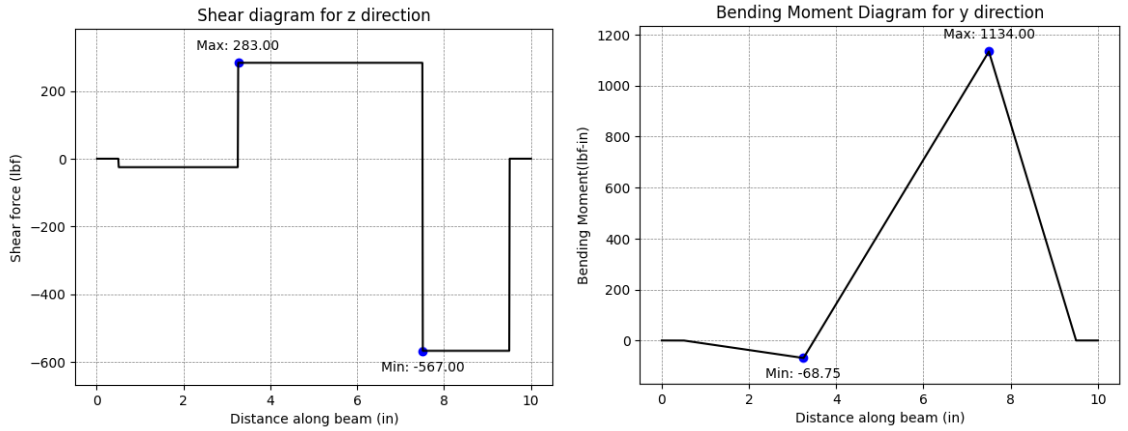

Fig 3. Shear Force in Z direction (V_z) and associated Bending Moment (M_y)

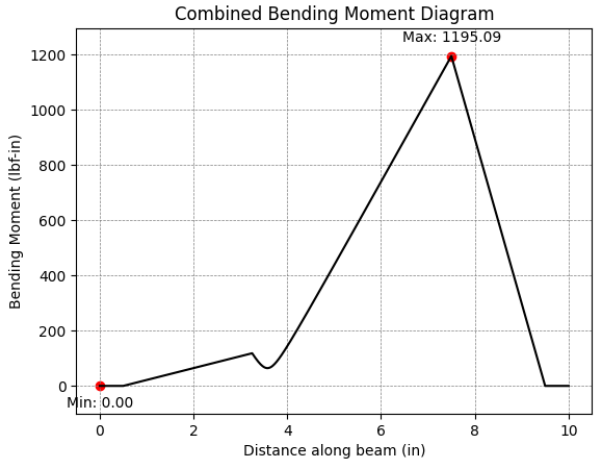

Fig 4. Combined Bending Moment Diagram

The torque is a constant value following equation (5) below for all points before 7.5 in

$$(5)\ T = r_p * F = 850 * 1.6\ in = 1360\ lbf$$

This gives the Following Torque Plot

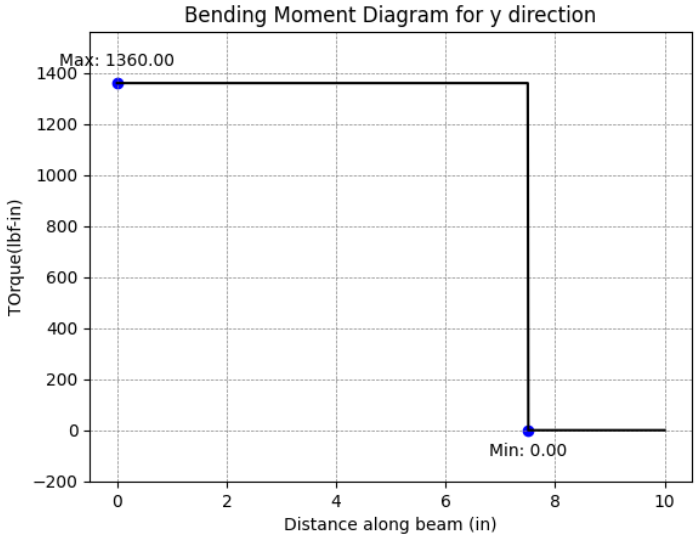

Fig 5. Torque Diagram[1]

From this we can calculate the amplitude factor of the loading and the mean loading for both Torque and Moments (M_a, M_m, T_a, T_m) at the points of interest: 1in, 4in,6.25in,7.5in,8.75in

$$(eq\ 6-8)\quad M_a = \frac{|M_{max} - M_{min}\,|}{2}, M_m = \frac{M_{max} + M_{min}}{2}$$

$$(eq\ 6-8)\ \ T_a = \frac{|T_{max} - T_{min}\,|}{2}, T_m = \frac{T_{max} + T_{min}}{2}$$

Since the Torque is steady and acts at all points it is the following at all points

$$(eq\ 6-8)\quad T_a = 0, T_m = T_{max}$$

Since the Moment is fully reversed loading cycle

$$(eq\ 6-8)\ \ M(x)_m = M(x), M(x)_m = 0$$

Table 1 shows the values amplitude and mean values at each point.

[1] There was quiet a lot of confusion regarding this but from FBD analysis there is no torque applied elsewhere that counters the torque caused by the gear so I went with this - Professor Lorenz recommended this as well

Table 1. Amplitude and Mean Coefficient for All Loadings

x_loc(in)	M (lbf-in)	Ma (lbf-in)	Mm (lbf-in)	Ta (lbf-in)	Tm (lbf-in)
1.00	21.51	21.51	0.00	0.00	1360.00
4.00	144.06	144.06	0.00	0.00	1360.00
6.25	815.73	815.73	0.00	0.00	1360.00
7.50	1195.09	1195.09	0.00	0.00	1360.00
8.75	448.16	448.16	0.00	0.00	1360.00

These can then be converted to nominal stresses. The conversion is shown in the equations below

$$(eq\ 3-26a)\ \ \sigma_{0,bend} = \frac{My}{I} = \frac{32M}{\pi d^3}$$

$$(where\ y = \frac{d}{2}\ and\ I = \frac{\pi}{64}d^4)$$

$$(eq\ 3-36)\tau_{0,bend} = \frac{T\rho}{J} = \frac{16T}{\pi d^3}$$

$$(where\ \rho = \frac{d}{2}\ and\ J = \frac{\pi}{32}d^4)$$

It is crucial to take into consideration the stress concentration factor into each point as well. From the Tables in Shigley's 11th edition, we can obtain K_t and K_{ts}values at each point. However, to convert them to fatigue coefficients we need to find q and q_s from Shigley's 11th edition as well. Then, we can use the equations below to convert them to Fatigue stress concentration factors

$$(eq\ 6-32)\ \ K_f = 1 + q(K_t - 1),\ \ K_{fs} = 1 + q_s(K_{ts} - 1)$$

After this is found, we can calculate a more accurate representation of the stress at stress raisers by multiplying the nominal stresses by the found coefficients. Doing so we get the following values. It is important to note that the main ratio $\frac{r}{d}$, is assumed to be 0.02 for the snap cover, groove, and end-mill key seat. The design method to tackle this problem is an iterative approach. To start with, let us assume d_1=1.5 in and $D_1 = 2.0\ in$. I chose to use AISI 1020 CD steel for the shaft as it is a cheap and strong material for the purpose. The constants were procured from **Figures A-15-9, A-15-8, A-15-14, A-15-16, 6-26,6-27** and Table **7-1.** This can be seen in Table 2 below – The calculations are attached in Python in the appendix

Table 2. Stresses and Fatigue Stress Concentration Factors

x_loc(in)	Siga(psi)	Sigm (psi)	Taua (psi)	Taum (psi)	Kt	Kts	q	qs	Kf	Kfs
1.00	64.91	0.00	0.00	2052.27	2.4	1.6	0.66	0.72	1.924	1.432
4.00	434.78	0.00	0.00	2052.27	2.8	2.21	0.66	0.72	2.188	1.8712
6.25	2461.90	0.00	0.00	2052.27	2.5	1.98	0.66	0.72	1.99	1.7056
7.50	3606.84	0.00	0.00	2052.27	2.14	3.0	0.66	0.72	1.7524	2.44
8.75	1352.56	0.00	0.00	0.00	2.4	1.6	0.66	0.72	1.924	1.432

Next, we must condense the amplitude stresses and the mean stresses to their von misses' stresses without considering axial force. This can be done using equation shown below (Calculations in Python)

$$(eq\ 6-66)\ \sigma_a' = \sqrt{\left(K_f * \sigma_{a0,bend}\right)^2 + 3\left(K_{fs} * \tau_{a0,bend}\right)^2}$$

$$(eq\ 6-67)\ \sigma_m' = \sqrt{\left(K_f * \sigma_{m0,bend}\right)^2 + 3\left(K_{fs} * \tau_{m0,bend}\right)^2}$$

Applying this to the data we get Table 3

Table 3. Von Misses Amplitude and Mean Stress

x_loc(in)	sigm_vm(psi)	siga_vm(psi)
1.00	5090.25	124.88
4.00	6651.45	951.30
6.25	6062.80	4899.19
7.50	8673.33	6320.62
8.75	0.00	2602.33

Now we need to determine the material properties before we can calculate the factor of safety for infinite life and first cycle yielding

We Know the following from Table A-20 that

$$S_{ut} = 68\ kpsi\ , S_y = 57 kpsi$$

$$(eq\ 6-10)\ \ Se' = 0.5 S_{ut} = 34\ kpsi$$

Now onto Marin Factors: Tables 6-2 were used to get the constant and the following equations were used

$$(eq\ 6-18)\ \ k_a = (2.00)(68)^{-0.217} = 0.800$$

$$(eq\ 6-19)\ k_b = 0.879(1.5)^{-1.07} = 0.842$$

$$(eq\ 6-25, assumption, Table\ 6-4)\ \ k_c = k_d = k_e = 1$$

The last three were all one as we had combined loading, 50% reliability and normal operating temperatures. Note, since the diameter changes the Endurance strength will also change accordingly. Applying the following we get Table 4

Table 4. Material Strengths at Each X location

x_loc(in)	Sut(kpsi)	Se(kpsi)	Sy(kpsi)
1.00	68.00	22.90	57.00
4.00	68.00	22.90	57.00
6.25	68.00	22.90	57.00
7.50	68.00	22.90	57.00
8.75	68.00	22.90	57.00

Now we can use the goodman criteria to determine the factor of safety at each location for infinite life and for first cycle yielding using the equations below

$$(eq\ 6-41)\quad n_f = \left(\frac{\sigma'_m}{S_{ut}} + \frac{\sigma'_a}{S_e}\right)^{-1}$$

$$(eq\ 6-43)\quad n_y = \frac{S_y}{\sqrt{{\sigma'_m}^2 + {\sigma'_a}^2}}$$

Applying the following we get Table 5:

Table 5: Factors of Safety at Critical x locations

x_loc(in)	nf	ny
1.00	12.4	11.1
4.00	7.1	8.4
6.25	3.3	7.3
7.50	2.4	5.3
8.75	8.8	21.9

Finally, from McMaster car, Heavy Duty External Retaining Rings 18-8 stainless steel internal retaining rings for 1.5’’ is a good choice. It satisfies the r/d ratio of 0.02 inches and ensured an interference fit between the two objects. The Mc-Master part number is 91650A515.

B) Output Shaft:

The same steps were followed in the Output Shaft to determine loading and stresses

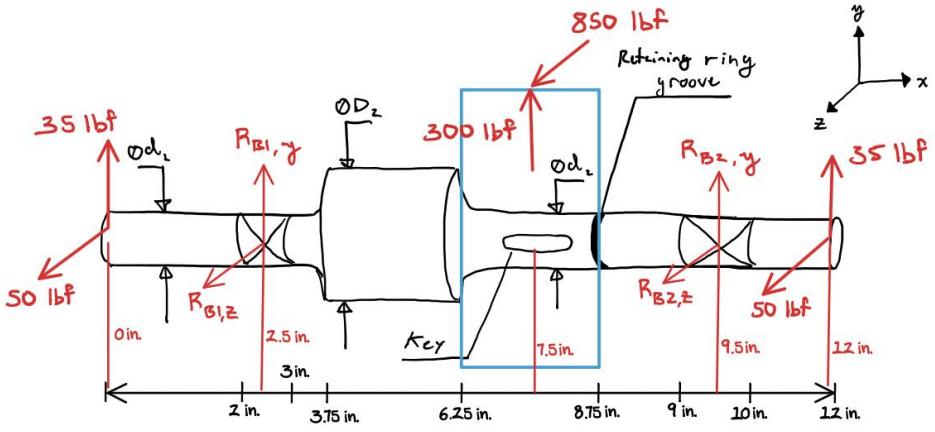

Fig 5. Free Body Diagram of Output shaft

$$(6)\ \sum F_y = 35 + R_{B1,y} + 300 + R_{B2,y} + 35 = 0$$

$$(7)\ \sum F_z = 50 + R_{B1,z} + 850 + R_{B2,z} + 50 = 0$$

$$(8)\ \sum \vec{M}_{k,x=0.0in} = R_{B1,y}(2.5) + 300(7.50) + R_{B2,y}(9.00) + 35(12) = 0$$

$$(9)\ \sum \vec{M}_{x,x=0.0in} = -R_{B1,z}(2.5) - 850(7.50) - R_{B2,z}(9.50) - 50(12) = 0$$

$$\begin{matrix} R_{B1,y} & R_{B2,y} & R_{B1,z} & R_{B2,z} \end{matrix}$$

$$\begin{pmatrix} 1.00 & 1.00 & 0.00 & 0.00 & -370 \\ 0.00 & 0.00 & 1.00 & 1.00 & -950 \\ 2.50 & 9.50 & 0.00 & 0.00 & -2670 \\ 0.00 & 0.00 & 2.50 & 9.50 & -6975 \end{pmatrix} \sim \begin{pmatrix} 1.00 & 0.00 & 0.00 & 0.00 & -845/7 \\ 0.00 & 1.00 & 0.00 & 0.00 & -1745/7 \\ 0.0 & 0.0 & 1.00 & 0.00 & -2050/7 \\ 0.00 & 0.00 & 0.0 & 1.00 & -4600/7 \end{pmatrix}$$

$$\vec{R}_{B1} = -120.71\hat{j} - 292.86\hat{k}\ lbf$$

$$\vec{R}_{B2} = -249.29\hat{j} - 657.14\ \hat{k}\ lbf$$

$$(10)\ T = r_g * F = 850 * 6.4\ in = 5440\ lbf$$

Now, using these reaction forces, we need to draw the shear diagrams in both y and z directions to determine the bending moment maxima and minima. This can be seen in Figures 6,7, and 8

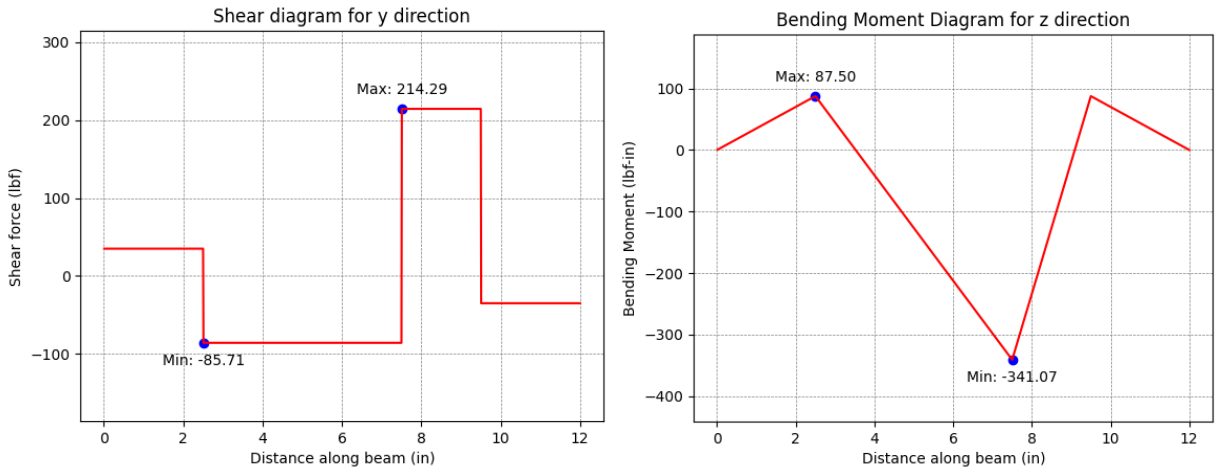

Fig 6. Shear Force in Y direction (V_y) and associated Bending Moment (M_z)

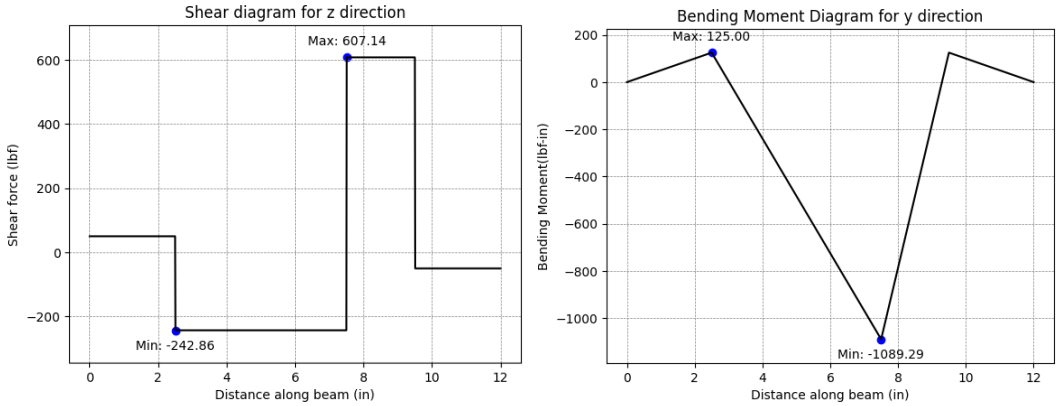

Fig 7. Shear Force in Z direction (V_z) and associated Bending Moment (M_y)

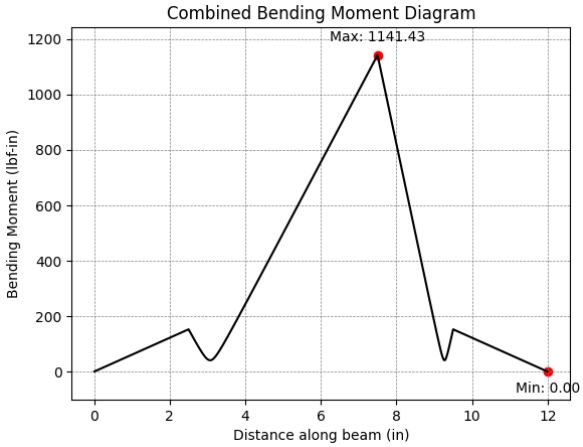

Fig 8. Combined Bending Moment Diagram

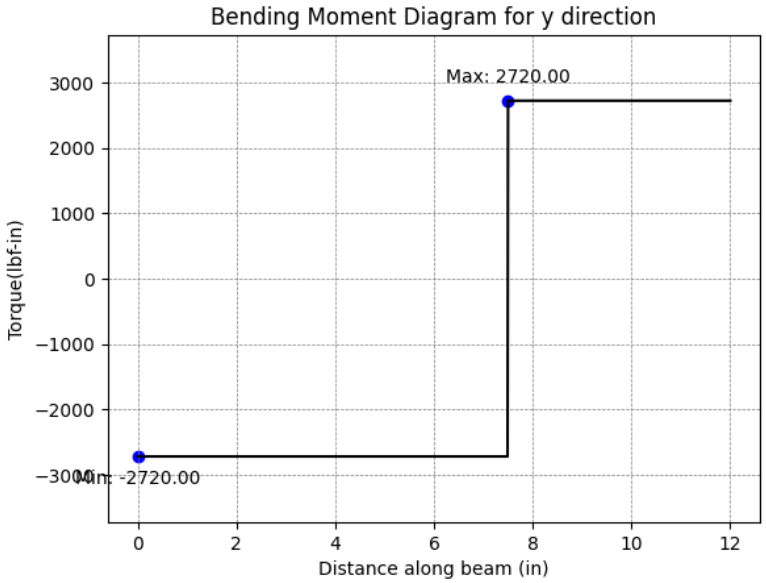

Fig 9. Torque Diagram[2]

Since the Torque is steady and there are wheels on either on either end of the output shaft

$$T_a = 0, |T_m| = \frac{T_{max}}{2}$$

Since the Moment is fully reversed loading cycle

$$M(x)_a = M(x), M(x)_m = 0$$

Table 6 shows the values at each point.

Table 6. Amplitude and Mean Coefficient for All Loadings

x_loc(in)	M (lbf-in)	Ma (lbf-in)	Mm (lbf-in)	Ta (lbf-in)	Tm (lbf-in)
3.75	179.65	179.65	0.00	0.00	-2720.00
6.25	819.80	819.80	0.00	0.00	-2720.00
7.50	1141.43	1141.43	0.00	0.00	2720.00
8.75	338.37	338.37	0.00	0.00	2720.00

To start with, let us assume d_2=2.0 in and $D_2 = 2.5\ in$. I chose to use AISI 1020 CD for the output shaft as well as it would be easier to order two of the same material in bulk and less expensive. The constants were procured from **Figures A-15-9, A-15-8, A-15-14, A-15-16, 6-26,6-27** and Table **7-1** From Shigley's Textbook**.** This can be seen in Table 7 below – The calculations are attached in Python in the appendix

[2] Same reason as foot note one.

Table 7. Stresses and Fatigue Stress Concentration Factors

x_loc(in)	Siga(psi)	Sigm (psi)	Taua (psi)	Taum (psi)	Kt	Kts	q	qs	Kf	Kfs
3.75	228.74	0.00	0.00	-1731.61	2.50	1.85	0.66	0.72	1.99	1.61
6.25	1043.80	0.00	0.00	-1731.61	2.50	1.85	0.66	0.72	1.99	1.61
7.50	1453.32	0.00	0.00	1731.61	2.14	3.00	0.66	0.72	1.75	2.44
8.75	430.83	0.00	0.00	1731.61	2.45	1.75	0.66	0.72	1.96	1.54

Table 8. Von Misses Amplitude and Mean Stress

x_loc(in)	sigm_vm(psi)	siga_vm(psi)
3.75	4834.76	455.18
6.25	4834.76	2077.16
7.50	7318.12	2546.80
8.75	4618.81	843.13

Table 9. Material Strengths at Each X location

x_loc(in)	Sut(kpsi)	Se(kpsi)	Sy(kpsi)
3.75	68.00	22.19	57.00
6.25	68.00	22.19	57.00
7.5	68.00	22.19	57.00
8.75	68.00	22.19	57.00

Table 10: Factors of Safety at Critical x locations

x_loc(in)	nf	ny
3.75	10.9	11.7
6.25	6.0	10.8
7.5	4.4	7.3
8.75	9.4	12.1

Hence, by following the same procedure for the output shaft as the input shaft, we have designed an output shaft that is designed for infinite life while avoiding first cycle yielding. We need a different snap ring as the Inner Diameter is different. Speared Ends External Retaining Rings can be used to satisfy the inner diameter and groove diameter requirements. The exact Mc-Master part number is 95524A152

II. Bearings

A) Input Shaft:

Recalling from section I.A), the reaction forces of the two bearings are

$$\vec{R}_{B1} = 146.40\hat{j} + 308.00\hat{k}\ lbf$$

$$\vec{R}_{B2} = 188.60\hat{j} + 567.00\hat{k}\ lbf$$

Since there is no axial forced experienced by the bearings, the equivalent force is the magnitude of the radial force. The radial force the bearings experience is

$$(11)\ F_{D,B1} = \sqrt{146.40^2 + 308.00^2} = 341.02\ lbf = 1.52\ kN$$

$$(12)\ F_{D,B2} = \sqrt{188.60^2 + 567.00^2} = 597.54\ lbf = 2.66\ kN$$

As we are building a mechanism for a gear box, let us use the traditional approach of ball bearings to reduce both contact area and wear and tear effects over a long period of time. Assuming the input shaft rotates at 4000 rev/ min and the expected life from the object is 15000 hrs, the total number of revolutions is

$$(13)\ L_D = 15000 * 60 * 4000 = 3.6 * 10^9\ revs$$

Now, the bearing life formula needs to be used to determine the required catalog dynamic life $F_R\ (or\ C_{10})$

$$(eq - 11 - 2)\ a_1 F_R (L_R)^{\frac{1}{a}} = F_D (L_D)^{\frac{1}{a}}$$

$$F_R = \frac{F_D}{a_1}\left(\frac{L_D}{L_R}\right)^{\frac{1}{a}}$$

Where a_1 is the reliability factor, F_R is the catalog force, L_R is the catalog life, a is the bearing factor, F_D is the experienced force, and L_D is the desired life. Assuming 90% reliability (a_1 = 1) and catalog life of 10^6 revs, the catalog forces for the two bearings are

$$F_{R,B1} = 23.25\ kN$$

$$F_{R,B2} = 40.737\ kN$$

We know that we need a bore diameter of 1.5 in (38.1 mm) and thickness of 1 in (25.4 mm). Since they are in metric, I went with the closest possible dimensions. Following these constraints, Timken has many possible bearings to choose. To reduce cost, the final two chosen are

$$B1 - 6208 : b = 40mm, w = 18\ mm, C_{10} = 29.50$$

$$B2 - 6408 : b = 40mm, w = 27\ mm, C_{10} = 63.70$$

Although the width for B1 is lesser than the desired 25.4 mm, 6208 reduces the cost, does not over compensate for the force, and weighs lesser than comparison to one of the only other options 6308 (which has a b = 40 mm, w= 23 mm, and a C10 of 40.70). Additionally since the bore is only slightly larger, there is no need to conduct shaft failure analysis again as it would succeed under the same loading.

B) Output Shaft:

Recalling from section I.B), the reaction forces of the two bearings are

$$\vec{R}_{B1} = -120.71\hat{\jmath} - 292.86\hat{k}\ lbf$$

$$\vec{R}_{B2} = -249.29\hat{\jmath} - 657.14\ \hat{k}\ lbf$$

Since there is no axial forced experienced by the bearings, the equivalent force is the magnitude of the radial force. The radial force the bearings experience is

$$(14)\quad F_{D,B1} = \sqrt{120.71^2 + 292.86^2} = 316.02\ lbf = 1.41\ kN$$

$$(15)\quad F_{D,B2} = \sqrt{249.29^2 + 657.14^2}\ = 702.84\ lbf = 3.12\ kN$$

As we are building a mechanism for a gear box, let us use the traditional approach of ball bearings to reduce both contact area and wear and tear effects over a long period of time. Assuming the input shaft rotates at 1000 rev/ min and the expected life from the object is 15000 hrs, the total number of revolutions is

$$(16)\quad L_D = 15000 * 60 * 1000 = 9 * 10^8\ revs$$

Assuming 90% reliability (a_1 = 1) and catalog life of 10^6 revs and using equation 21, the catalog forces for the two bearings are

$$F_{R,B1} = 13.60\ kN$$

$$F_{R,B2} = 30.18\ kN$$

We know that we need a bore diameter of 2.0 in (50.8 mm) and thickness of 1 in (25.4 mm). Since they are in metric, I went with the closest possible dimensions. Following these constraints, Timken has many possible bearings to choose. To reduce cost, the final chosen bearing for both points weas

$$B1\ \&\ B2 : 6210 - b = 50mm, w = 20mm, C_{10} = 35.00$$

Although the width for B1 and B2 is lesser than the desired 25.4 mm, 6210 reduces the cost, does not over compensate for the force, and weighs lesser than comparison to other options such as the 6310 (which has a b = 50 mm, w= 27 mm, and a C10 of 57.50). Since only a marginally smaller shaft is required and the factors of safety for the output shaft were multiples of the desired 1.5, the change required to the shaft would not cause failure.

III. Gear Train Design

To design the gear box, the first thing to determine is the gear ratio. Given the input speed is 4000 rpm, and the output seed is 1000 rpm we can use the formula below for the gear ratio

$$(eq\ 13-30)\ e = \frac{\omega_{out}}{\omega_{in}} = 0.25$$

Next, the minimum number of teeth on the pinion needed to be determined to avoid undercutting. This can be seen in the equation below

$$(eq\ 13-10)\ N_p = \frac{2\left(m + (m^2 + (1+2m)\sin^2\Phi)^{\frac{1}{2}}\right)}{(1+2m)\sin^2\Phi}$$

Where N_p is the minimum number of teeth on the pinion, $m = \frac{1}{e}$, and Φ is the pressure angle. Plugging in $\Phi = 20^o$ and rounding up to the nearest whole number, we get that $N_p = 16$. Hence it follows that N_g , the number of teeth on the gear, is 64 (to maintain gear ratio). From Section I, we know that the pitch diameter of the pinion and the gear are $d_p = 3.2\ in, d_g = 12.8\ in$ respectively. This results in a transversal pitch, $P_d = \frac{N_p}{d_p} =$ Using the equation below, we can determine the pitch line velocity of the 2-gear mesh

$$(eq\ 13-34)\ V = \frac{\pi d\omega}{12}$$

Where V is the pitch line velocity in ft/min, d is either pinion/gear diameter in inches, and ω is either the angular velocity in rpm of the pinion/gear. Substituting the relevant values, we get $V = 3351.03\ ft/min$. Furthermore, we can use the equation below to determine the tangential load acting between the mesh

$$(eq\ 13-35)\ W^t = \frac{33000H}{V}$$

Where W^t is the tangential load in lbf, and H is the power generated in Horsepower. Substituting the relevant values, we get $W^t = 837.06\ lbf$. Finally, using Table 13-3 from shigley's, the face width $F = 2.5\ in$ for both gears. Now that the key variables have been determined, we can move on to stress and factor of safety analysis.

The problem statement recommends using a quality factor $Q_v = 7$, Reliability $R = 99.9\%$, and a desired life $L = 15{,}000\ hrs$. Since we know the pinion is likely to fail first, due to having a four times larger angular velocity than the gear, if we design for the pinion to not fail, the gear will not fail either. Hence, the equivalent number of revolutions is $N = 15000\ hrs * 60\frac{min}{hrs} * 4000\frac{rev}{min} = 3.6 * 10^9\ rev$. The gear contact stress can be written as shown in equation 26 and the gear bending stress can be seen in the equations below

$$(eq\ 14-16)\ \sigma_c = C_p\left(W^t K_o K_s K_v * \frac{K_m}{F * d_p} * \frac{C_f}{I}\right)^{\frac{1}{2}}$$

$$(eq\ 14-15)\ \ \sigma = W^t K_o K_s K_v * \frac{P_d}{F} * \frac{K_m K_b}{J}$$

Each key variable and constant are listed in Table 11 below along with how the constants were found.

Table 11: Key Variables and Constants for Contact Stress and Bending Stress analysis*Check python attached in appendix for the calculations

Variable/Constant	Physical Meaning	Numeric Value	Shigley Source
σ_c	Gear Contact Stress	$\mathbf{95.14\ kpsi}$	eq 14-16
σ	Gear Bending Stress	$\mathbf{13.13\ kpsi}$	eq 14-15
C_p	Elastic Coefficient	$2300\ psi^{\frac{1}{2}}$	Table 14-8
W^t	Tangential Load	837.06	eq 13-35
K_o	Overload Factor	1.00	ANSI/AGMA standard
K_s	Size Factor	1.00	ANSI/AGMA standard
K_v	Dynamic Factor	1.55	eq 14-27
K_m	Load Distribution Factor	1.35	eq 14-30
K_b	Rim Thickness Factor	1.00	eq 14-40
F	Face Width	$2.50\ in$	table 13-3
d_p	Pinion Diameter	$3.20\ in$	Problem Statement (Term Project)
P_d	Transversal Pitch	$5.00\ teeth/in$	ANSI/AGMA Standard
C_f	Surface Condition Factor	1.00	ANSI/AGMA Standard
I	Geometry Factor	0.13	eq 14-23
J	Geometry Factor	0.27	fig 14-6

Now that the stresses have been determined, the fatigue factor of safety can be computed using the equations seen below

$$(eq\ 14-42)\ \ S_H = \frac{S_c Z_N C_H}{K_T K_R \sigma_c}$$

$$(eq\ 14-41)\ S_T = \frac{S_t Y_N}{K_T K_{R\sigma}}$$

To being with, the material chosen was Nitrided Steel, Grade 3 as it has the lowest strength hence presents the cheapest option. A table with all the key variables/constants is present below

Table 12: Key Variables and Constants for Factor of safety analysis*Check python attached in appendix for the calculations

Variable/Constant	Physical Meaning	Numeric Value	Shigley Source
S_H	Wear Factor of Safety	$\mathbf{1.3}$	eq 14-42
S_T	Bending Factor of safety	$\mathbf{4.1}$	eq 14-41
Z_N	Stress Cycle Factor	0.873	fig 14-15
Y_N	Stress Cycle Factor	0.902	fig 14-14
C_H	Hardness ratio Factor	1	section 14-12 (gear only assumption)
K_T	Temperature Factor	1	ANSI/AGMA Standard (T<250 F)
K_R	Reliability Factor	1.25	eq 14-38
S_t	Bending Strength	75 $kpsi$	table 14-4 (Used the entry in Grade 3)
S_c	Contact Strength	180 $kpsi$	table 14-6

The desired factor of safety was 1.2 as per the problem statement. From Table 12, we see that both the wear and bending factors of safety are greater than 1.2, Hence, Grade 3 Nitrided Steel gears would satisfy the design requirements

IV. Key Design

A) Input Shaft:

To design the key, we need to consider both the failure due to shear and failure due to crushing. The formulas to determine the corresponding factors of safety and stress can be seen below

$$(17)\ n_\tau = \frac{S_{sy}}{\tau} = \frac{0.577 S_y}{\tau}, \tau = \frac{T}{rwl}$$

$$(18)\ n_c = \frac{S_y}{\sigma_c}, \sigma_c = \frac{2T}{hrl}$$

Where n_τ is the shear factor of safety, n_c is the crushing factor of safety, S_y is the yield strength, τ is the shear stress, σ_c is the crushing stress, r is the radius of the shaft, w is the key width, l is the key length, and h is the key height. From part I.A) we know that the r = 0.75 in (1.5/2). To simplify the design, a square key will be used. According to table 7-6 from Shigley's, a 1.5 in shaft should have square key dimensions $w = h = \frac{3}{8}\ in$. From the force analysis in part I.A) we know the torque at the point is $T = 1360\ lbf$. The only unknown left to solve for is the l. From the problem statement we are given that the factor of safety to meet is 1.5 and the $S_y = 54\ kpsi$. The equations above can be rearranged and solved as follows

$$l \geq \frac{T n_\tau}{0.577 S_y r w} \leftrightarrow l \geq\ 0.233\ in$$

$$l \geq \frac{2 T n_c}{S_y r h} \leftrightarrow l \geq\ 0.267\ in$$

To satisfy both inequalities let $l \geq 0.267\ in$. Hence the final dimensions and the strength of the key on the input shaft are

$$w = 0.375\ in, h = 0.375\ in, l = 0.275 in$$

B) Output Shaft:

The exact same process was used for the output shaft as well. From part I.B) we know that the r = 1.0 in (2/2). To simplify the design, a square key will be used. According to table 7-6 from Shigley's, a 2.0 in shaft should have square key dimensions $w = h = \frac{1}{2}\ in$. From the force analysis in part I.B) we know the torque at the point is $T = 5440\ lbf$. The only unknown left to solve for is the l. From the problem statement we are given that the factor of safety to meet is 1.5 and the $S_y = 54\ kpsi$. The equations in part IV.A) remain the same as and can be solved as shown below

$$l \geq \frac{Tn_\tau}{0.577 S_y r w} \leftrightarrow l \geq\ 0.524\ in$$

$$l \geq \frac{2Tn_c}{S_y r h} \leftrightarrow l \geq\ 0.604\ in$$

To satisfy both inequalities let $l \geq 0.604\ in$. Hence the final dimensions and the strength of the key on the input shaft are

$$w = 0.500\ in, h = 0.500\ in, l = 0.625 in$$

V. APPENDIX (CODE)

A) Input Shaft

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
#Define constants
RINy = -35.0 #@ x=0.5 in
RINz = -25.0 #@ x=0.5 in

RB1y = 146.4 #@ x=3.25 in
RB1z = 308.0 #@ x=3.25 in

W12y = -300 #@ x=7.5 in
W12z = -850 #@ x=7.5 in

RB2y = 188.6 #@ x=9.5 in
RB2z = 567.0 #@ x=9.5 in

# Create a range of x values from 0 to 10 in increments of 0.01
x_inches = x_values = np.arange(0, 10.01, 0.01)
# Create the DataFrame with x in inches
df = pd.DataFrame({'x_inches': x_inches})


def setShear(x,dir):
    if 0.00<x<=0.50:
        return 0
    elif 0.50<x<=3.25:
        return RINy if dir=='y' else RINz
    elif 3.25<x<=7.5:
        return RINy+RB1y if dir=='y' else RINz+RB1z
    elif 7.5<x<=9.5:
        return RINy+RB1y+W12y if dir=='y' else RINz+RB1z+W12z
    elif x>=9.5:
        return RINy+RB1y+ W12y+RB2y if dir=='y' else RINz+RB1z+W12z+RB2z
def setMoment(x,dir):
    if dir == 'z':
        if 0.00<=x<=0.50:
            return 0
        elif 0.50<=x<=3.25:
            return RINy*(x-0.5)
        elif 3.25<=x<=7.5:
            return RINy*(3.25-0.5) + (RINy+RB1y)*(x-3.25)
        elif 7.5<=x<=9.5:
```

```
            return RINy*(3.25-0.5) + (RINy+RB1y)*(7.5-3.25) +
(RINy+RB1y+W12y)*(x-7.5)
        elif x>=9.5:
            return RINy*(3.25-0.5) + (RINy+RB1y)*(7.5-3.25) +
(RINy+RB1y+W12y)*(9.5-7.5) + (RINy+RB1y+W12y+RB2y)*(x-9.5)
    else :
        if 0.00<=x<=0.50:
            return 0
        elif 0.50<=x<=3.25:
            return RINz*(x-0.5)
        elif 3.25<=x<=7.5:
            return RINz*(3.25-0.5) + (RINz+RB1z)*(x-3.25)
        elif 7.5<=x<=9.5:
            return RINz*(3.25-0.5) + (RINz+RB1z)*(7.5-3.25) +
(RINz+RB1z+W12z)*(x-7.5)
        elif x>=9.5:
            return RINz*(3.25-0.5) + (RINz+RB1z)*(7.5-3.25) +
(RINz+RB1z+W12z)*(9.5-7.5) + (RINz+RB1z+W12z+RB2z)*(x-9.5)
def setTorque(x):
    if 0.00<=x<=7.5:
        return 1360
    else:
        return 0.0

df['Vy'] = df['x_inches'].apply(setShear, dir='y')
df['Vz'] = df['x_inches'].apply(setShear, dir='z')
df['Mz'] = df['x_inches'].apply(setMoment, dir='z')
df['My'] = df['x_inches'].apply(setMoment, dir='y')
df['T'] = df['x_inches'].apply(setTorque)

# Find maxima and minima
max_Vy = df.loc[df['Vy'].idxmax()]
min_Vy = df.loc[df['Vy'].idxmin()]
max_Mz = df.loc[df['Mz'].idxmax()]
min_Mz = df.loc[df['Mz'].idxmin()]
max_Vz = df.loc[df['Vz'].idxmax()]
min_Vz = df.loc[df['Vz'].idxmin()]
max_My = df.loc[df['My'].idxmax()]
min_My = df.loc[df['My'].idxmin()]
max_T = df.loc[df['T'].idxmax()]
min_T = df.loc[df['T'].idxmin()]

# Plot Shear for y direction
plt.plot(df['x_inches'], df['Vy'], color='red')
```

```
plt.scatter([max_Vy['x_inches'], min_Vy['x_inches']], [max_Vy['Vy'],
min_Vy['Vy']], color='blue')  # Add max and min points
plt.annotate(f'Max: {max_Vy["Vy"]:.2f}', (max_Vy['x_inches'], max_Vy['Vy']),
textcoords="offset points", xytext=(0, 10), ha='center')
plt.annotate(f'Min: {min_Vy["Vy"]:.2f}', (min_Vy['x_inches'], min_Vy['Vy']),
textcoords="offset points", xytext=(0, -15), ha='center')
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Shear force (lbf)')
# Adjust the y-axis limits for Shear for y direction
plt.ylim(df['Vy'].min() - 100, df['Vy'].max() + 100)
plt.title('Shear diagram for y direction')
plt.show()

# Plot Bending Moment for z direction
plt.plot(df['x_inches'], df['Mz'], color='red')
plt.scatter([max_Mz['x_inches'], min_Mz['x_inches']], [max_Mz['Mz'],
min_Mz['Mz']], color='blue')  # Add max and min points
plt.annotate(f'Max: {max_Mz["Mz"]:.2f}', (max_Mz['x_inches'], max_Mz['Mz']),
textcoords="offset points", xytext=(0, 10), ha='center')
plt.annotate(f'Min: {min_Mz["Mz"]:.2f}', (min_Mz['x_inches'], min_Mz['Mz']),
textcoords="offset points", xytext=(0, -15), ha='center')
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Bending Moment (lbf-in)')
# Adjust the y-axis limits for Bending Moment for z direction
plt.ylim(df['Mz'].min() - 100, df['Mz'].max() + 100)
plt.title('Bending Moment Diagram for z direction')
plt.show()

# Plot Shear for z direction
plt.plot(df['x_inches'], df['Vz'], color='black')
plt.scatter([max_Vz['x_inches'], min_Vz['x_inches']], [max_Vz['Vz'],
min_Vz['Vz']], color='blue')  # Add max and min points
plt.annotate(f'Max: {max_Vz["Vz"]:.2f}', (max_Vz['x_inches'], max_Vz['Vz']),
textcoords="offset points", xytext=(0, 10), ha='center')
plt.annotate(f'Min: {min_Vz["Vz"]:.2f}', (min_Vz['x_inches'], min_Vz['Vz']),
textcoords="offset points", xytext=(0, -15), ha='center')
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Shear force (lbf)')
# Adjust the y-axis limits for Shear for z direction
plt.ylim(df['Vz'].min() - 100, df['Vz'].max() + 100)
plt.title('Shear diagram for z direction')
plt.show()
```

```
# Plot Bending for y moment
plt.plot(df['x_inches'], df['My'], color='black')
plt.scatter([max_My['x_inches'], min_My['x_inches']], [max_My['My'],
min_My['My']], color='blue')  # Add max and min points
plt.annotate(f'Max: {max_My["My"]:.2f}', (max_My['x_inches'], max_My['My']),
textcoords="offset points", xytext=(0, 10), ha='center')
plt.annotate(f'Min: {min_My["My"]:.2f}', (min_My['x_inches'], min_My['My']),
textcoords="offset points", xytext=(0, -15), ha='center')
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Bending Moment(lbf-in)')
# Adjust the y-axis limits for Torque
plt.ylim(df['My'].min() - 100, df['My'].max() + 100)
plt.title('Bending Moment Diagram for y direction')
plt.show()

#Plot Combine moment
df['Mcombined'] = ((df['My'])**2 + (df['Mz'])**2)**0.5
# Find maxima and minima for combined bending moment
max_Mcombined = df.loc[df['Mcombined'].idxmax()]
min_Mcombined = df.loc[df['Mcombined'].idxmin()]
# Plot Combined Bending Moment
plt.plot(df['x_inches'], df['Mcombined'], color='black')
# Add max and min points
plt.scatter([max_Mcombined['x_inches'], min_Mcombined['x_inches']],
[max_Mcombined['Mcombined'], min_Mcombined['Mcombined']], color='red')
# Annotate max and min points
plt.annotate(f'Max: {max_Mcombined["Mcombined"]:.2f}',
(max_Mcombined['x_inches'], max_Mcombined['Mcombined']), textcoords="offset
points", xytext=(0, 10), ha='center', color='black')
plt.annotate(f'Min: {min_Mcombined["Mcombined"]:.2f}',
(min_Mcombined['x_inches'], min_Mcombined['Mcombined']), textcoords="offset
points", xytext=(0, -15), ha='center', color='black')
# Other plot settings
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Bending Moment (lbf-in)')
# Adjust the y-axis limits for Combined Bending Moment
plt.ylim(df['Mcombined'].min() - 100, df['Mcombined'].max() + 100)
plt.title('Combined Bending Moment Diagram')
plt.show()

# Plot Bending for Torque
plt.plot(df['x_inches'], df['T'], color='black')
```

```
plt.scatter([max_T['x_inches'], min_T['x_inches']], [max_T['T'], min_T['T']],
color='blue')  # Add max and min points
plt.annotate(f'Max: {max_T["T"]:.2f}', (max_T['x_inches'], max_T['T']),
textcoords="offset points", xytext=(0, 10), ha='center')
plt.annotate(f'Min: {min_T["T"]:.2f}', (min_T['x_inches'], min_T['T']),
textcoords="offset points", xytext=(0, -15), ha='center')
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('TOrque(lbf-in)')
# Adjust the y-axis limits for Torque
plt.ylim(-200, df['T'].max() + 200)
plt.title('Bending Moment Diagram for y direction')
plt.show()


import math
df2 = pd.DataFrame({'x_loc': np.array([1.0, 4.0, 6.25, 7.5, 8.75])})
df2['M'] = df2['x_loc'].apply(lambda x_ref: df.loc[df['x_inches'] == x_ref,
'Mcombined'].iloc[0])
df2['Ma'] = df2['M']
df2['Mm'] = df2['M']-df2['M']
df2['Ta'] = np.zeros_like(df2['x_loc'])
df2['Tm']= np.array([1360.0,1360.0,1360.0,1360.0,0.0])


d1 =1.5
dfsigma = pd.DataFrame({'x_loc': np.array([1.0, 4.0, 6.25, 7.5, 8.75])})
dfsigma['d'] = np.array([d1, d1, d1, d1, d1])
dfsigma['siga'] = (32/(math.pi*dfsigma['d']**3))*df2['Ma']
dfsigma['sigm'] = (32/(math.pi*dfsigma['d']**3))*df2['Mm']
dfsigma['taua'] = (16/(math.pi*dfsigma['d']**3))*df2['Ta']
dfsigma['taum'] = (16/(math.pi*dfsigma['d']**3))*df2['Tm']
dfsigma['Kt']=np.array([2.4,2.8,2.5,2.14,2.4])
dfsigma['Kts']=np.array([1.6,2.21,1.98,3.00,1.6])
dfsigma['q']=np.ones_like(df2['x_loc'])*0.66
dfsigma['qs']=np.ones_like(df2['x_loc'])*0.72
dfsigma['Kf']= 1 + dfsigma['q']*(dfsigma['Kt']-1)
dfsigma['Kfs']= 1 + dfsigma['qs']*(dfsigma['Kts']-1)
dfsigma['sigm_vm'] = (((dfsigma['sigm'])*dfsigma['Kf'])**2 +
3*(dfsigma['taum']*dfsigma['Kfs'])**2)**(0.5)
dfsigma['siga_vm'] = (((dfsigma['siga'])*dfsigma['Kf'])**2 +
3*(dfsigma['taua']*dfsigma['Kfs'])**2)**(0.5)
dfsigma.to_excel('partAP2.xlsx')

dfMat= pd.DataFrame(df2['x_loc'])
```

```
dfMat['Sut']=np.ones_like(df2['x_loc'])*68
dfMat['SeExp']=dfMat['Sut']/2
dfMat['kb'] =  np.array ([0.842,0.842,0.842,0.842,0.842])
dfMat['Se']= dfMat['SeExp']*0.800*dfMat['kb']
dfMat['nf'] = ((dfsigma['sigm_vm'])/(dfMat['Sut']*10**3) +
(dfsigma['siga_vm'])/(dfMat['Se']*10**3))**-1
dfMat['ny'] = (57*10**3)/(((dfsigma['sigm_vm'])**2+
(dfsigma['siga_vm'])**2)**0.5)
dfMat.to_excel('partAP3.xlsx')

RB1=[146.40,308.00]
RB2=[188.60,567.00]

FD1 = ((RB1[0])**2 + (RB1[1])**2)**0.5
FD2 = ((RB2[0])**2 + (RB2[1])**2)**0.5
print (FD1,FD2)

#Convert to kilo newtons
FD1 = FD1 * 4.44822/1000
FD2 = FD2 * 4.44822/1000
print(FD1,FD2)

t = 15000 * 60  ##Need to be changed
N = 4000 ##Needs to be changed

LR = 10**6
LD = N*t
a = 3
a1 = 1

FR1 = (FD1/a1)*((LD/LR)**(1/a))
FR2 = (FD2/a1)*((LD/LR)**(1/a))

print(FR1,FR2,LD)
T = 1360
n = 1.5
w = 3/8
h = 3/8
r = 1.5/2
Sy = 54*(10**3)

ltau = T*n/(0.577*Sy*r*w)
lc = 2*T*n/(Sy*r*h)

print (ltau,lc)
```

B) Output Shaft

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
#Define constants
R1y = 35.0 #@ x=0
R1z = 50.0 #@ x=0

B1y =  -845.0/7#@ x=2.50 in
B1z =  -2050.0/7#@ x=2.50 in

W12y = 300 #@ x=7.5 in
W12z = 850 #@ x=7.5 in

B2y =  -1745.0/7 #@9.50 in
B2z =  -4600.0/7 #@9.50 in

R2y = 35.0 #@ x=12.00 in
R2z = 50.0 #@ x=12.00 in

# Create a range of x values from 0 to 10 in increments of 0.01
x_inches = x_values = np.arange(0, 12.01, 0.01)
# Create the DataFrame with x in inches
df = pd.DataFrame({'x_inches': x_inches})

def setShear(x,dir):
    if 0.00<x<=2.50:
        return R1y if dir == 'y' else R1z
    elif 2.50<=x<=7.5:
        return R1y+B1y if dir=='y' else R1z+B1z
    elif 7.5<=x<=9.5:
        return R1y+B1y+W12y if dir=='y' else R1z+B1z+W12z
    elif x>=9.5:
        return R1y+B1y+W12y+B2y if dir=='y' else R1z+B1z+W12z+B2z
def setMoment(x,dir):
    if dir == 'z':
        if 0.00<x<=2.50:
            return R1y*x
        elif 2.50<x<=7.5:
            return R1y*(2.5) + (R1y+B1y)*(x-2.5)
        elif 7.5<x<=9.5:
            return R1y*(2.5) + (R1y+B1y)*(7.5-2.5) + (R1y+B1y+W12y)*(x-7.5)
        elif x>=9.5:
```

```
            return R1y*(2.5) + (R1y+B1y)*(7.5-2.5) + (R1y+B1y+W12y)*(9.5-7.5)+
(R1y+B1y+W12y+B2y)*(x-9.5)
    else :
        if 0.00<=x<=2.50:
            return R1z*x
        elif 2.50<=x<=7.5:
            return R1z*(2.5) + (R1z+B1z)*(x-2.5)
        elif 7.5<=x<=9.5:
            return R1z*(2.5) + (R1z+B1z)*(7.5-2.5) + (R1z+B1z+W12z)*(x-7.5)
        elif x>=9.5:
            return R1z*(2.5) + (R1z+B1z)*(7.5-2.5) + (R1z+B1z+W12z)*(9.5-7.5)+
(R1z+B1z+W12z+B2z)*(x-9.5)

def setTorque(x):
    if 0.00<=x<7.5:
        return -2720
    else:
        return 2720

df['Vy'] = df['x_inches'].apply(setShear, dir='y')
df['Vz'] = df['x_inches'].apply(setShear, dir='z')
df['Mz'] = df['x_inches'].apply(setMoment, dir='z')
df['My'] = df['x_inches'].apply(setMoment, dir='y')
df['T'] = df['x_inches'].apply(setTorque)

# Find maxima and minima
max_Vy = df.loc[df['Vy'].idxmax()]
min_Vy = df.loc[df['Vy'].idxmin()]
max_Mz = df.loc[df['Mz'].idxmax()]
min_Mz = df.loc[df['Mz'].idxmin()]
max_Vz = df.loc[df['Vz'].idxmax()]
min_Vz = df.loc[df['Vz'].idxmin()]
max_My = df.loc[df['My'].idxmax()]
min_My = df.loc[df['My'].idxmin()]
max_T = df.loc[df['T'].idxmax()]
min_T = df.loc[df['T'].idxmin()]

# Plot Shear for y direction
plt.plot(df['x_inches'], df['Vy'], color='red')
plt.scatter([max_Vy['x_inches'], min_Vy['x_inches']], [max_Vy['Vy'],
min_Vy['Vy']], color='blue')  # Add max and min points
plt.annotate(f'Max: {max_Vy["Vy"]:.2f}', (max_Vy['x_inches'], max_Vy['Vy']),
textcoords="offset points", xytext=(0, 10), ha='center')
plt.annotate(f'Min: {min_Vy["Vy"]:.2f}', (min_Vy['x_inches'], min_Vy['Vy']),
textcoords="offset points", xytext=(0, -15), ha='center')
```

```
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Shear force (lbf)')
# Adjust the y-axis limits for Shear for y direction
plt.ylim(df['Vy'].min() - 100, df['Vy'].max() + 100)
plt.title('Shear diagram for y direction')
plt.show()

# Plot Bending Moment for z direction
plt.plot(df['x_inches'], df['Mz'], color='red')
plt.scatter([max_Mz['x_inches'], min_Mz['x_inches']], [max_Mz['Mz'],
min_Mz['Mz']], color='blue')  # Add max and min points
plt.annotate(f'Max: {max_Mz["Mz"]:.2f}', (max_Mz['x_inches'], max_Mz['Mz']),
textcoords="offset points", xytext=(0, 10), ha='center')
plt.annotate(f'Min: {min_Mz["Mz"]:.2f}', (min_Mz['x_inches'], min_Mz['Mz']),
textcoords="offset points", xytext=(0, -15), ha='center')
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Bending Moment (lbf-in)')
# Adjust the y-axis limits for Bending Moment for z direction
plt.ylim(df['Mz'].min() - 100, df['Mz'].max() + 100)
plt.title('Bending Moment Diagram for z direction')
plt.show()

# Plot Shear for z direction
plt.plot(df['x_inches'], df['Vz'], color='black')
plt.scatter([max_Vz['x_inches'], min_Vz['x_inches']], [max_Vz['Vz'],
min_Vz['Vz']], color='blue')  # Add max and min points
plt.annotate(f'Max: {max_Vz["Vz"]:.2f}', (max_Vz['x_inches'], max_Vz['Vz']),
textcoords="offset points", xytext=(0, 10), ha='center')
plt.annotate(f'Min: {min_Vz["Vz"]:.2f}', (min_Vz['x_inches'], min_Vz['Vz']),
textcoords="offset points", xytext=(0, -15), ha='center')
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Shear force (lbf)')
# Adjust the y-axis limits for Shear for z direction
plt.ylim(df['Vz'].min() - 100, df['Vz'].max() + 100)
plt.title('Shear diagram for z direction')
plt.show()

# Plot Bending for y moment
plt.plot(df['x_inches'], df['My'], color='black')
plt.scatter([max_My['x_inches'], min_My['x_inches']], [max_My['My'],
min_My['My']], color='blue')  # Add max and min points
```

```
plt.annotate(f'Max: {max_My["My"]:.2f}', (max_My['x_inches'], max_My['My']),
textcoords="offset points", xytext=(0, 10), ha='center')
plt.annotate(f'Min: {min_My["My"]:.2f}', (min_My['x_inches'], min_My['My']),
textcoords="offset points", xytext=(0, -15), ha='center')
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Bending Moment(lbf-in)')
# Adjust the y-axis limits for Torque
plt.ylim(df['My'].min() - 100, df['My'].max() + 100)
plt.title('Bending Moment Diagram for y direction')
plt.show()

#Plot Combine moment
df['Mcombined'] = ((df['My'])**2 + (df['Mz'])**2)**0.5
# Find maxima and minima for combined bending moment
max_Mcombined = df.loc[df['Mcombined'].idxmax()]
min_Mcombined = df.loc[df['Mcombined'].idxmin()]
# Plot Combined Bending Moment
plt.plot(df['x_inches'], df['Mcombined'], color='black')
# Add max and min points
plt.scatter([max_Mcombined['x_inches'], min_Mcombined['x_inches']],
[max_Mcombined['Mcombined'], min_Mcombined['Mcombined']], color='red')
# Annotate max and min points
plt.annotate(f'Max: {max_Mcombined["Mcombined"]:.2f}',
(max_Mcombined['x_inches'], max_Mcombined['Mcombined']), textcoords="offset
points", xytext=(0, 10), ha='center', color='black')
plt.annotate(f'Min: {min_Mcombined["Mcombined"]:.2f}',
(min_Mcombined['x_inches'], min_Mcombined['Mcombined']), textcoords="offset
points", xytext=(0, -15), ha='center', color='black')
# Other plot settings
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Bending Moment (lbf-in)')
# Adjust the y-axis limits for Combined Bending Moment
plt.ylim(df['Mcombined'].min() - 100, df['Mcombined'].max() + 100)
plt.title('Combined Bending Moment Diagram')
plt.show()

# Plot Bending for Torque
plt.plot(df['x_inches'], df['T'], color='black')
plt.scatter([max_T['x_inches'], min_T['x_inches']], [max_T['T'], min_T['T']],
color='blue')  # Add max and min points
plt.annotate(f'Max: {max_T["T"]:.2f}', (max_T['x_inches'], max_T['T']),
textcoords="offset points", xytext=(0, 10), ha='center')
```

```
plt.annotate(f'Min: {min_T["T"]:.2f}', (min_T['x_inches'], min_T['T']),
textcoords="offset points", xytext=(0, -15), ha='center')
plt.grid(True, color='grey', linewidth=0.5, linestyle='--')
plt.xlabel('Distance along beam (in)')
plt.ylabel('Torque(lbf-in)')
# Adjust the y-axis limits for Torque
plt.ylim(df['T'].min() - 1000, df['T'].max() + 1000)
plt.title('Bending Moment Diagram for y direction')
plt.show()

import math

df2 = pd.DataFrame({'x_loc': np.array([3.75, 6.25,7.5, 8.75])})
df2['M'] = df2['x_loc'].apply(lambda x_ref: df.loc[df['x_inches'] == x_ref,
'Mcombined'].iloc[0])
df2['Ma'] = df2['M']
df2['Mm'] = df2['M']-df2['M']
df2['Ta'] = np.zeros_like(df2['x_loc'])
df2['Tm']= np.array([5440,5440,5440,5440])
df2.to_excel('partBP1.xlsx')

d2=2
D2 = 2.5
dfsigma = pd.DataFrame({'x_loc': np.array([3.75, 6.25, 7.5, 8.75])})
dfsigma['d'] = np.ones_like(dfsigma['x_loc'])*d2
dfsigma['siga'] = (32/(math.pi*dfsigma['d']**3))*df2['Ma']
dfsigma['sigm'] = (32/(math.pi*dfsigma['d']**3))*df2['Mm']
dfsigma['taua'] = (16/(math.pi*dfsigma['d']**3))*df2['Ta']
dfsigma['taum'] = (16/(math.pi*dfsigma['d']**3))*df2['Tm']

dfsigma['Kt']=np.array([2.5,2.5,2.14,2.45])
dfsigma['Kts']=np.array([1.85,1.85,3.00,1.75])
dfsigma['q']=np.ones_like(df2['x_loc'])*0.66
dfsigma['qs']=np.ones_like(df2['x_loc'])*0.72
dfsigma['Kf']= 1 + dfsigma['q']*(dfsigma['Kt']-1)
dfsigma['Kfs']= 1 + dfsigma['qs']*(dfsigma['Kts']-1)
dfsigma['sigm_vm'] = (((dfsigma['sigm'])*dfsigma['Kf'])**2 +
3*(dfsigma['taum']*dfsigma['Kfs'])**2)**(0.5)
dfsigma['siga_vm'] = (((dfsigma['siga'])*dfsigma['Kf'])**2 +
3*(dfsigma['taua']*dfsigma['Kfs'])**2)**(0.5)
dfsigma.to_excel('partBP2.xlsx')

dfMat= pd.DataFrame(df2['x_loc'])
dfMat['Sut']=np.ones_like(df2['x_loc'])*68
dfMat['SeExp']=dfMat['Sut']/2
```

```
dfMat['kb'] =  np.ones_like(df2['x_loc'])*0.816
dfMat['Se']= dfMat['SeExp']*0.800*dfMat['kb']
dfMat['nf'] = ((dfsigma['sigm_vm'])/(dfMat['Sut']*10**3) +
(dfsigma['siga_vm'])/(dfMat['Se']*10**3))**-1
dfMat['ny'] = (57*10**3)/(((dfsigma['sigm_vm'])**2+
(dfsigma['siga_vm'])**2)**0.5)
dfMat.to_excel('partBP3.xlsx')

RB1=[-845.0/7,-2050/7]
RB2=[-1745.0/7,-4600/7]

FD1 = ((RB1[0])**2 + (RB1[1])**2)**0.5
FD2 = ((RB2[0])**2 + (RB2[1])**2)**0.5
print (FD1,FD2)

#Convert to kilo newtons
FD1 = FD1 * 4.44822/1000
FD2 = FD2 * 4.44822/1000
print(FD1,FD2)

t = 15000 * 60  ##Need to be changed
N = 1000 ##Needs to be changed

LR = 10**6
LD = N*t
a = 3
a1 = 1

FR1 = (FD1/a1)*((LD/LR)**(1/a))
FR2 = (FD2/a1)*((LD/LR)**(1/a))

print(FR1,FR2,LD)

T = 5440
n = 1.5
w = 1/2
h = 1/2
r = 2.0/2
Sy = 54*(10**3)

ltau = T*n/(0.577*Sy*r*w)
lc = 2*T*n/(Sy*r*h)

print (ltau,lc)
```

C) Gear Box

```
import pandas as pd
import numpy as np
import math

dp = 3.2
dg = 12.8
win = 4000 #rpm
wout = 1000 #rpm
H = 85
e =  wout/win #(a)
e_inv = 1/e
phi = 20*math.pi/180
print("The gear ratio is {:.2f}".format(e))

Np_min = (2/((1+2*e_inv)*(math.sin(phi))**2))*(e_inv+ (e_inv**2 +
((1+2*e_inv)*(math.sin(phi))**2))**0.5) #(b)
Ng_min = e_inv*math.ceil(Np_min)
print("The minimum number of tooth on the pinion is
{:.2f}".format(math.ceil(Np_min)))
print("It follows that the gear would has a minimum of
 {:.2f}".format(math.ceil(Ng_min)))

#Actual Values used (Table 13-3)
Np = math.ceil(Np_min)
Ng = Ng_min

Pp = Np/dp
Pg = Ng/dg

V = math.pi*dp*win/12
print("The pitch line velocity is {:.2f} ft/min".format(V)) #(c)
W = 33000*H/V
print("The tangential load is {:.2f} lbf".format(W)) #(c)

#Used (Table 13-3)
Fp = 12.5/Pp
Fg = 12.5/Pg
print("The face widths of the pinion and the gear are  {:.2f} in and {:.2f} in
respectively".format(Fp,Fg))#(d)

Qv = 7
F = 2.5
B = 0.25*(12-Qv)**(2/3)
```

```
A = 56 + 56*(1-B)
R = 0.999
L = 15000*4000*60

#Constants from tables (assume Grade 3 Nitried-steel)
Ko = 1 #ANSI uniform loading
Kv = ((A+(V)**(1/2))/A)**B #Eq 14-27
Ks = 1 #ANSI standard
Cmc = 1#uncrowned teeth (Eq 14-31)
Cpf = (F/(10*dp))-0.0375+0.0125*F # (Eq 14-32)
Cpm = 1 #Eq (14-33)
Cma = 0.247 + 0.0167*F - 0.765*(10**-4)*(F**2) #Eq 14-34 - Open gear
Ce = 1 #Eq 14-35
Km =  1 + Cmc*(Cpf*Cpm + Cma*Ce) #(Eq 14 -30)
Kb = 1.00 #Eq 14-40
Pd = Np/dp
Cf = 1
Cp = 2300 # Eq 14-13 (table 14-8)
I = math.cos(phi)*math.sin(phi)*e_inv/(2*e_inv + 2)#Eq 14-23
J = 0.268 #Fig 14-6


Sc = 180000
St = 75000 #Table 14-4
YN = 1.3358*(L**-0.0178) #Fig 14-14
ZN =1.4488*(L**-0.023) #Fig 14-15
CH = 1 #Section 14-12
KT = 1 #T<250 F
KR = 0.5 - 0.109*math.log(1-R) #Eq 14-38

#Bending
sig_c = Cp*(W*Kv*Ks*Ko*(Km/(F*dp))*(Cf/I))**(0.5)
Sh = Sc*ZN*CH/(KT*KR*sig_c)
print (sig_c,Sh)

#Wear
sig = W*Kv*Ks*Ko*(Pd/F)*(Km*Kb/J)
Sf = St*YN/(KT*KR*sig)

print(sig,Sf)
```