University of Waterloo

Department of Mechanical and Mechatronics Engineering

Designing a Gear Train That Will Not Break

ME321 – Kinematics and Dynamics of Machines

Prepared by:

Justin Borromeo, 20661006

Sean O’Sullivan, 20660254

Introduction

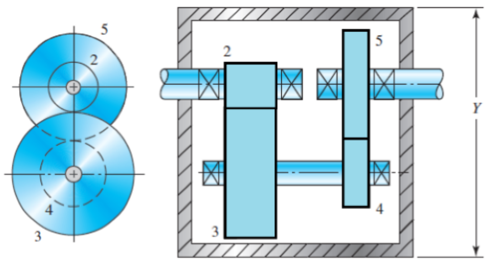
The provided problem requires the design of a double-reduction gear train (shaft, gears, and gearbox) to transmit 20 horsepower from a moderate-shock source to a moderate-shock output. This gear train must fit in a 14 x 14 x 22-inch gearbox and must reduce the 1750 rpm input speed to an output speed of between 82 and 88 rpm. The gears should be designed for a 12000-hour lifetime while the shaft should have an infinite lifetime; all components should be designed with a 99% reliability. A summary of inputs can be found in the Inputs section.

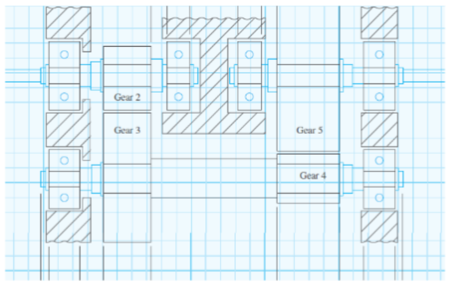
To design the gear train, the following assumptions were made:

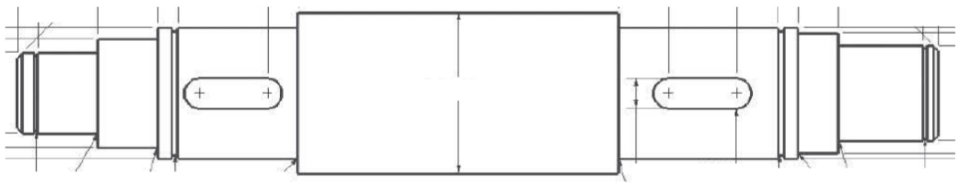
1) The gear train is used in an automotive transmission (which is appropriate given the moderate-shock input and moderate-shock output). This is relevant in the selection of an AGMA quality number in gear design.
2) A safety factor of 3 will be used throughout the analysis.
3) 20^{o} full-depth involute spur gears will be used throughout the gear train.
4) The thickness of the gearbox wall is 1-inch so each input and output shaft will have a 1-inch section where the shafts will rest against the container.

Analysis

The general gear train design (see below) provided in the project document was used. Analysis primarily consisted of material selection and solving for the measurements of the four gears and three shafts.







Inputs

Power: 20 hp

Input Speed: 1750 rpm

Output Speed: 82 – 88 rpm

Gearbox Dimensions: 14 x 14 x 22 in

Gear life: 12000 hours

Shaft life: Infinite

Reliability: 99%

Gear Design - Procedure

To calculate gear dimensions and determine a suitable material for each gearset, the following steps were followed:

1) Determine target overall velocity ratio:

$$VR_{overall,target} = \frac{input\ speed}{output\ speed}$$

2) Determine individual target gearset velocity ratio:

$$VR_{gearset,target} = \sqrt[num\ gearsets]{VR_{overall}}$$

3) Choose the number of teeth for the pinion N_P. For convenience, the minimum number of teeth for a 20°, full-depth pinion from Table 8-7; this value is 17.
4) Calculate the number of teeth on the gear:

$$N_G = round(N_P * VR_{gearset})$$

5) Check that the actual velocity ratio is within the required limits:

$$VR_{actual} = (N_G * N_P)^{num\ gearsets}$$

6) Calculate design power. $K_o = 2.00$ from Table 9-1 based on the moderate-shock input and moderate-shock output.

$$P_{des} = P * \ K_o$$

7) Using P_{des} and n_p, determine pitch diameter P_d from Figure 9-11:

$$P_d = 7.00$$

8) Calculate gear diameter D_g and pinion diameter D_p:

$$D_x = N_x / P_d$$

9) Check that the gear train height is within the maximum height established in the problem definition:

$$h = 1.5 * D_g + 0.5 * \ D_p + \frac{3}{P_d}$$

10) Calculate center distance C:

$$C = \frac{D_p + D_G}{2 * P_d}$$

For each gearset, perform steps 11 to 31

11) Calculate pitch line speed V_t:

$$v_t = \frac{\pi * P_d * n_p}{12}$$

12) Calculate transmitted load W_t:

$$W_t = \frac{33000 * P}{v_t}$$

13) Calculate nominal face width F

$$F = \frac{12}{P_d}$$

14) Using the material (steel), get the elastic coefficient C_p from Table 9-7
15) Select an AGMA quality number A_v from Table 9-5. Using the automobile transmission assumption (stated in the Introduction), an AGMA quality number of 6 should be chosen.
16) From Table 9-16, select dynamic factor K_v using the calculated tangential velocity V_t and selected AGMA quality number A_v.
17) Find the F/D_p ratio. If the ratio is greater than 2 or less than 0.5, we can't use the provided formulas.
18) Calculate pinion proportion factor C_{pf} using the formulas from Fig 9-12. If 1 < F < 15,

$$C_{pf} = \frac{F}{10 * D_p} - 0.0375 + 0.0125F$$

19) Calculate mesh alignment factor C_{ma} using the formulas from Figure 9-13. Assuming the gearset is a commercial enclosed gear unit,

$$C_{ma} = 0.127 + 0.0158F - 1.093 * 10^{-4} * F^2$$

20) Calculate K_m:

$$K_m = 1 + C_{ma} + C_{pf}$$

21) Choose a size factor K_s using Table 9-2
22) Choose a rim thickness factor for the pinion (K_{BP}) and gear (K_{BG}) using Table 9-14
23) Choose a reliability factor K_r based on the required reliability using Table 9-11.

24) Calculate number of load cycles for the pinion (N_P) and gear (N_G) using the formula

$$N = 60 * \{lifetime\ in\ hours\} * \{rotation\ speed\ in\ rpm\}$$

25) Calculate the bending stress cycle factor for the pinion (Y_{NP}) and the gear (Y_{NG}) using the formulas from Figure 9-21.

26) Calculate the pitting stress cycle factor for the pinion (Z_{NP}) and the gear (Z_{NG}) using the formulas from Figure 9-22.

27) Determine the bending geometry factor for the pinion (J_P) using Figure 9-10 and for the gear (J_G) using Figure 9-17.

28) Determine the pitting geometry factor (I) from Figure 9-17.

29) Find the required bending stress number s_{at} for the pinion and the gear using the formula:

$$s_{at} = \frac{W_t P_d}{FJ} K_o K_s K_m K_B K_v \frac{SF\, K_R}{Y_N}$$

30) Find the required pitting stress number s_{ac} for the pinion and the gear using the formula:

$$s_{ac} = C_p \sqrt{\frac{W_t K_o K_s K_m K_v}{F D_P I}} \frac{SF\, K_R}{Z_N}$$

31) Find the limiting stress by finding the maximum of all the s_{at}s and s_{ac}s.

32) Calculate Brinell Hardness using Figure 9-19 and the calculated limiting stress

33) Select a material with an appropriate Brinell Hardness

This procedure was implemented in Python (see Appendix A). For the sake of brevity, the outputted values from the Python script without intermediate calculations and equations will be shown for each gear set's design analysis.

Gear Design – Gear Set 4-5 (Input)

1) Target Velocity Ratio: 20.58823529411765

2) Target Gearset Velocity Ratio: 4.5374260648651505

3) Num Pinion Teeth: 17

4) Num Gear Teeth: 77

5) Actual Velocity Ratio: 4.529411764705882

6) Design Power [hp]: 40.0

7) Pitch Diameter [in]: 7.0

8) Gear Diameter [in]: 11.0

8) Pinion Diameter [in]: 2.4285714285714284

9) Geartrain Height [in]: 18.142857142857142

10) Center Distance [in]: 0.9591836734693878

11) Pitch Line Speed [ft/min]: 3207.0425005395805

12) Transmitted Load [lbs]: 205.79708559800997

13) Nominal Face Width [in]: 1.7142857142857142

14) Elastic Coefficient: 2300

15) AGMA Quality Number: 6

16) Dynamic Factor: 1.13

17) F/Dp Ratio: 0.7058823529411765

18) Pinion Proportion Factor: 0.054516806722689076

19) Mesh Alignment Factor: 0.153764506122449

20) Km: 1.2082813128451382

21) Ks: 1

22) Rim Thickness Factor: 1.0

23) Reliability Factor: 1.0

24) Number Pinion Cycles: 1260000000

24) Number Gear Cycles: 278181818.1818182

25) Ynp: 0.9337036012656141

25) Yng: 0.9591501034620954

26) Znp: 0.8947463074559197

26) Zng: 0.9263793727366358

27) Bending Geometry Factor Pinion: 0.295

27) Bending Geometry Factor Gear: 0.41

28) Pitting Geometry Factor: 0.105

29) Bending Stress Number Pinion [psi]: 8331.045421001683

29) Bending Stress Number Gear [psi]: 5835.258735137693

30) Contact Stress Number Pinion [psi]: 92166.71041765004

30) Contact Stress Number Gear [psi]: 41827.72303949037

31) Limiting Stress [psi]: 92166.71041765004

32) Brinell Hardness: 195.85934912313675

33) Choose 1144 Cold-Drawn Steel with HB = 200

Gear Design – Gear Set 2-3 (Output)

1) Target Velocity Ratio: 4.5374260648651505

2) Target Gearset Velocity Ratio: 4.5374260648651505

3) Num Pinion Teeth: 17

4) Num Gear Teeth: 77

5) Actual Velocity Ratio: 4.529411764705882

6) Design Power [hp]: 40.0

7) Pitch Diameter [in]: 7.0

8) Gear Diameter [in]: 11.0

8) Pinion Diameter [in]: 2.4285714285714284

9) Geartrain Height [in]: 18.142857142857142

10) Center Distance [in]: 0.9591836734693878

11) Pitch Line Speed [ft/min]: 708.0483442749724

12) Transmitted Load [lbs]: 932.1397406498098

13) Nominal Face Width [in]: 1.7142857142857142

14) Elastic Coefficient: 2300

15) AGMA Quality Number: 6

16) Dynamic Factor: 1.07

17) F/Dp Ratio: 0.7058823529411765

18) Pinion Proportion Factor: 0.054516806722689076

19) Mesh Alignment Factor: 0.153764506122449

20) Km: 1.2082813128451382

21) Ks: 1

22) Rim Thickness Factor: 1.0

23) Reliability Factor: 1.0

24) Number Pinion Cycles: 278181818.1818182

24) Number Gear Cycles: 61416765.05312869

25) Ynp: 0.9591501034620954

25) Yng: 0.985290106758024

26) Znp: 0.9263793727366358

26) Zng: 0.9591308006311069

27) Bending Geometry Factor Pinion: 0.295

27) Bending Geometry Factor Gear: 0.41

28) Pitting Geometry Factor: 0.105

29) Bending Stress Number Pinion [psi]: 34783.16493829622

29) Bending Stress Number Gear [psi]: 24362.940877775713

30) Contact Stress Number Pinion [psi]: 184356.60192996546

30) Contact Stress Number Gear [psi]: 83665.96628093896

31) Limiting Stress [psi]: 184356.60192996546

32) Brinell Hardness: 482.1633600309486

33) Choose SAE 4150 OQT 700 with HB = 495

Dividing input speed by total velocity ratio yields a gear train output speed of 85.301 rpm. This is well within the specified 82 – 88 rpm range, meaning that this gear design is acceptable from a functionality standpoint.

Shaft Design – Procedure

To calculate dimensions for each shaft, the following steps were followed:

1) Using the face widths calculated when designing the gears, select lengths for each section of the shaft.
2) Guess a suitable material. Record the material's S_n and S_y strengths.
3) Using Table 5-4, calculate C_s
4) Calculate ${S_n}'$:

$$S_n' = C_s C_r S_n$$

5) Choose a safety factor (N)
6) Calculate the torque on the shaft:

$$T = \frac{P}{\omega}$$

7) Calculate tangential force on the shaft due to each gear and pinion on the shaft:

$$F_{tG} = \frac{T}{r_G}$$

8) Calculate radial force on the shaft due to each gear and pinion on the shaft:

$$F_{rG} = F_{tG}\tan(20^o)$$

9) Calculate the reaction forces in both the tangential and radial planes.
10) Draw the shear force diagrams for both the tangential and radial planes.

11) Draw the moment diagrams for both the tangential and radial planes by integrating the shear force.

12) Multiply the moment by stress concentration factor K_t where stress is concentrated.

13) Calculate the diameter of each section using the following formula where M is the maximum K_t adjusted moment of each section:

$$D^3 = \frac{32N}{\pi}\sqrt{\left(\frac{K_t M}{s_n'}\right)^2 + \frac{3}{4}\left(\frac{T}{s_y}\right)^2}$$

14) If the shaft diameter of any of the sections is greater than the pinion/gear diameter, return to step 2 and repeat with a different material.

Shaft Design – Input Shaft

1) Section Lengths [in]: [0.5, 1, 1.964, 1, 0.25, 0.25]

2) Sn [psi]: 310000

2) Sy [psi]: 287000

3) Cs: 0.8116523277245

4) Sn' [psi]: 203805.89949162197

5) N: 3

6) Torque [lbs-in]: 6875.493541569879

7) Tangential Force Gear [lbs]: 5662.171151881077

8) Radial Force Gear [lbs]: 2060.8617606054886

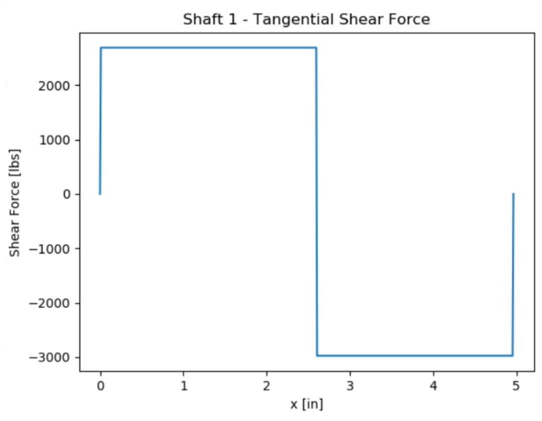
10) Tangential Reaction Force C [lbs]: 2973.666436936738

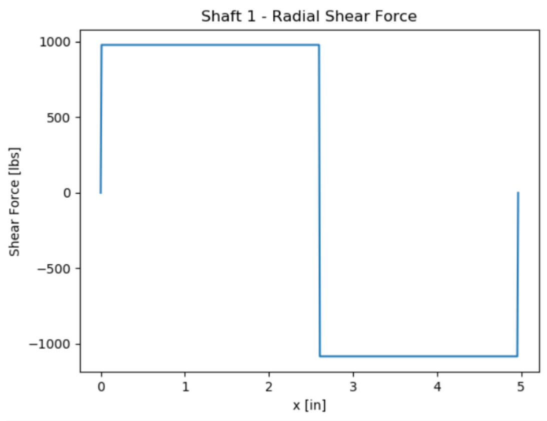
10) Tangential Reaction Force A [lbs]: 2688.504714944339

10) Radial Reaction Force C [lbs]: 1082.3260696814077

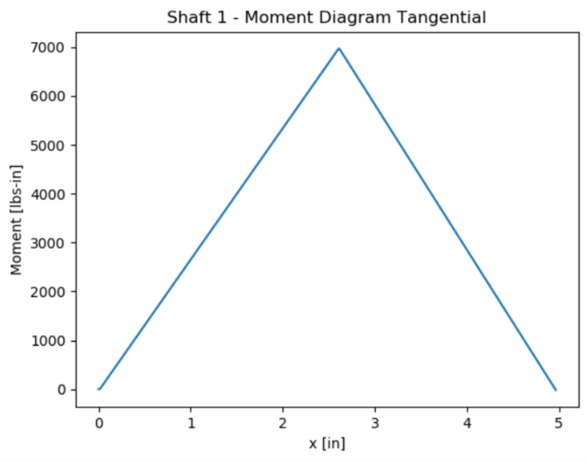
10) Radial Reaction Force A [lbs]: 978.535690924080911

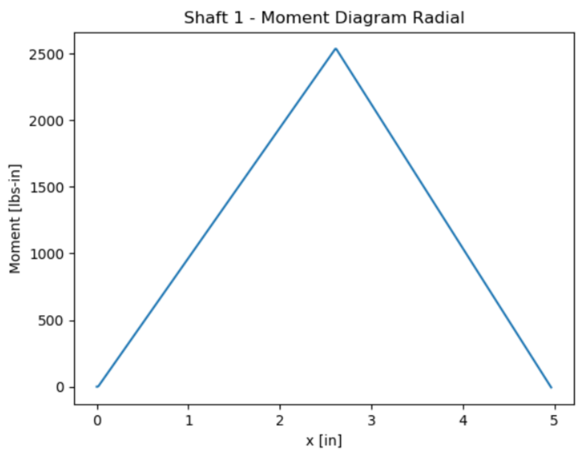
11)


Shaft 1 - Tangential Shear Force
Shear Force [lbs]
x [in]

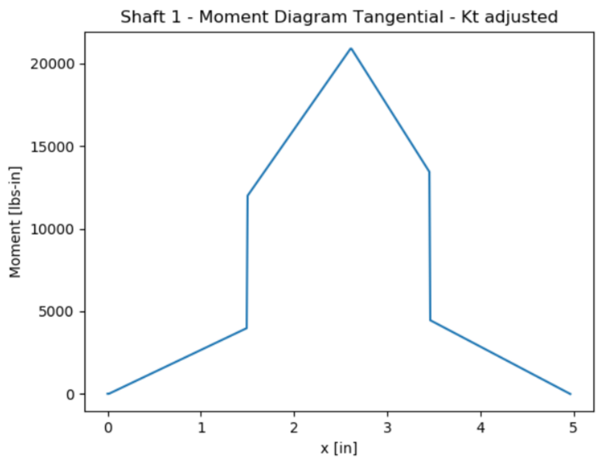

Shaft 1 - Radial Shear Force
Shear Force [lbs]
x [in]

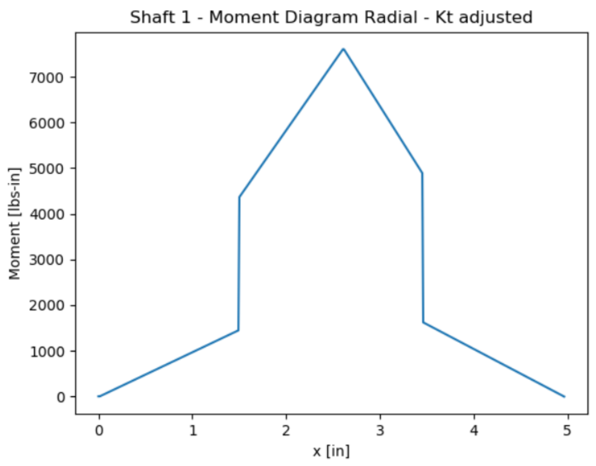
12)


Shaft 1 - Moment Diagram Tangential
Moment [lbs-in]
x [in]


Shaft 1 - Moment Diagram Radial
Moment [lbs-in]
x [in]

13)


Shaft 1 - Moment Diagram Tangential - Kt adjusted
Moment [lbs-in]
x [in]


Shaft 1 - Moment Diagram Radial - Kt adjusted
Moment [lbs-in]
x [in]

14) Section Diameters [in]: [0.23286368451684086, 0.40354024190358395, 1.180330714294978, 0.42036596721285024, 0.20024439943937306, 0.12524559101164875]

15) These diameters work with the calculated gear and pinion diameters, so recalculation is unnecessary.

Shaft Design – Output Shaft

1) Section Lengths [in]: [0.5, 1, 1.964, 1, 0.25, 0.25]

2) Sn [psi]: 310000

2) Sy [psi]: 287000

3) Cs : 0.8116523277245

4) Sn' [psi]: 203805.89949162197

5) N: 3

6) Torque [lbs-in]: 31141.94133534592

7) Tangential Force Gear [lbs]: 5662.171151881076

8) Radial Force Gear [lbs]: 2060.861760605488

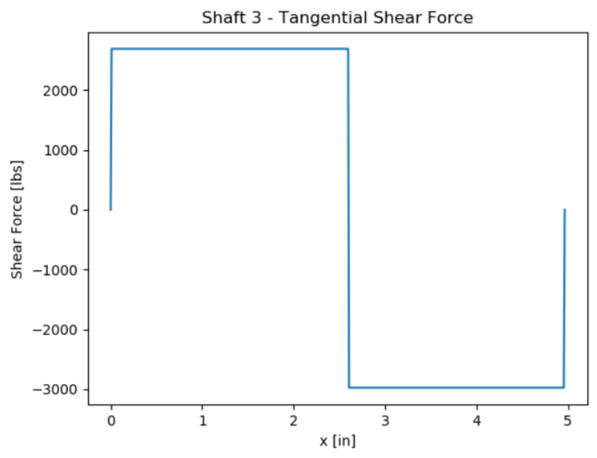
10) Tangential Reaction Force C [lbs]: 2973.6664369367377

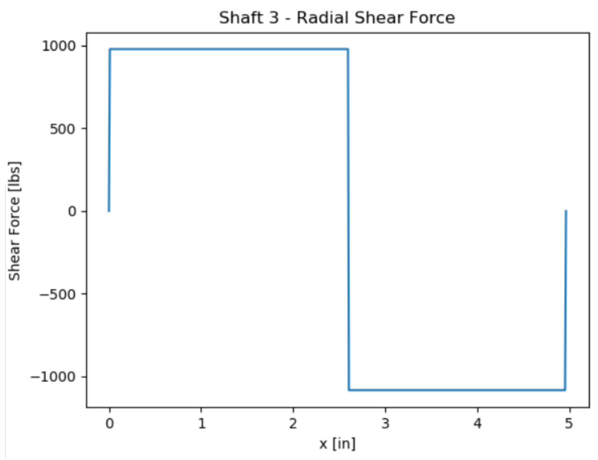
10) Tangential Reaction Force A [lbs]: 2688.5047149443385

10) Radial Reaction Force C [lbs]: 1082.3260696814077

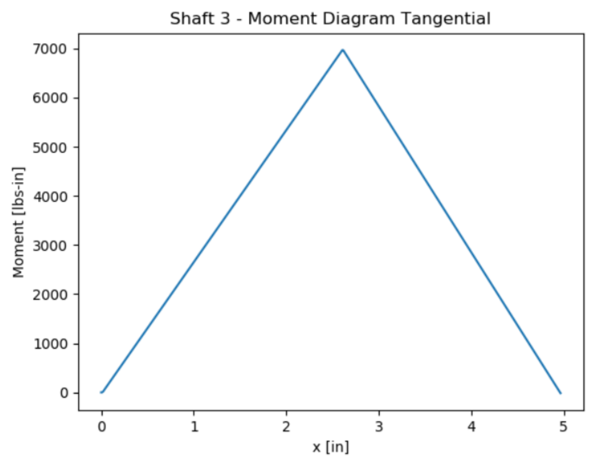
10) Radial Reaction Force A [lbs]: 978.5356909240804

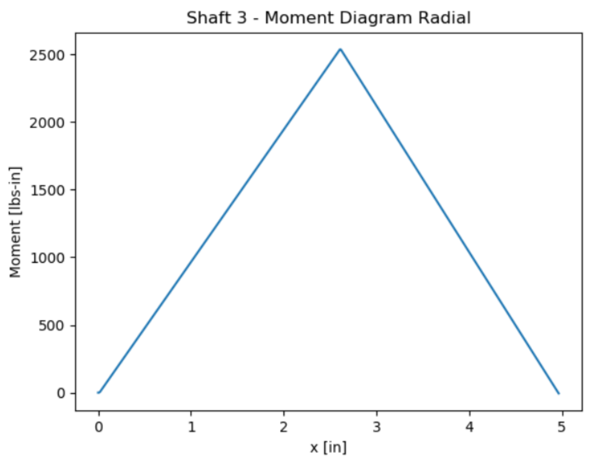
11)



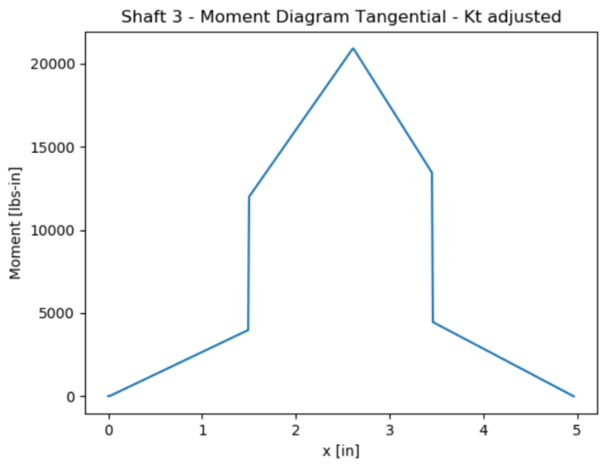


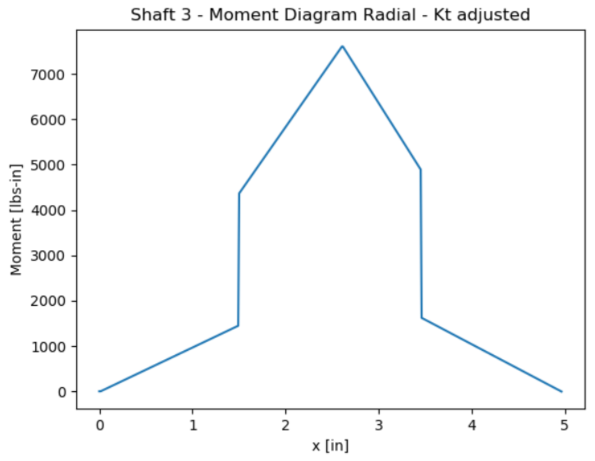
12)





13)





14) Output Section Diameters [in]: [0.5204481136653292, 0.5789547911004376, 1.2102725796730098, 0.42036596721285024, 0.20024439943937306, 0.1252455910116488]

15) These diameters work with the calculated gear and pinion diameters, so recalculation is unnecessary.

Shaft Design – Transfer Shaft

1) Section Lengths [in]: [0.5, 1, 0.125, 0.125, 1.713, 6.074, 1.713, 0.125, 0.125, 1, 0.5]

2) Sn [psi]: 310000

2) Sy [psi]: 287000

3) Cs: 0.8116523277245

4) Sn' [psi]: 203805.89949162197

5) N: 3

6) Torque [lbs-in]: 16806.761990504147

7) Tangential Force Input Gear [lbs]: 3055.7749073643904

7) Tangential Force Output Pinion [lbs]: 13840.862815709299

8) Radial Force Input Gear [lbs]: 2060.861760605488

8) Radial Force Output Pinion [lbs]: 5037.662081480083

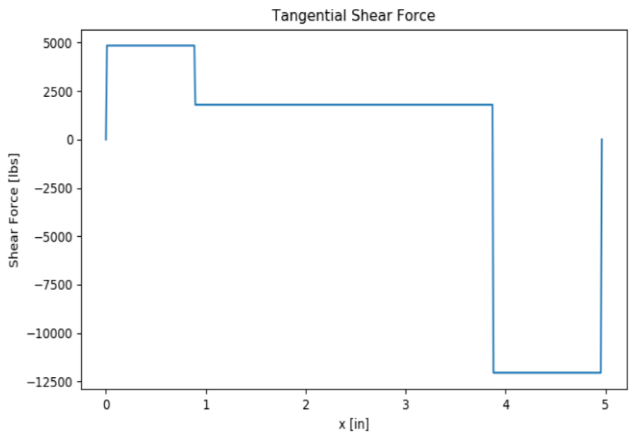
9) Tangential Reaction Force K: 12048.63065635522

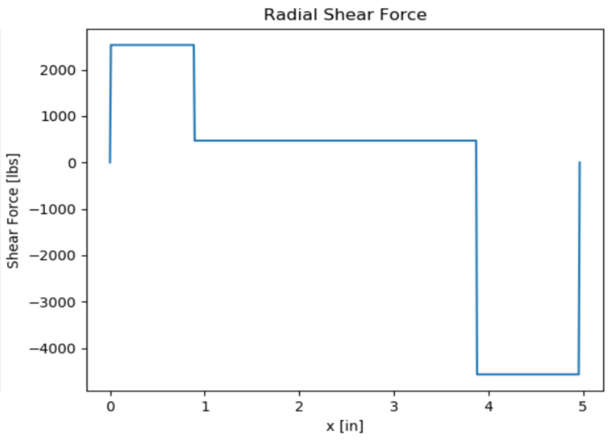
9) Tangential Reaction Force A: 4848.00706671847

10) Radial Reaction Force K: 4567.832331621228

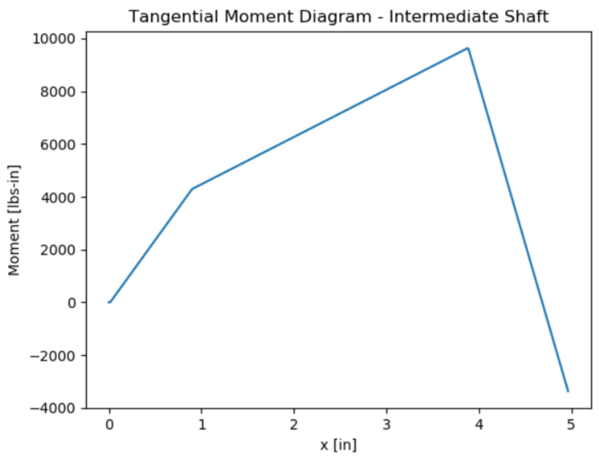
10) Radial Reaction Force A: 2530.691510464343

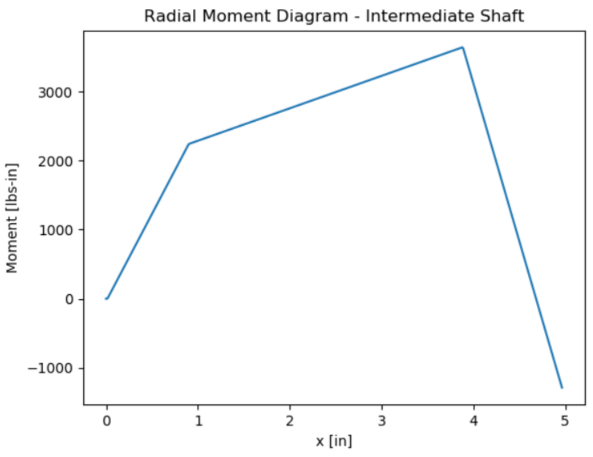
11)



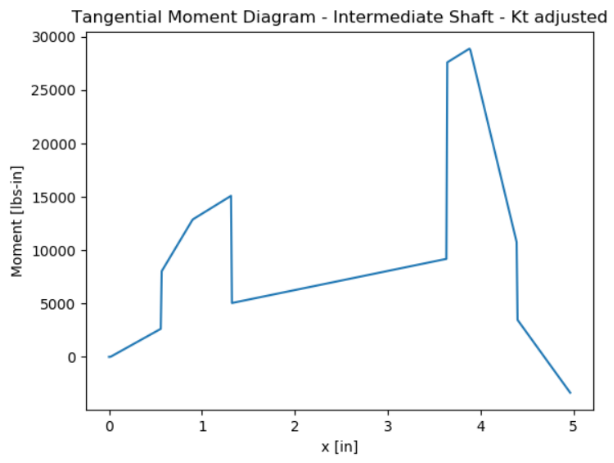


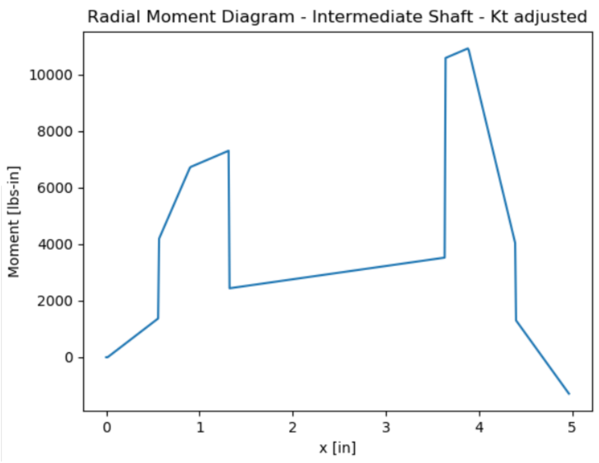
12)





13)





14) Transfer Shaft Diameters [in]: [0.13868749077397977, 0.3075948047965429, 0.9900256069639011, 0.33997917166327013, 1.4733669132301461, 0.3568389793739999, 0.350192712709277]

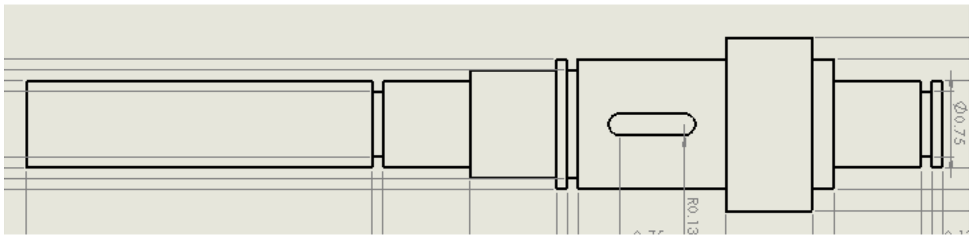
15) These diameters work with the calculated gear and pinion diameters, so recalculation is unnecessary.

Final Design

The final design can be broken down into 3 subassemblies which are fully dimensioned in Appendix B. The subassemblies are the input shaft and pinion, the output shaft and gear, and finally the transfer shaft, gear and pinion. Each of these subassemblies have the same key seat that attaches into both a notch on the gear and a slot on the shafts. These are not visible in the overall assembly drawing in

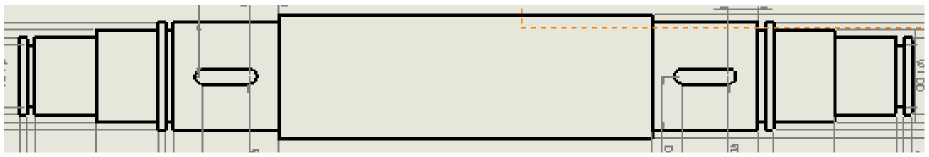
Appendix B however are what connect the gears and pinions to their corresponding shafts. Each of the shafts is also complete with retaining rings that are meant to attach to the outside frame and any internal structuring to prevent the shaft moving along its length causing the gears to disengage.

For simplicity of design and machining, we opted to use the same diameters for the input and output shaft; differences in stresses are accounted for in material selection. The basic design of the input and output shafts is that the gear or pinion will sit on the shaft and be affixed to prevent rotation with a key seat that goes in between the gear and a slot in the shaft. The translational motion along the shaft will be prevented by having a shoulder behind the gear or pinion with a small slot for a retaining ring to sit and prevent movement in away from the shoulder. The shaft then steps down to a diameter that acts as a shoulder to hold the shaft to the outside container. Then drops the diameter to go through the container and has another slot for a second retaining ring to sit and affix the shaft to the container. An analogous stepping of diameters is chosen for the side of the shaft facing into the internal supports in the container doing the exact same logic and requiring the same retaining ring size. The diameters were decided rather arbitrarily to give the desired geometry and to be at regular sizes. They could be chosen this way as the required diameter values everywhere else other than where the gear or pinion sits is much lower than where it does. There is also an extension of the shaft on the input and output sides of 4 inches to meet the project requirements.



Input / Output Shaft General Shape

The transfer shaft design is based on a similar methodology to the input and output shafts. The shaft was designed to be symmetrical to once again simplify the manufacturing of the overall assembly. The values for the diameter required at the two locations where the gear and pinion sit are not as close as the input and output shaft values, but we still determine were reasonable to make the same diameter. Once again, the values of required diameter of the rest of the shaft was much smaller than the values required at the locations where the gear and pinion sit. Due to this the values for the rest of the diameters could be decided arbitrarily to yield the desired geometry. The main large diameter section in the middle of the shaft serves as a shoulder for the pinion and the gear to sit against. On the other side of the gear and pinion is a slot for the retaining rings meant to hold the shaft to the arbitrary internal frame and fully locks in the gear from translational motion along the shaft. The gear and pinion are held in place from free rotation by the same common key seat which sits in the slot and the notch in the gear and pinion. After the gear and pinion the diameter drops down to act as a shoulder for the ends of the shaft to sit in an internal frame through hole. Then, there is a portion to act as the through portion of the shaft which will receive the supporting forces. The ends of the shaft are then held in place by a second type of retaining ring which locks in the translational motion along the shaft at the supports.



Transfer Shaft General Shape

Due to the differences in diameter between the two designed shafts, the pinions and gears are all unique. However, the outer diameters, tooth design, and teeth numbers will all be the same from pinion to pinion and gear to gear. However, the circular cut-out in the center of the gears and pinions will be corresponding in diameter to the section where the gear sits. As mentioned, the gears will have the key seat notch machined into them as well to allow for a locking of rotational motion. Gears and pinions are designed with 20-degree full-depth involute teeth and thus are assumed to mate well; as seen in the overall assembly in Appendix B, this is the case.

Each of the retaining rings is shown with screw holes; however, these are completely optional and could be filled with magnetic pieces to magnetically clip to the steel gears. In our scenario, there are no forces along the shaft and thus translational movement that would be prevented by the retaining rings will not happen. We have left the option for screws but have not included this in our proposed design.

Recommendations for Future Work

The recommendations for future work on this project include material, retaining ring considerations and cost.

Regarding material selection, it would have been wise to make all the gears and shafts out of the same materials. At the bare minimum, it would be nice to have all the gears made from one material and the shafts made of another. This would allow for uniformity in the purchase of the materials. This would, however, give up the advantage of keeping the geometry the same for input and output shafts and/or identical geometry of the gears and pinions for both stages. Finally, it was not considered how the different materials of gears affect the wear and tear as two different hardness materials are pushing against each other. This has some trade-offs and thus if the project were to be furthered it would be worth investigating how this affects this design.

Only a very basic design has been made for the retaining rings and they have not been fully designed. The retaining rings have been, as mentioned, shown with screw holes and how they are affixed to the gears has not been designed for. So, as a future continuation would be to design how the retaining rings will be attached to the gears. Additionally, the retaining rings would have to be taken on and from the shaft at least once and thus the material they are made off would need to be able to do this. This has not been designed for or considered in material selection; as such, it would be a necessary continuation of the project. These issues were not addressed already as they were out of the scope of the project.

The final future thing to consider is the cost. The cost of this assembly was not considered in any way, shape, or form. The materials were chosen at need with no consideration for their cost or the cost of their machining. The gears were designed with no consideration of how large and expensive they would be. The cost of sourcing and machining many materials was not considered either.

Appendix A – Python Code

```
import math
import numpy as np
import matplotlib.pyplot as plt

### GEAR SET 1 ###

# INPUTS

power = 20  # horsepower
lifetime = 12000  # gear lifetime [hours]
inputSpeed = 1750  # rpm
outputSpeed = 85
minNumTeeth = 17  # From table xxxx
Ko = 2.00  # Adjustment factor from 9-1 -> Moderate Shock Input, Moderate Shock Output
serviceFactor = 1.00

# SOLVER

targetVelocityRatio = inputSpeed / outputSpeed

print("GEAR SET 1 PARAMS \n")
print("1) Target Velocity Ratio: " + str(targetVelocityRatio))
targetVR = math.sqrt(targetVelocityRatio)  # Velocity Ratio for both sets of gears
print("2) Target Gearset Velocity Ratio:" + str(targetVR))
print("3) Num Pinion Teeth: " + str(minNumTeeth))
pinionTeeth = minNumTeeth
gearTeeth = round(pinionTeeth * targetVR)
print("4) Num Gear Teeth: " + str(gearTeeth))

VR = gearTeeth / pinionTeeth
print("5) Actual Velocity Ratio: " + str(VR))

adjustedPower = Ko * power
print("6) Design Power [W]: " + str(adjustedPower))
pitchDiameter = 7.00  # Pd: From 9-11, adjusted power corresponds to a pitch diameter
of 7 in
print("7) Pitch Diameter [in]: " + str(pitchDiameter))
gearDiameter = gearTeeth / pitchDiameter
pinionDiameter = pinionTeeth / pitchDiameter

adjustmentDiameter = 1 / pitchDiameter

print("8) Gear Diameter [in]: " + str(gearDiameter))
print("8) Pinion Diameter [in]: " + str(pinionDiameter))

addendum = adjustmentDiameter
dedendum = 1.25 / pitchDiameter

print("Addendum: " + str(addendum))
print("Dedendum: " + str(dedendum))

addendumDiameter = 2 * addendum + pinionDiameter
dedendumDiameter = pinionDiameter - 2 * dedendum

print("Pinion Addendum Diameter: " + str(addendumDiameter))
print("Pinion Dedendum Diameter: " + str(dedendumDiameter))

addendumDiameter = 2 * addendum + gearDiameter
dedendumDiameter = gearDiameter - 2 * dedendum
```

```
print("Gear Addendum Diameter: " + str(addendumDiameter))
print("Gear Dedendum Diameter: " + str(dedendumDiameter))

geartrainHeight = (3 / 2) * gearDiameter + pinionDiameter / 2 + 3 * adjustmentDiameter
# calculated Y (in)

print("9) Geartrain Height [in]: " + str(geartrainHeight))

C = (pinionDiameter + gearDiameter) / (2 * pitchDiameter)  # center distance
print("10) Center Distance [in]: " + str(C))
tangentialVelocity = (math.pi * pitchDiameter * inputSpeed) / 12
transmittedLoad = 33000 * power / tangentialVelocity  # Wt

print("11) Pitch Line Speed: " + str(tangentialVelocity))
print("12) Transmitted Load: " + str(transmittedLoad))

nominalFaceWidth = 12 / pitchDiameter

print("13) Nominal Face Width: " + str(nominalFaceWidth))

# Material is steel
elasticCoefficient = 2300  # Cp
print("14) Elastic Coefficient: " + str(elasticCoefficient))

agmaQualityNumber = 6  # A6 - automotive transmission equiv. from 9-5
print("15) AGMA Quality Number: " + str(agmaQualityNumber))

dynamicFactorsAt3207VT = {}  # Mapping between AGMA Quality Numbers and Dynamic
Factors at Vt of 3207
dynamicFactorsAt3207VT[6] = 1.13
dynamicFactorsAt3207VT[7] = 1.23
dynamicFactorsAt3207VT[8] = 1.34
dynamicFactorsAt3207VT[9] = 1.46
dynamicFactorsAt3207VT[10] = 1.58
dynamicFactorsAt3207VT[11] = 1.75

dynamicFactor = dynamicFactorsAt3207VT[agmaQualityNumber]  # Kv

print("16) Dynamic Factor: " + str(dynamicFactor))

KMRatio = nominalFaceWidth / pinionDiameter

print("17) F/Dp Ratio: " + str(KMRatio))

# Use this formula because 1 < F < 15 (See Figure 9-12)
pinionProportionFactor = nominalFaceWidth / (10 * pinionDiameter) - 0.0375 + 0.0125 *
nominalFaceWidth  # from fig 9-12

print("18) Pinion Proportion Factor: " + str(pinionProportionFactor))

# Assume commercial gears
meshAlignmentFactor = 0.127 + 0.0158 * nominalFaceWidth - (1.093 * pow(10, -4) *
pow(nominalFaceWidth, 2))  # Cma
print("19) Mesh Alignment Factor: " + str(meshAlignmentFactor))

Km = 1 + meshAlignmentFactor + pinionProportionFactor

print("20) Km: " + str(Km))

sizeFactor = 1  # Ks since pitch diameter is greater than 5 inches from Table 9-2
print("21) Ks: " + str(sizeFactor))
```

```
rimThicknessFactor = 1.0  # Kb since VR is greater than 1.2, thickness factor is 1.0
print("22) Rim Thickness Factor: " + str(rimThicknessFactor))

reliabilityFactor = 1.00  # Km From table 9-11, at 99% reliability
print("23) Reliability Factor: " + str(reliabilityFactor))

numPinionCycles = 60 * lifetime * inputSpeed  # Ncp
numGearCycles = 60 * lifetime * (inputSpeed / VR)  # Ncg

print("24) Number Pinion Cycles: " + str(numPinionCycles))
print("24) Number Gear Cycles: " + str(numGearCycles))


def bending_strength_stress_cycle_factor(numCycles):
    # This is linear because our load cycles is greater than 10^7 (Fig 9-21)
    return 1.3558 * numCycles ** -0.0178


def pitting_strength_stress_cycle_factor(numCycles):
    # This is linear because our load cycles is greater than 10^7 (Fig 9-22)
    return 1.4488 * numCycles ** -0.023


bendingStressCycleFactorPinion = bending_strength_stress_cycle_factor(numPinionCycles)
# Ynp from Figure 9-21
bendingStressCycleFactorGear = bending_strength_stress_cycle_factor(numGearCycles)  #
Yng from Figure 9-21

pittingResistanceStressCycleFactorPinion =
pitting_strength_stress_cycle_factor(numPinionCycles)  # Znp from Figure 9-22
pittingResistanceStressCycleFactorGear =
pitting_strength_stress_cycle_factor(numGearCycles)  # Zng from Figure 9-22

print("25) Ynp: " + str(bendingStressCycleFactorPinion))
print("25) Yng: " + str(bendingStressCycleFactorGear))

print("26) Znp: " + str(pittingResistanceStressCycleFactorPinion))
print("26) Zng: " + str(pittingResistanceStressCycleFactorGear))

# From Figure 9-10: Np = 17, Ng = 77
bendingGeometryFactorPinion = 0.295  # Jp from Fig 9-17
bendingGeometryFactorGear = 0.41  # Jg from Fig 9-17
print("27) Bending Geometry Factor Pinion " + str(bendingGeometryFactorPinion))
print("27) Bending Geometry Factor Gear " + str(bendingGeometryFactorGear))

# From Figure 9-17 VR = 4.54 Np = 17

pittingGeometryFactor = 0.105  # I from Figure 9-17, using 20deg pressure angle

print("28) Pitting Geometry Factor: " + str(pittingGeometryFactor))

allowableBendingStressNumberPinion = (((transmittedLoad * pitchDiameter) /
(nominalFaceWidth * bendingGeometryFactorPinion)) * Ko \
                                      * Km * sizeFactor * rimThicknessFactor *
dynamicFactor) * (
                                                 serviceFactor * reliabilityFactor / (
                                             bendingStressCycleFactorPinion))  # Satp
allowableBendingStressNumberGear = (((transmittedLoad * pitchDiameter) /
(nominalFaceWidth * bendingGeometryFactorGear)) * Ko \
                                    * Km * sizeFactor * rimThicknessFactor *
dynamicFactor) * (
                                               serviceFactor * reliabilityFactor / (
                                           bendingStressCycleFactorGear))  # Satg
```

```
allowableContactStressNumberPinion = ((elasticCoefficient * reliabilityFactor *
serviceFactor) /
                                      pittingResistanceStressCycleFactorPinion) *
((transmittedLoad * Ko * sizeFactor * Km *

dynamicFactor) / (nominalFaceWidth *

pinionDiameter *

pittingGeometryFactor)) ** 0.5  # Sacp

allowableContactStressNumberGear = ((elasticCoefficient * reliabilityFactor *
serviceFactor) /
                                      pittingResistanceStressCycleFactorGear) *
((transmittedLoad * Ko * sizeFactor * Km *

dynamicFactor) / (nominalFaceWidth *

gearDiameter *

pittingGeometryFactor)) ** 0.5  # Sacg

print("29) Bending Stress Number Pinion: " + str(allowableBendingStressNumberPinion))
print("29) Bending Stress Number Gear: " + str(allowableBendingStressNumberGear))
print("30) Contact Stress Number Pinion: " + str(allowableContactStressNumberPinion))
print("30) Contact Stress Number Gear: " + str(allowableContactStressNumberGear))

limitingStress = max([allowableBendingStressNumberPinion,
allowableBendingStressNumberGear, allowableContactStressNumberGear,
allowableContactStressNumberPinion])

print("31) Limiting Stress: " + str(limitingStress))

def calc_brinell_hardness(limitingStress):
    return (limitingStress / 1000 - 29.10) / 0.322

brinellHardness = calc_brinell_hardness(limitingStress) # HB from Fig 9-19 using Grade
1 Steel
print("32) Brinell Hardness: " + str(brinellHardness))

# Use appendix 3 to select a steel -> Use 1144 Cold-Drawn (HB = 200)
print("33) Choose 1144 Cold-Drawn Steel with HB = 200")

### GEAR SET 2 ###

print("\nGEAR SET 2 PARAMS \n")

print("1) Target Velocity Ratio: " + str(targetVR))
targetVR = math.sqrt(targetVelocityRatio)  # Velocity Ratio for both sets of gears
print("2) Target Gearset Velocity Ratio:" + str(targetVR))
pinionTeeth = minNumTeeth
print("3) Num Pinion Teeth: " + str(minNumTeeth))
gearTeeth = round(pinionTeeth * targetVR)
print("4) Num Gear Teeth: " + str(gearTeeth))

VR = gearTeeth / pinionTeeth
print("5) Actual Velocity Ratio: " + str(VR))

inputSpeed = inputSpeed / VR

adjustedPower = Ko * power
print("6) Design Power: " + str(adjustedPower))
```

```python
pitchDiameter = 7.00  # Pd: From 9-11, adjusted power corresponds to a pitch diameter
of 7 in
print("7) Pitch Diameter: " + str(pitchDiameter))

gearDiameter = gearTeeth / pitchDiameter
pinionDiameter = pinionTeeth / pitchDiameter

adjustmentDiameter = 1 / pitchDiameter

print("8) Gear Diameter: " + str(gearDiameter))
print("8) Pinion Diameter: " + str(pinionDiameter))

addendumDiameter = 2 * addendum + pinionDiameter
dedendumDiameter = pinionDiameter - 2 * dedendum

print("Pinion Addendum Diameter: " + str(addendumDiameter))
print("Pinion Dedendum Diameter: " + str(dedendumDiameter))

addendumDiameter = 2 * addendum + gearDiameter
dedendumDiameter = gearDiameter - 2 * dedendum

print("Gear Addendum Diameter: " + str(addendumDiameter))
print("Gear Dedendum Diameter: " + str(dedendumDiameter))

geartrainHeight = (3 / 2) * gearDiameter + pinionDiameter / 2 + 3 * adjustmentDiameter
# calculated Y (in)

print("9) Geartrain Height: " + str(geartrainHeight))

C = (pinionDiameter + gearDiameter) / (2 * pitchDiameter)  # center distance
print("10) Center Distance: " + str(C))
tangentialVelocity = (math.pi * pitchDiameter * inputSpeed) / 12
transmittedLoad = 33000 * power / tangentialVelocity  # Wt

print("11) Pitch Line Speed: " + str(tangentialVelocity))
print("12) Transmitted Load: " + str(transmittedLoad))

nominalFaceWidth = 12 / pitchDiameter

print("13) Nominal Face Width: " + str(nominalFaceWidth))

# Material is steel
elasticCoefficient = 2300  # Cp
print("14) Elastic Coefficient: " + str(elasticCoefficient))
agmaQualityNumber = 6  # A6 - automotive transmission equiv. from 9-5
print("15) AGMA Quality Number: " + str(agmaQualityNumber))

# Mapping between AGMA Quality Numbers and Dynamic Factors at Vt of 708
dynamicFactorsAt3207VT = {6: 1.07, 7: 1.12, 8: 1.18, 9: 1.24, 10: 1.31, 11: 1.39}

dynamicFactor = dynamicFactorsAt3207VT[agmaQualityNumber]  # Kv

print("16) Dynamic Factor: " + str(dynamicFactor))

KMRatio = nominalFaceWidth / pinionDiameter

print("17) F/Dp Ratio: " + str(KMRatio))

# Use this formula because 1 < F < 15 (See Figure 9-12)
pinionProportionFactor = nominalFaceWidth / (10 * pinionDiameter) - 0.0375 + 0.0125 *
nominalFaceWidth  # from fig 9-12

print("18) Pinion Proportion Factor: " + str(pinionProportionFactor))
```

```
# Assume commercial gears
meshAlignmentFactor = 0.127 + 0.0158 * nominalFaceWidth - (1.093 * pow(10, -4) *
pow(nominalFaceWidth, 2))  # Cma

print("19) Mesh Alignment Factor: " + str(meshAlignmentFactor))

Km = 1 + meshAlignmentFactor + pinionProportionFactor

print("20) Km: " + str(Km))

sizeFactor = 1  # Ks since pitch diameter is greater than 5 inches from Table 9-2

print("21) Ks: " + str(sizeFactor))

rimThicknessFactor = 1.0  # Kb since VR is greater than 1.2, thickness factor is 1.0

print("22) Rim Thickness Factor: " + str(rimThicknessFactor))

reliabilityFactor = 1.00  # Km From table 9-11, at 99% reliability

print("23) Reliability Factor: " + str(reliabilityFactor))

numPinionCycles = 60 * lifetime * inputSpeed  # Ncp
numGearCycles = 60 * lifetime * (inputSpeed / VR)  # Ncg

print("24) Number Pinion Cycles: " + str(numPinionCycles))
print("24) Number Gear Cycles: " + str(numGearCycles))

bendingStressCycleFactorPinion = bending_strength_stress_cycle_factor(numPinionCycles)
# Ynp from Figure 9-21
bendingStressCycleFactorGear = bending_strength_stress_cycle_factor(numGearCycles)  #
Yng from Figure 9-21

pittingResistanceStressCycleFactorPinion =
pitting_strength_stress_cycle_factor(numPinionCycles)  # Znp from Figure 9-22
pittingResistanceStressCycleFactorGear =
pitting_strength_stress_cycle_factor(numGearCycles)  # Zng from Figure 9-22

print("25) Ynp: " + str(bendingStressCycleFactorPinion))
print("25) Yng: " + str(bendingStressCycleFactorGear))

print("26) Znp: " + str(pittingResistanceStressCycleFactorPinion))
print("26) Zng: " + str(pittingResistanceStressCycleFactorGear))

# From Figure 9-10: Np = 17, Ng = 77
bendingGeometryFactorPinion = 0.295  # Jp from Fig 9-17
bendingGeometryFactorGear = 0.41  # Jg from Fig 9-17
print("27) Bending Geometry Factor Pinion " + str(bendingGeometryFactorPinion))
print("27) Bending Geometry Factor Gear " + str(bendingGeometryFactorGear))

# From Figure 9-17 VR = 4.54 Np = 17

pittingGeometryFactor = 0.105  # I from Figure 9-17, using 20deg pressure angle
print("28) Pitting Geometry Factor: " + str(pittingGeometryFactor))
allowableBendingStressNumberPinion = (((transmittedLoad * pitchDiameter) /
(nominalFaceWidth * bendingGeometryFactorPinion)) * Ko \
                                      * Km * sizeFactor * rimThicknessFactor *
dynamicFactor) * (
                                                 serviceFactor * reliabilityFactor / (
                                             bendingStressCycleFactorPinion))  # Satp
allowableBendingStressNumberGear = (((transmittedLoad * pitchDiameter) /
(nominalFaceWidth * bendingGeometryFactorGear)) * Ko \
                                    * Km * sizeFactor * rimThicknessFactor *
dynamicFactor) * (
                                                serviceFactor * reliabilityFactor / (
                                            bendingStressCycleFactorGear))  # Satg

allowableContactStressNumberPinion = ((elasticCoefficient * reliabilityFactor *
serviceFactor) /
                                      pittingResistanceStressCycleFactorPinion) *
((transmittedLoad * Ko * sizeFactor * Km *

dynamicFactor) / (nominalFaceWidth *

pinionDiameter *

pittingGeometryFactor)) ** 0.5  # Sacp

allowableContactStressNumberGear = ((elasticCoefficient * reliabilityFactor *
serviceFactor) /
                                      pittingResistanceStressCycleFactorGear) *
((transmittedLoad * Ko * sizeFactor * Km *

dynamicFactor) / (nominalFaceWidth *

gearDiameter *

pittingGeometryFactor)) ** 0.5  # Sacg

print("29) Bending Stress Number Pinion: " + str(allowableBendingStressNumberPinion))
print("29) Bending Stress Number Gear: " + str(allowableBendingStressNumberGear))
print("30) Contact Stress Number Pinion: " + str(allowableContactStressNumberPinion))
print("30) Contact Stress Number Gear: " + str(allowableContactStressNumberGear))

limitingStress = max([allowableBendingStressNumberPinion,
allowableBendingStressNumberGear, allowableContactStressNumberGear,
allowableContactStressNumberPinion])

print("31) Limiting Stress: " + str(limitingStress))

brinellHardness = calc_brinell_hardness(limitingStress) # HB from Fig 9-19 using Grade
1 Steel
print("32) Brinell Hardness: " + str(brinellHardness))

# Use appendix 3 to select a steel -> Use SAE 4150 OQT 700 (HB = 495)
print("33) Choose SAE 4150 OQT 700 with HB = 495")

print("Final Output Speed: " + str(inputSpeed / VR))


# Shaft 1

print("\nSHAFT 1\n")

powerIn = 20 * 63000 # watts
lengthSec1 = 0.5 # inches
lengthSec2 = 1
lengthSec3ToMiddle = 1.107
lengthSec3MiddleToEnd = 0.857
lengthSec4 = 1
lengthSec5 = 0.25
lengthSec6 = 0.25

lengthAToCenter = lengthSec1 + lengthSec2 + lengthSec3ToMiddle
lengthCenterToC = lengthSec3MiddleToEnd + lengthSec4 + lengthSec5 + lengthSec6;
```

```python
totalLength = lengthAToCenter + lengthCenterToC

def plot_shear_diagram(force, lengthAToCenter, lengthCenterToC, figureNumber,
plotTitle):
    rc = force * lengthAToCenter / (lengthAToCenter + lengthCenterToC)

    ra = force - rc
    print("10) Reaction Force C [lbs]: " + str(rc))
    print("10) Reaction Force A [lbs]: " + str(ra))
    x = np.linspace(0, lengthAToCenter + lengthCenterToC, discreteSubdivisions)
    y = np.linspace(0, 0, discreteSubdivisions)
    for i in range(0, int(discreteSubdivisions * lengthAToCenter / (lengthAToCenter +
lengthCenterToC))) :
        y[i] = ra

    for i in range(int(discreteSubdivisions * lengthAToCenter / (lengthAToCenter +
lengthCenterToC)), discreteSubdivisions):
        y[i] = -1 * rc

    y[499] = 0
    y[0] = 0
    plt.figure(figureNumber)
    plt.title(plotTitle)
    plt.ylabel("Shear Force [lbs]")
    plt.xlabel("x [in]")
    plt.plot(x, y)

    return [x, y]

def plot_moment_diagram(distance, shearForce, figureNumber, plotTitle):
    moment = np.linspace(0, 0, discreteSubdivisions)
    for i in range(0, discreteSubdivisions):
        moment[i] = np.trapz(shearForce[0 : i], distance[0 : i])
    plt.figure(figureNumber)
    plt.title(plotTitle)
    plt.ylabel("Moment [lbs-in]")
    plt.xlabel("x [in]")
    plt.plot(distance, moment)

    # Adjust by Kt
    for i in range(int(discreteSubdivisions * (lengthSec1 + lengthSec2) /
(lengthAToCenter + lengthCenterToC)),
                   int(discreteSubdivisions * (lengthAToCenter +
lengthSec3MiddleToEnd) / (lengthAToCenter + lengthCenterToC))):
        moment[i] = moment[i] * Kt
    plt.figure(figureNumber + 1)
    plt.title(plotTitle + " - Kt adjusted")
    plt.ylabel("Moment [lbs-in]")
    plt.xlabel("x [in]")
    plt.plot(distance, moment)
    return [distance, moment]

def cs_from_d(D): # From Table 5-4
    if D <= 0.3:
        return 1
    elif D <= 2 :
        return (D/0.3)**-0.11
    else:
        return 0.859 - 0.02125 * D


def get_section_diameter(torque, moment):
    return ((32 * N / math.pi) * ((Kt * moment / SnPrime) ** 2 + 0.75 * (torque /
Sy)**2) ** 1/2) ** (1/3)

def get_section_diameters(tangentialMoment, radialMoment, torque):
    section1MaxMoment = 0
    for i in range(0, int(discreteSubdivisions * lengthSec1 / totalLength)):
        if math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i] ** 2) >
section1MaxMoment:
            section1MaxMoment = math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i]
** 2)
    section1Diameter = get_section_diameter(torque, section1MaxMoment)

    section2MaxMoment = 0
    for i in range(int(discreteSubdivisions * lengthSec1 / totalLength),
                   int(discreteSubdivisions * (lengthSec1 + lengthSec2) /
totalLength)):
        if math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i] ** 2) >
section2MaxMoment:
            section2MaxMoment = math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i]
** 2)
    section2Diameter = get_section_diameter(torque, section2MaxMoment)

    section3MaxMoment = 0

    for i in range(int(discreteSubdivisions * (lengthSec1 + lengthSec2) /
totalLength),
                   int(discreteSubdivisions * (lengthSec1 + lengthSec2 +
lengthSec3MiddleToEnd + lengthSec3ToMiddle) / totalLength)):
        if math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i] ** 2) >
section3MaxMoment:
            section3MaxMoment = math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i]
** 2)
    section3Diameter = get_section_diameter(torque, section3MaxMoment)

    section4MaxMoment = 0
    for i in range(int(discreteSubdivisions * (lengthSec1 + lengthSec2 +
lengthSec3MiddleToEnd + lengthSec3ToMiddle) / totalLength),
                   int(discreteSubdivisions * (lengthSec1 + lengthSec2 +
lengthSec3MiddleToEnd + lengthSec3ToMiddle + lengthSec4) / totalLength)):
        if math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i] ** 2) >
section4MaxMoment:
            section4MaxMoment = math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i]
** 2)
    section4Diameter = get_section_diameter(0, section4MaxMoment)

    section5MaxMoment = 0
    for i in range(int(discreteSubdivisions * (lengthSec1 + lengthSec2 +
lengthSec3MiddleToEnd + lengthSec3ToMiddle + lengthSec4) / totalLength),
                   int(discreteSubdivisions * (lengthSec1 + lengthSec2 +
lengthSec3MiddleToEnd + lengthSec3ToMiddle + lengthSec4 + lengthSec5) / totalLength)):
        if math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i] ** 2) >
section5MaxMoment:
            section5MaxMoment = math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i]
** 2)
    section5Diameter = get_section_diameter(0, section5MaxMoment)


    section6MaxMoment = 0
    for i in range(int(discreteSubdivisions * (lengthSec1 + lengthSec2 +
lengthSec3MiddleToEnd + lengthSec3ToMiddle + lengthSec4 + lengthSec5) / totalLength),
                   int(discreteSubdivisions)):
        if math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i] ** 2) >
section6MaxMoment:
            section6MaxMoment = math.sqrt(tangentialMoment[i] ** 2 + radialMoment[i]
** 2)
    section6Diameter = get_section_diameter(0, section6MaxMoment)

    return [section1Diameter, section2Diameter, section3Diameter, section4Diameter,
section5Diameter, section6Diameter]

inputSpeed = 1750
Kt = 3
Cr = 0.81 # 99% reliability from Table 5-3
D = 2
Sn = 310000 # SAE 9255 Q&T400 (Appendix 3) [ksi]
Sy = 287000 # SAE 9255 Q&T400 (Appendix 3) [ksi]
discreteSubdivisions = 500

Cs = cs_from_d(D)
SnPrime = Cs * Cr * Sn
N = 3 # Safety Factor

print("1) Section Lengths [in]: " + str([lengthSec1, lengthSec2, lengthSec3ToMiddle +
lengthSec3MiddleToEnd, lengthSec4, lengthSec5, lengthSec6]))
print("2) Sn [psi]: " + str(Sn))
print("2) Sy [psi]: " + str(Sy))
print("3) Cs : " + str(Cs))
print("4) Sn' [psi]: " + str(SnPrime))
print("5) N: " + str(N))

angularVelocityIn = inputSpeed * 2 * math.pi / 60 # rpm
torque = powerIn / angularVelocityIn
print("6) Torque [lbs-in]: " + str(torque))

tangentialForceGear = torque / (pinionDiameter / 2)
radialForceGear = tangentialForceGear * math.tan(math.pi / 9)

print("7) Tangential Force Gear [lbs]: " + str(tangentialForceGear))
print("8) Radial Force Gear [lbs]: " + str(radialForceGear))

def father_of_get_section_diameters(tangentialForceGear, radialForceGear,
startingFigureNumber, torque):
    [distance, shearForceTangential] = plot_shear_diagram(tangentialForceGear,
lengthAToCenter, lengthCenterToC, startingFigureNumber, "Shaft 3 - Tangential Shear
Force")
    [distance, shearForceRadial] = plot_shear_diagram(radialForceGear,
lengthAToCenter, lengthCenterToC, startingFigureNumber + 1, "Shaft 3 - Radial Shear
Force")

    [distance, tangentialMoment] = plot_moment_diagram(distance, shearForceTangential,
startingFigureNumber + 2, "Shaft 3 - Moment Diagram Tangential")
    [distance, radialMoment] = plot_moment_diagram(distance, shearForceRadial,
startingFigureNumber + 4, "Shaft 3 - Moment Diagram Radial")

    return get_section_diameters(tangentialMoment, radialMoment, torque)

section_diameters = father_of_get_section_diameters(tangentialForceGear,
radialForceGear, 1, torque)
print("14) Section Diameters [in]: " + str(section_diameters))

print("\nSHAFT 3\n")

print("1) Section Lengths [in]: " + str([lengthSec1, lengthSec2, lengthSec3ToMiddle +
lengthSec3MiddleToEnd, lengthSec4, lengthSec5, lengthSec6]))
print("2) Sn [psi]: " + str(Sn))
print("2) Sy [psi]: " + str(Sy))
print("3) Cs : " + str(Cs))
print("4) Sn' [psi]: " + str(SnPrime))
print("5) N: " + str(N))

angularVelocityIn = (inputSpeed / VR) * 2 * math.pi / 60
torque = powerIn / angularVelocityIn
print("6) Torque [lbs-in]: " + str(torque))

tangentialForceGear = torque / (gearDiameter / 2)
radialForceGear = tangentialForceGear * math.tan(math.pi / 9)
print("7) Tangential Force Gear [lbs]: " + str(tangentialForceGear))
print("8) Radial Force Gear [lbs]: " + str(radialForceGear))

section_diameters = father_of_get_section_diameters(tangentialForceGear,
radialForceGear, 7, torque)
print("14) Output Section Diameters [in]: " + str(section_diameters))

print("\nSHAFT 2\n")

lengthA = 0.5
lengthB = 1
lengthC = 0.125
lengthD = 0.125
lengthEFirstHalf = 0.8565
lengthESecondHalf = 0.8565
lengthF = 6.074
lengthGFirstHalf = 0.8565
lengthGSecondHalf = 0.8565
lengthH = 0.125
lengthI = 0.125
lengthJ = 1
lengthK = 0.5
totalLength = lengthA + lengthB + lengthC + lengthD + lengthEFirstHalf +
lengthESecondHalf + lengthF + lengthGFirstHalf + \
    lengthGSecondHalf + lengthH + lengthI + lengthJ + lengthK

print("1) Section Lengths [in]: " + str([lengthA, lengthB, lengthC, lengthD,
lengthEFirstHalf + lengthESecondHalf, lengthF,
                                         lengthGFirstHalf + lengthGSecondHalf,
lengthH, lengthI, lengthJ, lengthK]))
print("2) Sn [psi]: " + str(Sn))
print("2) Sy [psi]: " + str(Sy))
print("3) Cs : " + str(Cs))
print("4) Sn' [psi]: " + str(SnPrime))
print("5) N: " + str(N))

shaftSpeed = (inputSpeed / VR) * 2 * math.pi / 60 # rpm
torque = power * 34000 / shaftSpeed
print("6) Torque [lbs-in]: " + str(torque))
tangentialForceInputGear = torque / (gearDiameter / 2)
tangentialForceOutputPinion = torque / (pinionDiameter / 2)

radialForceInputGear = tangentialForceGear * math.tan(math.pi / 9)
radialForceOutputPinion = tangentialForceOutputPinion * math.tan(math.pi / 9)

print("7) Tangential Force Input Gear [lbs]: " + str(tangentialForceInputGear))
print("7) Tangential Force Output Pinion [lbs]: " + str(tangentialForceOutputPinion))

print("8) Radial Force Input Gear [lbs]: " + str(radialForceInputGear))
print("8) Radial Force Output Pinion [lbs]: " + str(radialForceOutputPinion))

rkx = ((tangentialForceInputGear * (0.5 * lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf)) + \
      tangentialForceOutputPinion * (0.5 * lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf
                                   + lengthF + lengthGFirstHalf)) \
      / (totalLength - 0.5 * lengthA - 0.5 * lengthB)

rax = tangentialForceOutputPinion + tangentialForceInputGear - rkx

print("9) Tangential Reaction Force K: " + str(rkx))
print("9) Tangential Reaction Force A: " + str(rax))

rky = ((radialForceInputGear * (0.5 * lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf)) + \
      radialForceOutputPinion * (0.5 * lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf
                                   + lengthF + lengthGFirstHalf))\
      / (totalLength - 0.5 * lengthA - 0.5 * lengthB)
ray = radialForceOutputPinion + radialForceInputGear - rky

print("10) Radial Reaction Force K: " + str(rky))
print("10) Radial Reaction Force A: " + str(ray))

x = np.linspace(0, lengthAToCenter + lengthCenterToC, discreteSubdivisions)
shearForceTangential = np.linspace(0, 0, discreteSubdivisions)
for i in range(0, int(discreteSubdivisions * ((0.5 * lengthA + lengthB + lengthC +
lengthD + lengthEFirstHalf) / totalLength))):
    shearForceTangential[i] = rax

for i in range(int(discreteSubdivisions * ((0.5 * lengthA + lengthB + lengthC +
lengthD + lengthEFirstHalf) / totalLength)),
               int(discreteSubdivisions * (0.5 * lengthA + lengthB + lengthC + lengthD
+ lengthEFirstHalf + lengthESecondHalf
                                   + lengthF + lengthGFirstHalf) / totalLength)):
    shearForceTangential[i] = rax - tangentialForceInputGear

for i in range(int(discreteSubdivisions * (0.5 * lengthA + lengthB + lengthC + lengthD
+ lengthEFirstHalf + lengthESecondHalf
                                   + lengthF + lengthGFirstHalf) / totalLength),
discreteSubdivisions):
    shearForceTangential[i] = rax - tangentialForceInputGear -
tangentialForceOutputPinion

shearForceTangential[discreteSubdivisions - 1] = 0
shearForceTangential[0] = 0

plt.figure(13)
plt.title("Tangential Shear Force")
plt.ylabel("Shear Force [lbs]")
plt.xlabel("x [in]")
plt.plot(x, shearForceTangential)

shearForceRadial = np.linspace(0, 0, discreteSubdivisions)
for i in range(0, int(discreteSubdivisions * ((0.5 * lengthA + lengthB + lengthC +
lengthD + lengthEFirstHalf) / totalLength))):
    shearForceRadial[i] = ray

for i in range(int(discreteSubdivisions * ((0.5 * lengthA + lengthB + lengthC +
lengthD + lengthEFirstHalf) / totalLength)),
               int(discreteSubdivisions * (0.5 * lengthA + lengthB + lengthC + lengthD
+ lengthEFirstHalf + lengthESecondHalf
                                   + lengthF + lengthGFirstHalf) / totalLength)):
    shearForceRadial[i] = ray - radialForceInputGear

for i in range(int(discreteSubdivisions * (0.5 * lengthA + lengthB + lengthC + lengthD
+ lengthEFirstHalf + lengthESecondHalf
```

```
                                        + lengthF + lengthGFirstHalf) / totalLength),
discreteSubdivisions):
    shearForceRadial[i] = ray - radialForceInputGear - radialForceOutputPinion

shearForceRadial[discreteSubdivisions - 1] = 0
shearForceRadial[0] = 0

plt.figure(14)
plt.title("Radial Shear Force")
plt.ylabel("Shear Force [lbs]")
plt.xlabel("x [in]")
plt.plot(x, shearForceRadial)

momentTangential = np.linspace(0, 0, discreteSubdivisions)
for i in range(0, discreteSubdivisions):
    momentTangential[i] = np.trapz(shearForceTangential[0: i], x[0: i])
plt.figure(15)
plt.title("Tangential Moment Diagram - Intermediate Shaft")
plt.ylabel("Moment [lbs-in]")
plt.xlabel("x [in]")
plt.plot(x, momentTangential)

# Adjust by Kt
for i in range(int(discreteSubdivisions * (lengthA + lengthB) / (totalLength)),
               int(discreteSubdivisions * (lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf) / (totalLength))):
    momentTangential[i] = momentTangential[i] * Kt

for i in range(int(discreteSubdivisions * (lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf + lengthF) / (totalLength)),
               int(discreteSubdivisions * (totalLength - lengthK - lengthJ) /
(totalLength))):
    momentTangential[i] = momentTangential[i] * Kt

plt.figure(16)
plt.title("Tangential Moment Diagram - Intermediate Shaft - Kt adjusted")
plt.ylabel("Moment [lbs-in]")
plt.xlabel("x [in]")
plt.plot(x, momentTangential)

momentRadial = np.linspace(0, 0, discreteSubdivisions)
for i in range(0, discreteSubdivisions):
    momentRadial[i] = np.trapz(shearForceRadial[0: i], x[0: i])
plt.figure(17)
plt.title("Radial Moment Diagram - Intermediate Shaft")
plt.ylabel("Moment [lbs-in]")
plt.xlabel("x [in]")
plt.plot(x, momentRadial)

# Adjust by Kt
for i in range(int(discreteSubdivisions * (lengthA + lengthB) / (totalLength)),
               int(discreteSubdivisions * (lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf) / (totalLength))):
    momentRadial[i] = momentRadial[i] * Kt

for i in range(int(discreteSubdivisions * (lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf + lengthF) / (totalLength)),
               int(discreteSubdivisions * (totalLength - lengthK - lengthJ) /
(totalLength))):
    momentRadial[i] = momentRadial[i] * Kt

plt.figure(18)
plt.title("Radial Moment Diagram - Intermediate Shaft - Kt adjusted")
plt.ylabel("Moment [lbs-in]")
plt.xlabel("x [in]")
plt.plot(x, momentRadial)

sectionAMaxMoment = 0
for i in range(0, int(discreteSubdivisions * lengthA / totalLength)):
    if math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2) > sectionAMaxMoment:
        sectionAMaxMoment = math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2)
sectionADiameter = get_section_diameter(0, sectionAMaxMoment)

sectionBMaxMoment = 0
for i in range(int(discreteSubdivisions * lengthA / totalLength),
               int(discreteSubdivisions * (lengthA + lengthB) / totalLength)):
    if math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2) > sectionBMaxMoment:
        sectionBMaxMoment = math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2)
sectionBDiameter = get_section_diameter(0, sectionBMaxMoment)

sectionCDEMaxMoment = 0
for i in range(int(discreteSubdivisions * (lengthA + lengthB) / totalLength),
               int(discreteSubdivisions * (lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf) / totalLength)):
    if math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2) >
sectionCDEMaxMoment:
        sectionCDEMaxMoment = math.sqrt(momentTangential[i] ** 2 + momentRadial[i] **
2)
sectionCDEDiameter = get_section_diameter(torque, sectionCDEMaxMoment)

sectionFMaxMoment = 0
for i in range(int(discreteSubdivisions * (lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf)),
               int(discreteSubdivisions * (lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf + lengthF) / totalLength)):
    if math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2) > sectionFMaxMoment:
        sectionFMaxMoment = math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2)
sectionFDiameter = get_section_diameter(torque, sectionFMaxMoment)

sectionGHIMaxMoment = 0
for i in range(int(discreteSubdivisions * (lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf + lengthF) / totalLength),
               int(discreteSubdivisions * (lengthA + lengthB + lengthC + lengthD +
lengthEFirstHalf + lengthESecondHalf + lengthF + lengthGFirstHalf + lengthGSecondHalf
+ lengthH + lengthI) / totalLength)):
    if math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2) >
sectionGHIMaxMoment:
        sectionGHIMaxMoment = math.sqrt(momentTangential[i] ** 2 + momentRadial[i] **
2)
sectionGHIDiameter = get_section_diameter(torque, sectionGHIMaxMoment)

sectionJMaxMoment = 0
for i in range(int(discreteSubdivisions * (totalLength - lengthK - lengthJ) /
totalLength),
               int(discreteSubdivisions * (totalLength - lengthK) / totalLength)):
    if math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2) > sectionJMaxMoment:
        sectionJMaxMoment = math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2)
sectionJDiameter = get_section_diameter(0, sectionJMaxMoment)

sectionKMaxMoment = 0
for i in range(int(discreteSubdivisions * (totalLength - lengthK) / totalLength),
               int(discreteSubdivisions * (totalLength) / totalLength)):
    if math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2) > sectionKMaxMoment:
        sectionKMaxMoment = math.sqrt(momentTangential[i] ** 2 + momentRadial[i] ** 2)
sectionKDiameter = get_section_diameter(0, sectionKMaxMoment)
```

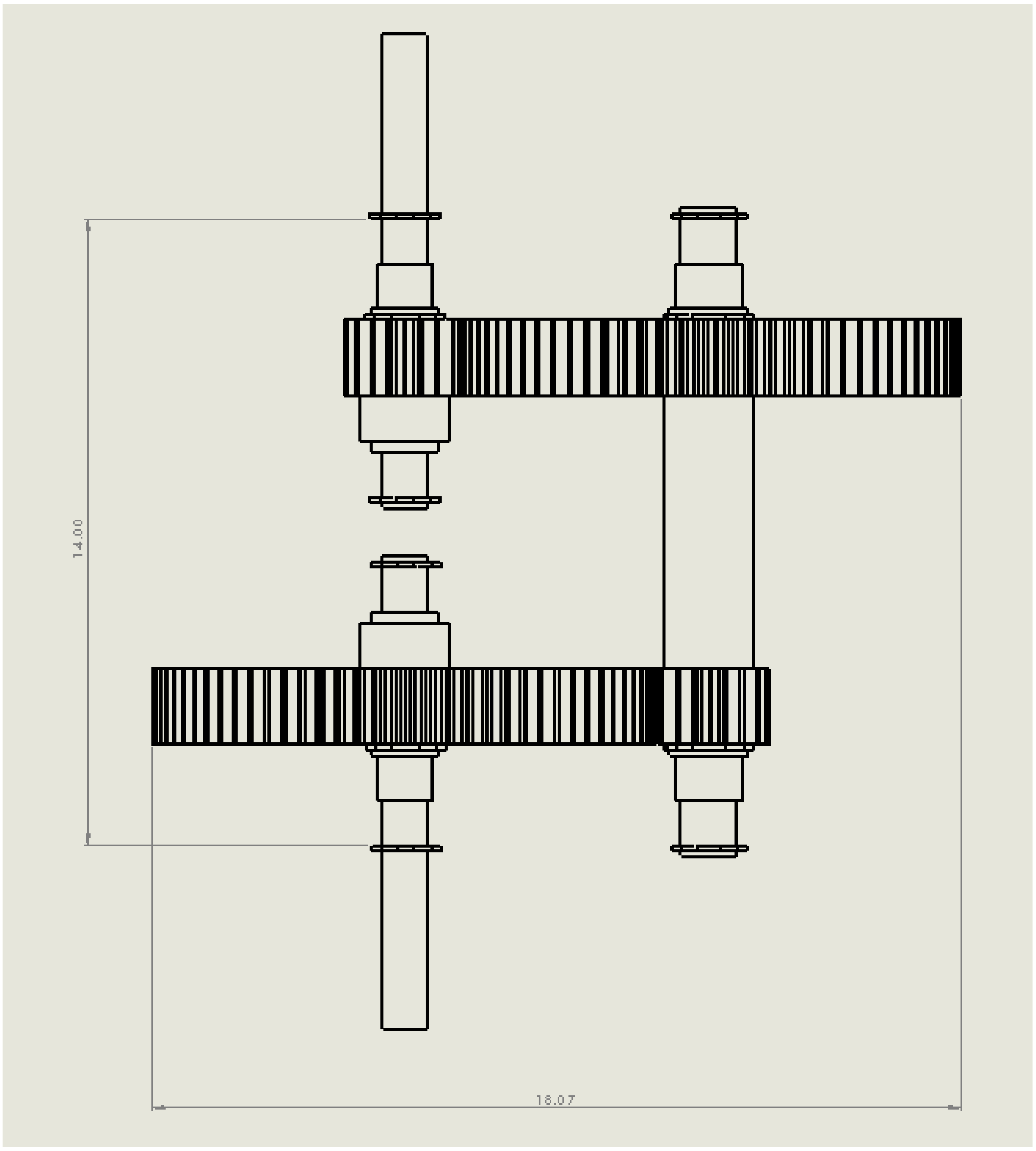
```
intermediateShaftOutputSection = [sectionADiameter, sectionBDiameter,
sectionCDEDiameter, sectionFDiameter, sectionGHIDiameter, sectionJDiameter,
sectionKDiameter]
print("14) Intermediate Shaft Diameters [in]: " + str(intermediateShaftOutputSection))

plt.show()
```

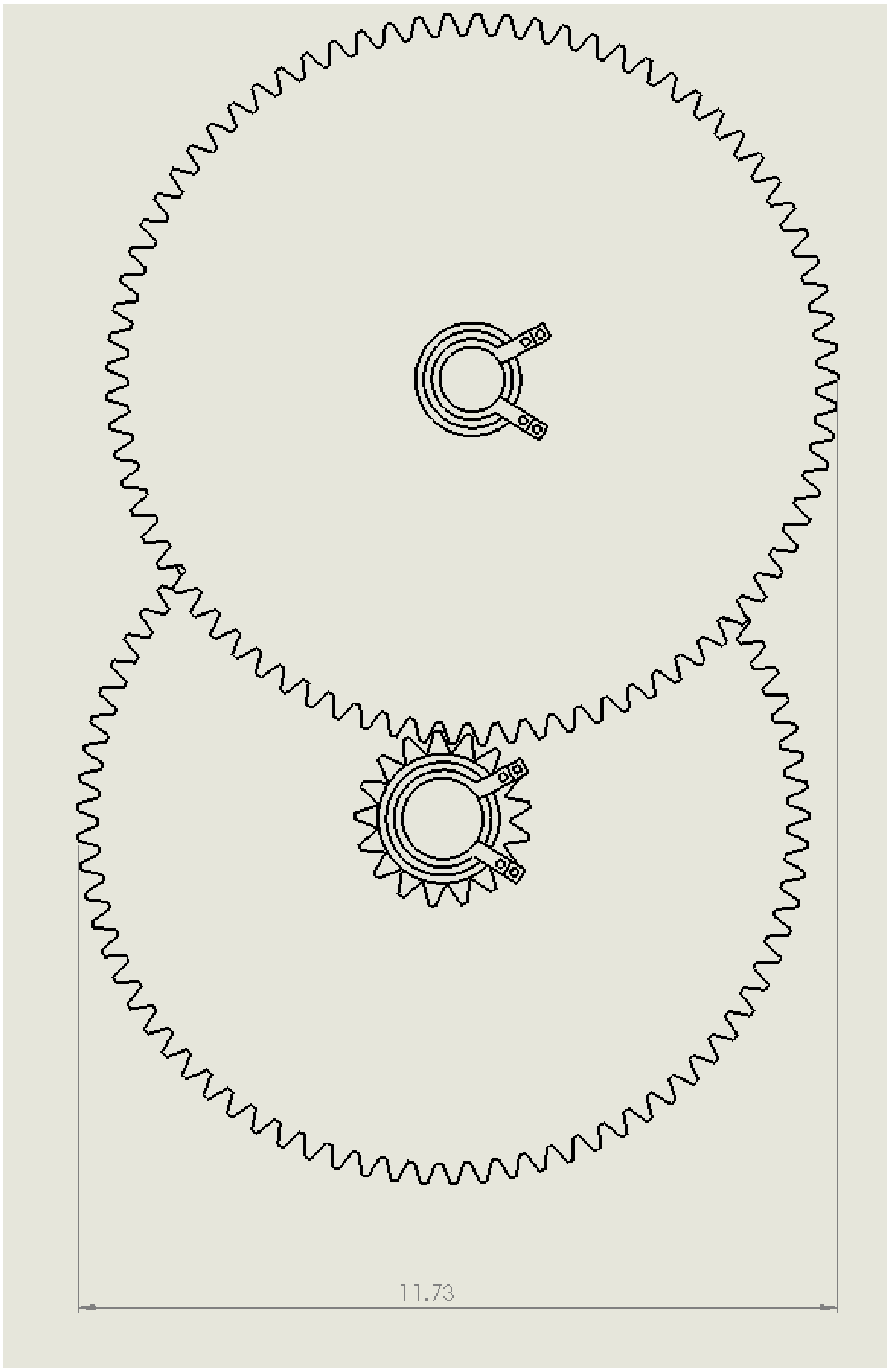
Appendix B – Dimensioned Drawings from Solidworks (in Inches)

Overall Assembly of the Speed Reducer

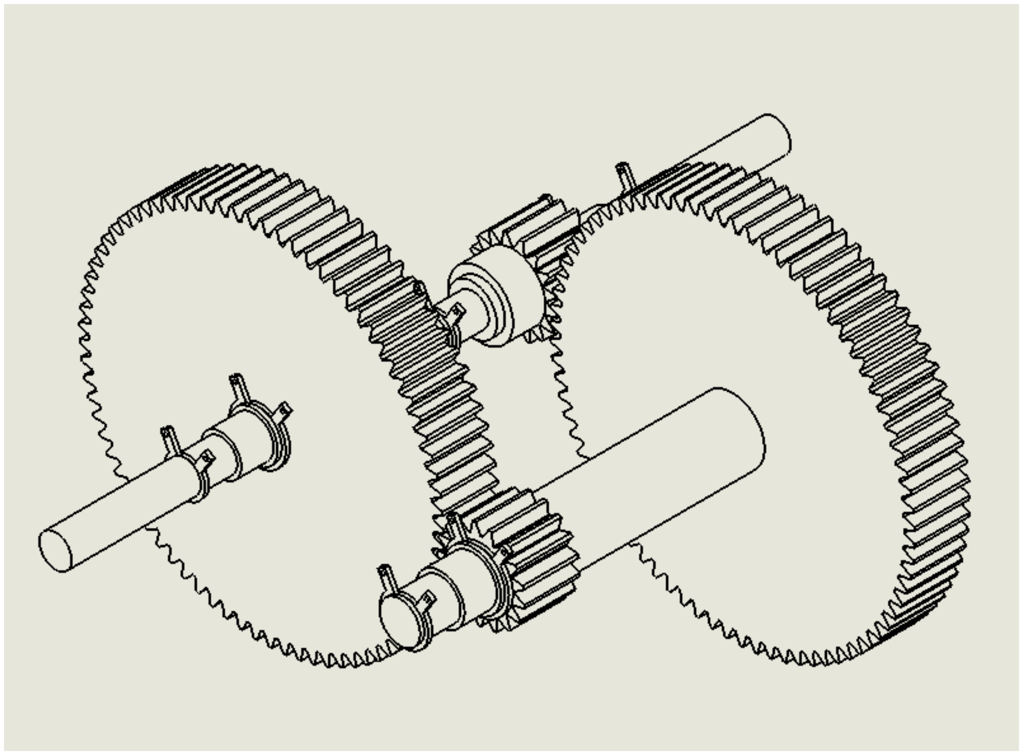
Top View showing the length of the entire assembly fitting within the dimension constraints of 22 inches and 14 inches.



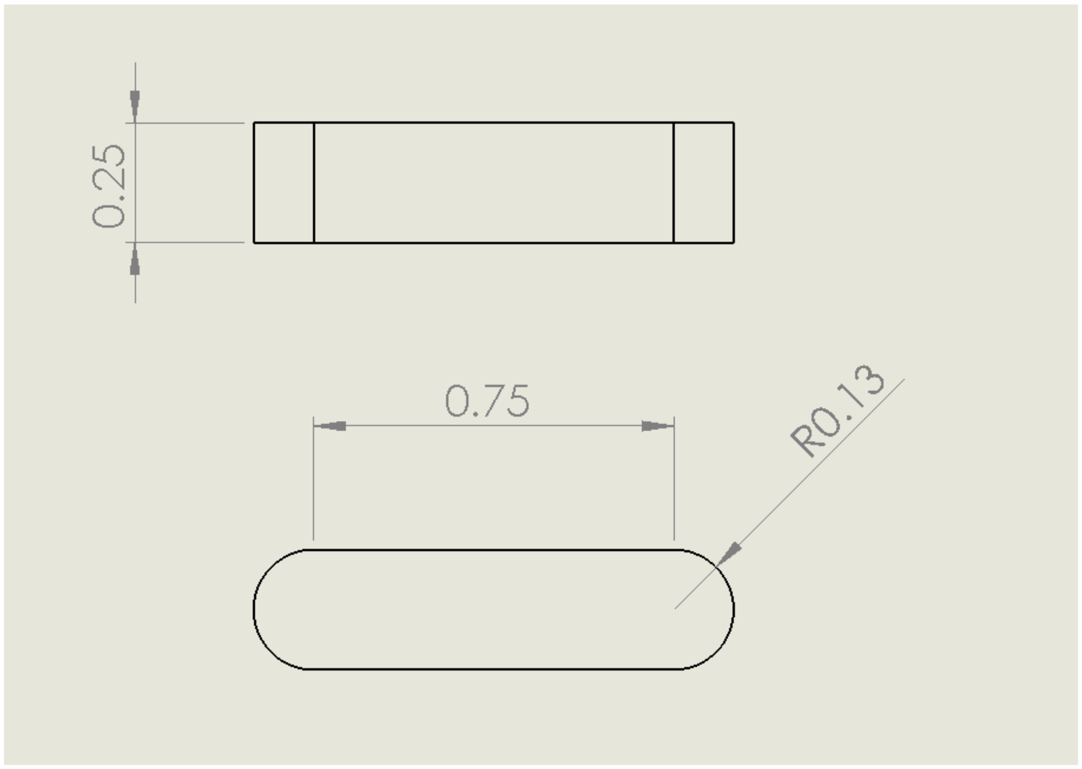
Front View showing the gears fitting within the height constraint of 14 inches.



Isometric view to show the overall assembly.

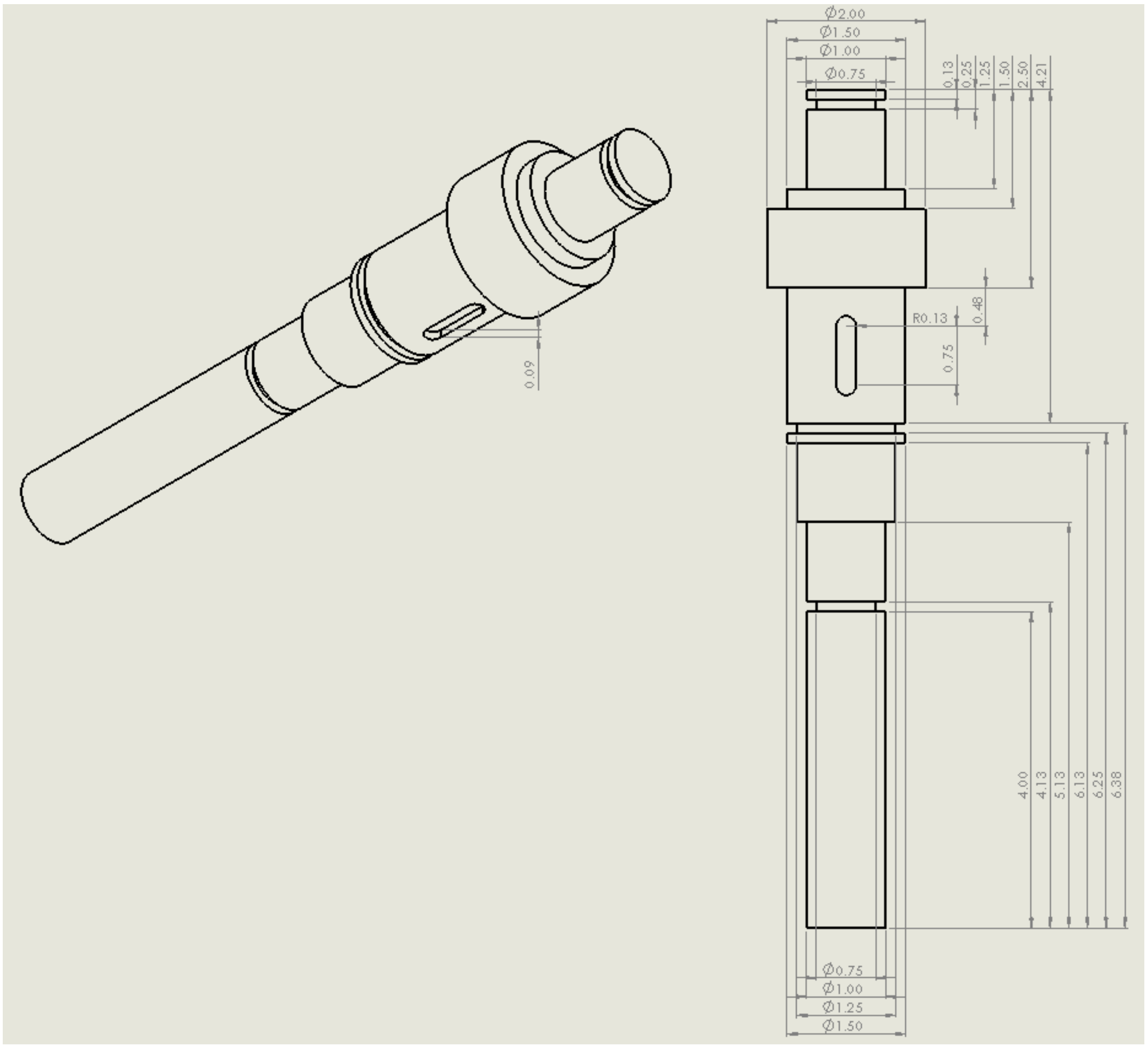


Common Key set that is used for All of the Gear to Shaft Slots

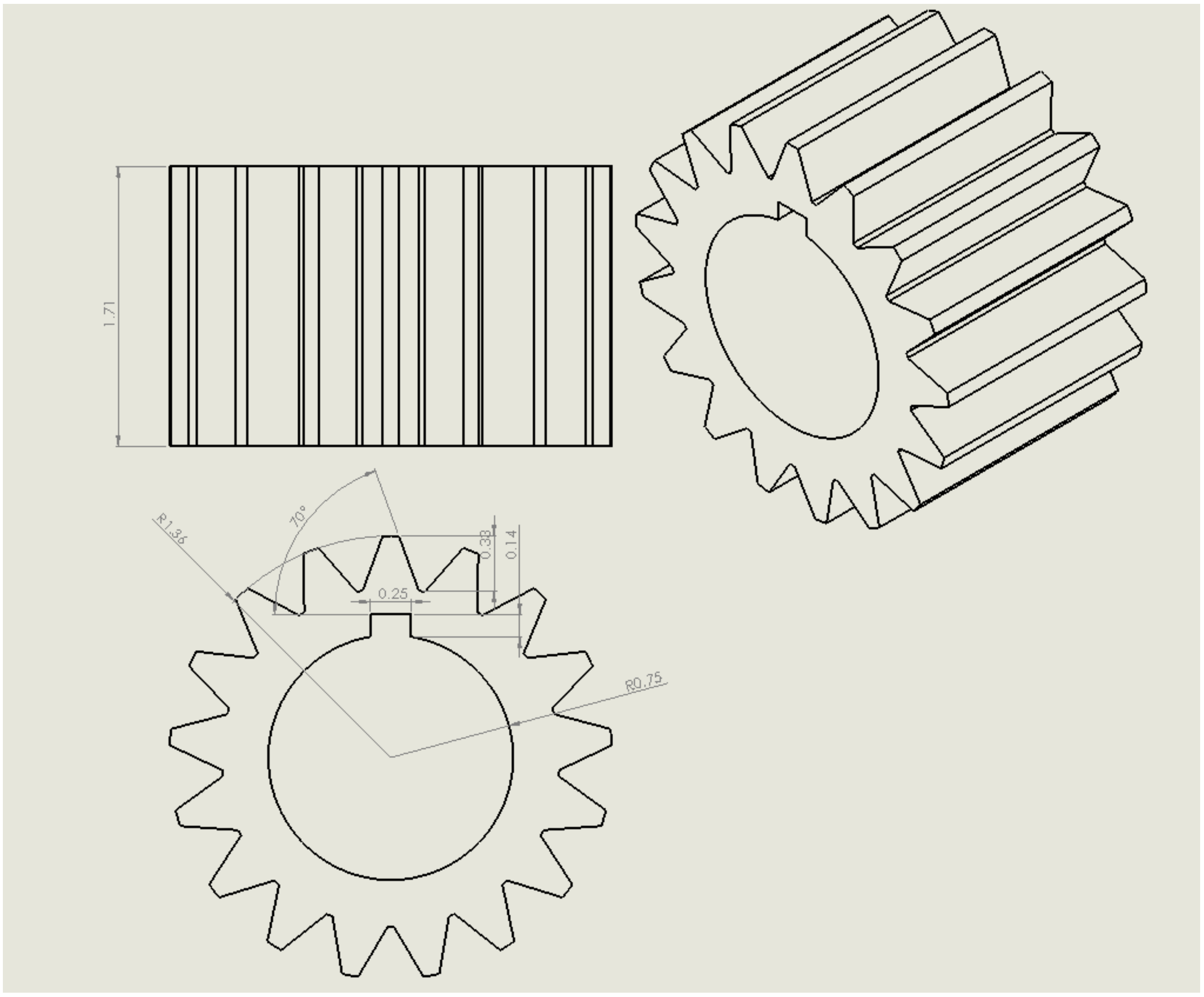


Input and Output Shaft Parts

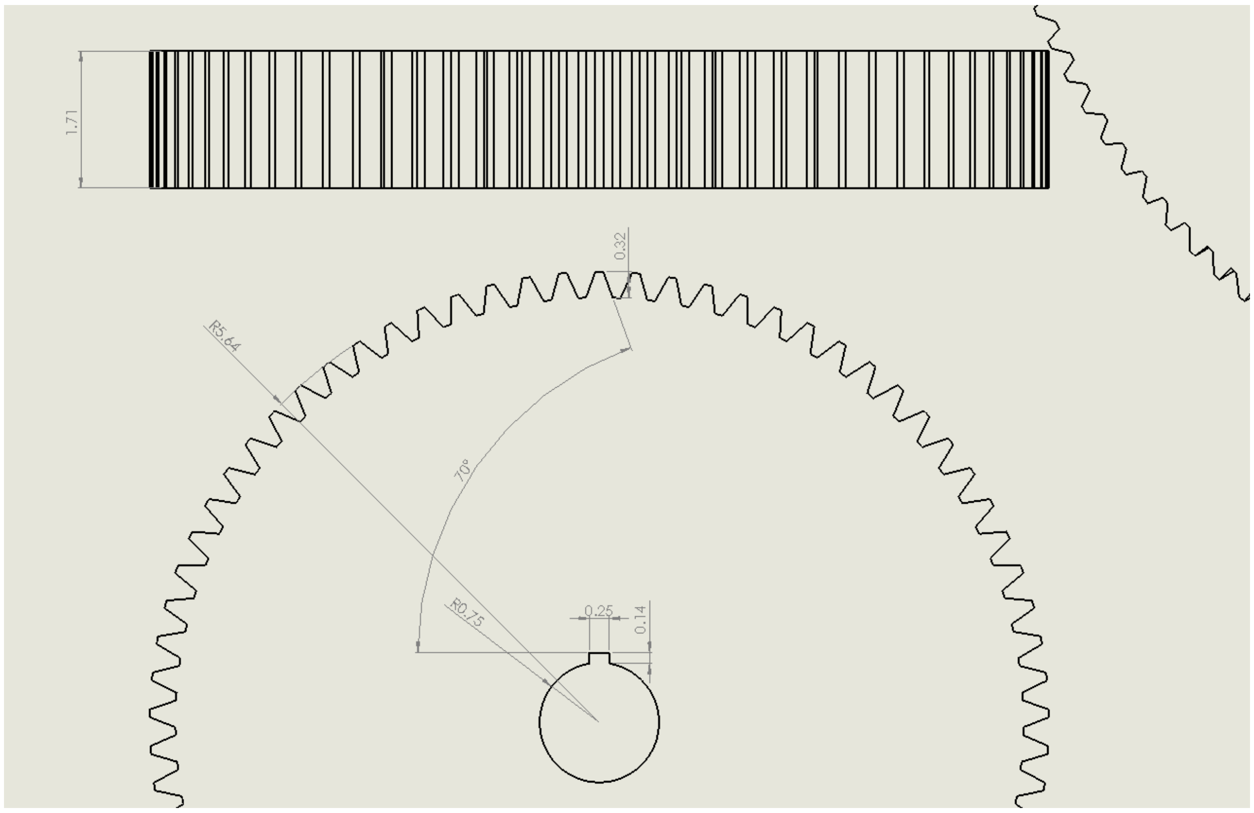
Input / Output Shaft



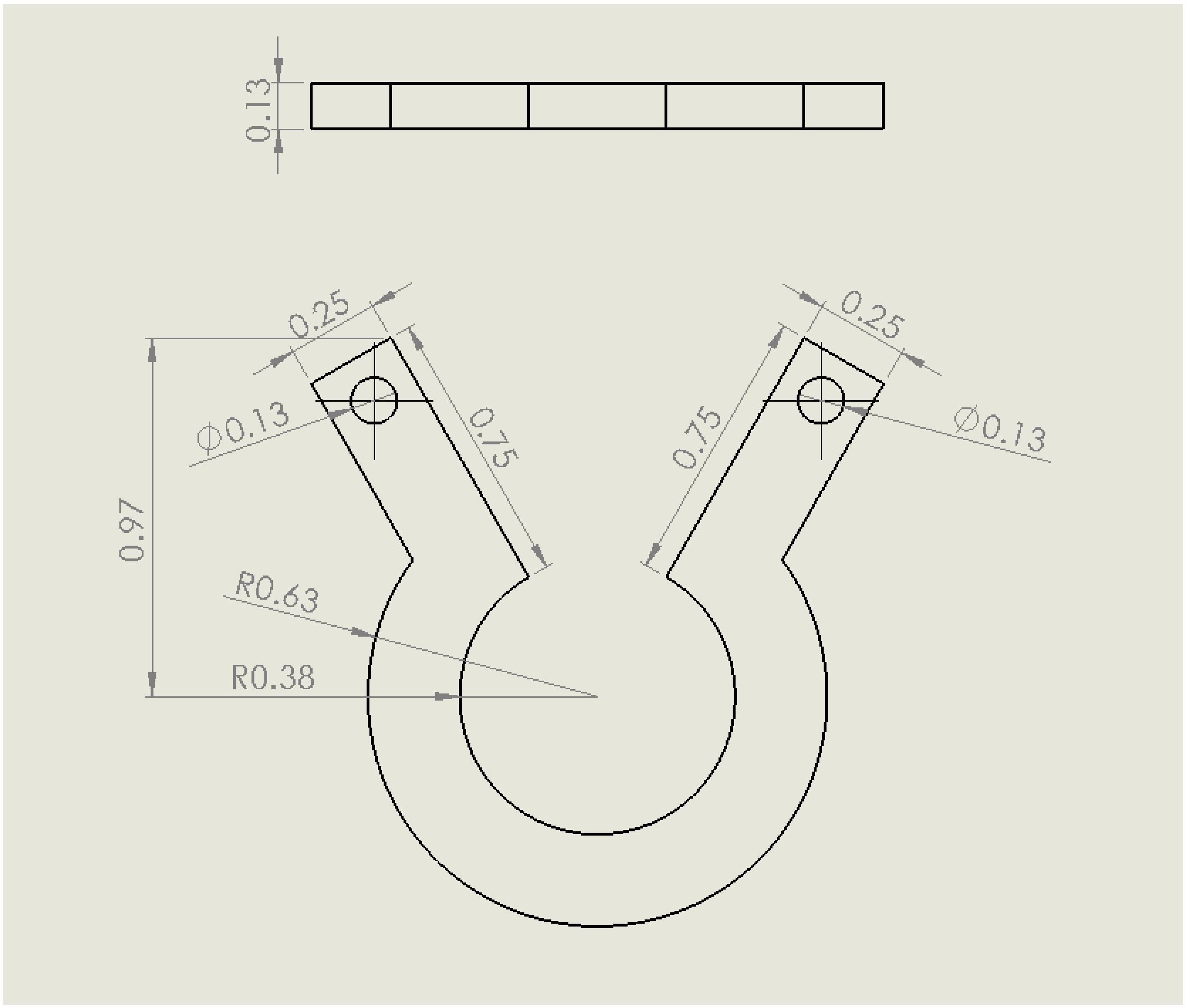
Input Shaft Pinion



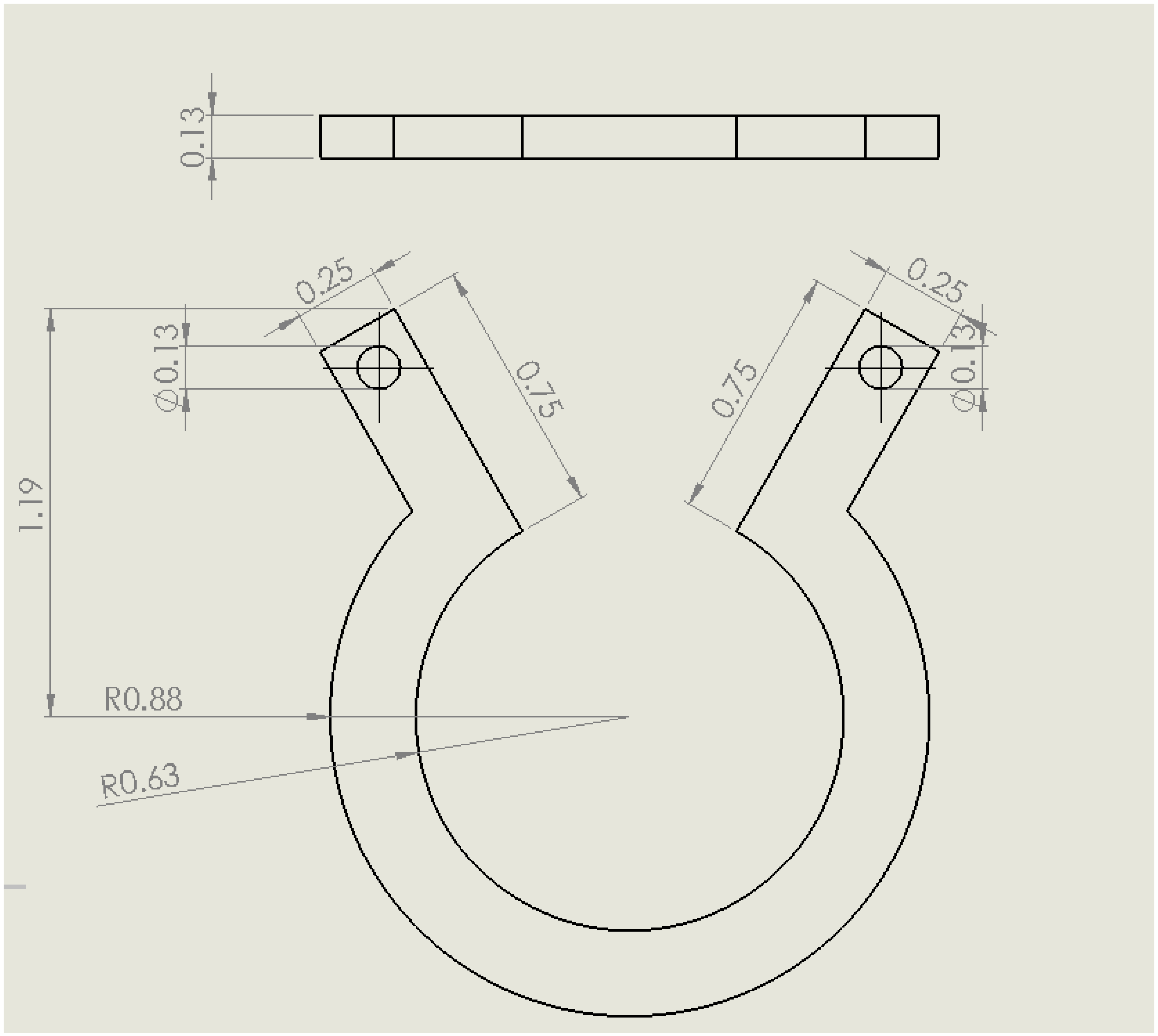
Output Shaft Gear



Retaining Rings (2x per Shaft) for affixing shaft to the supporting container walls

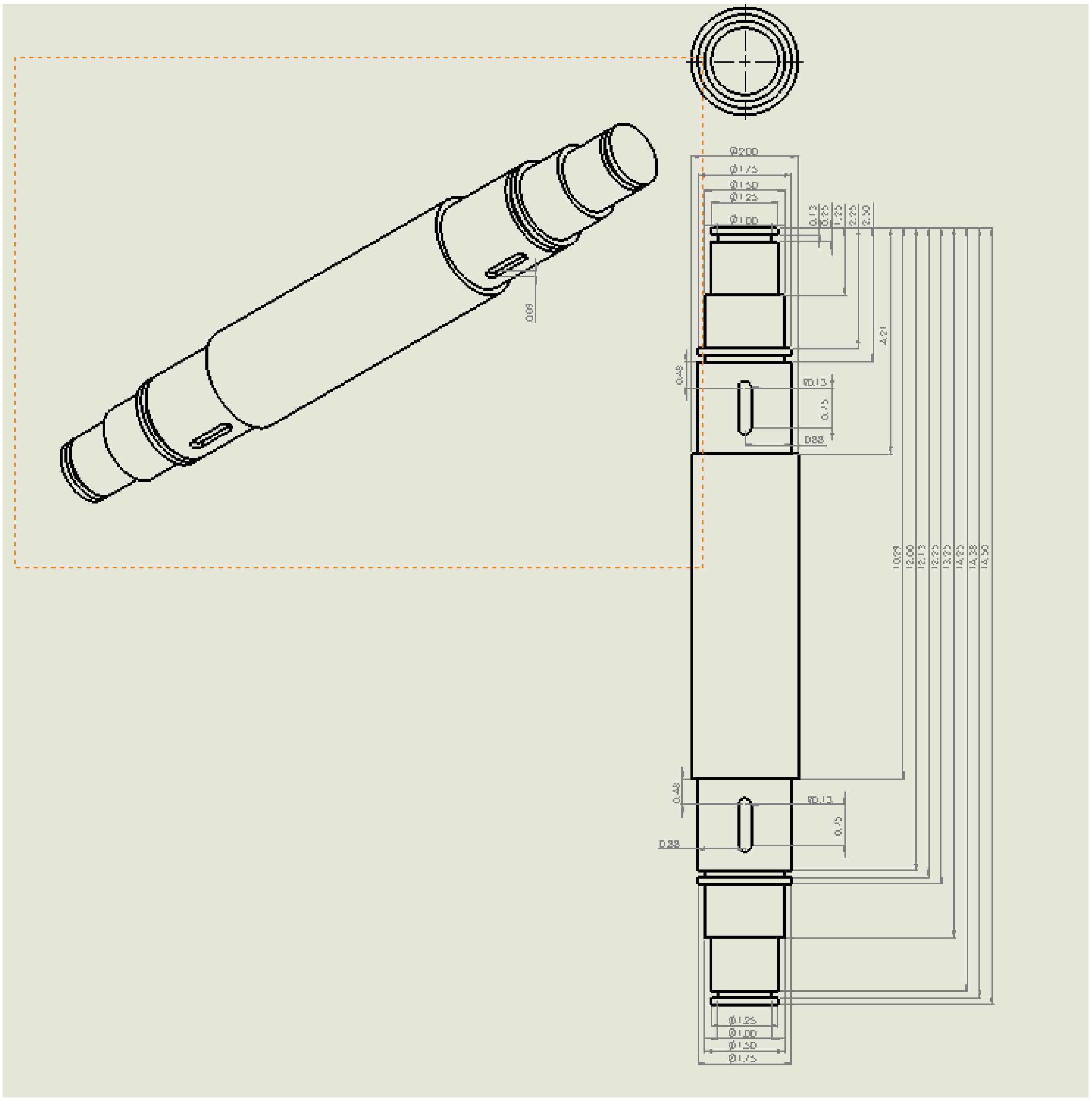


Retaining Ring (1x per Shaft) for affixing gears/pinion to arbitrary internal frame

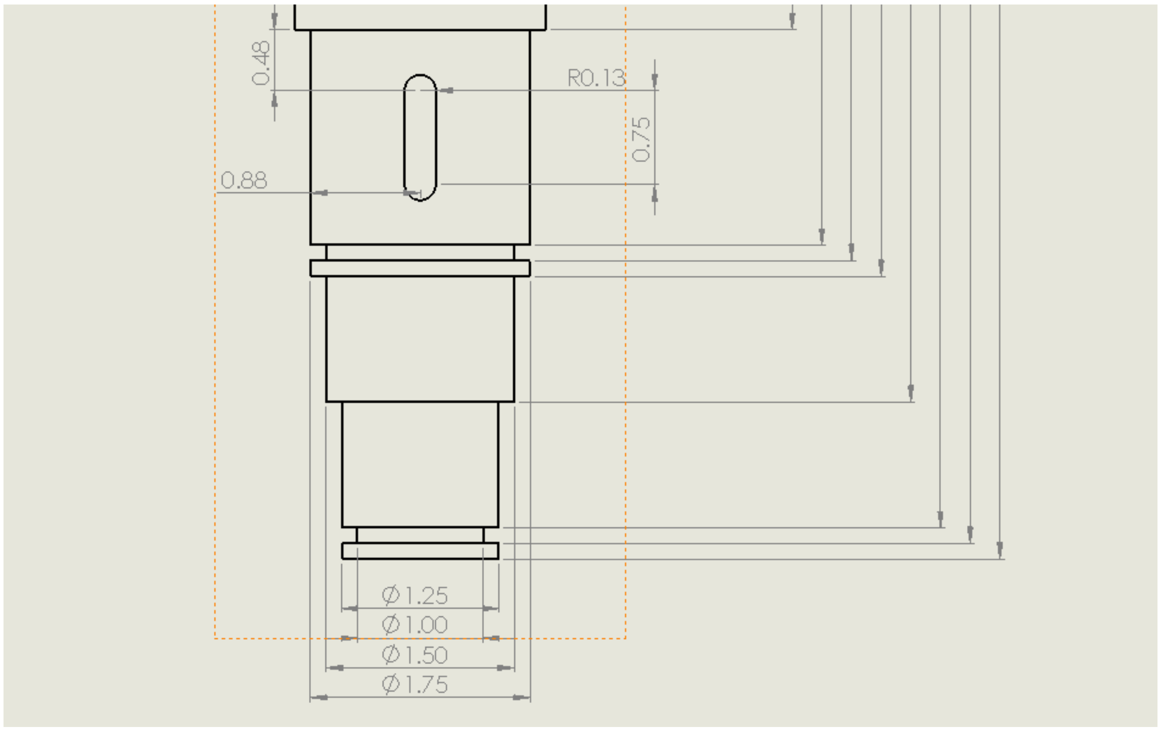


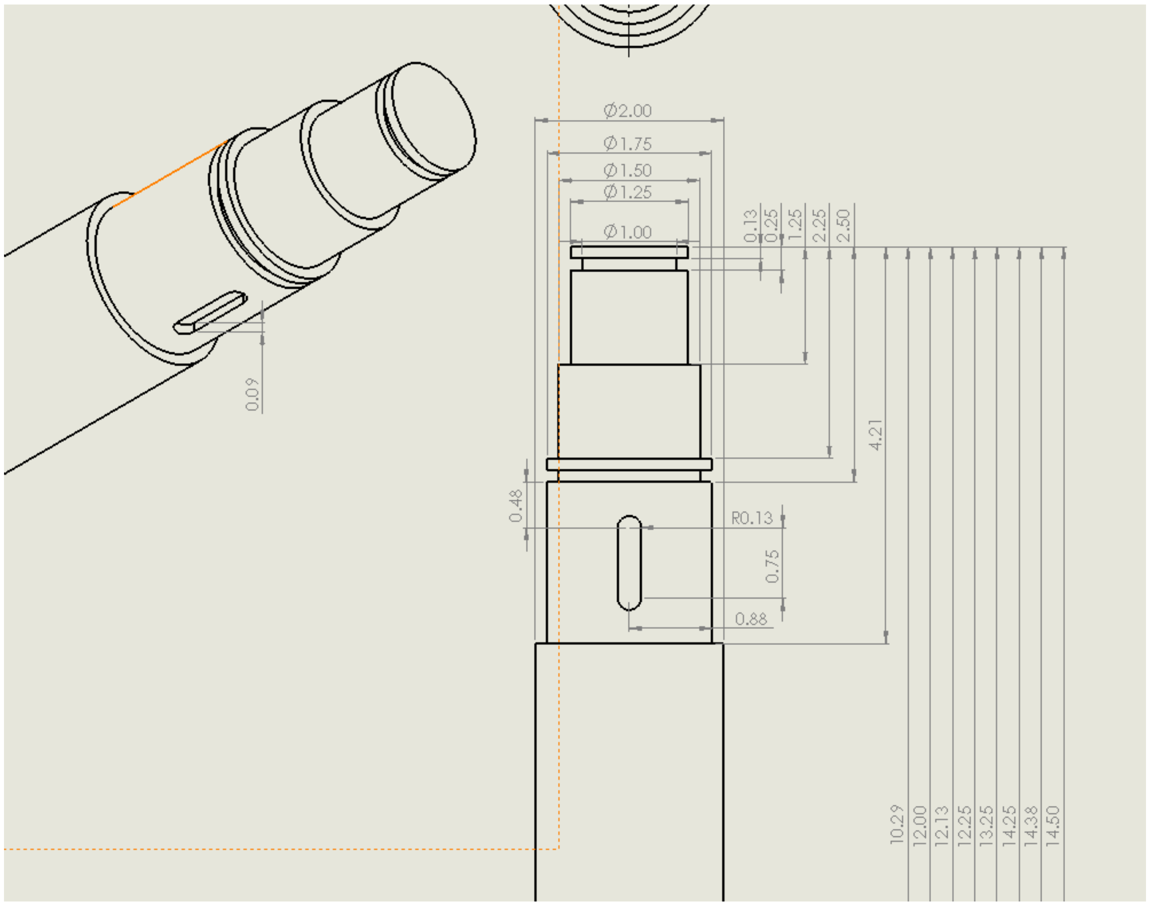
Transfer Shaft Parts

Transfer Shaft Total View

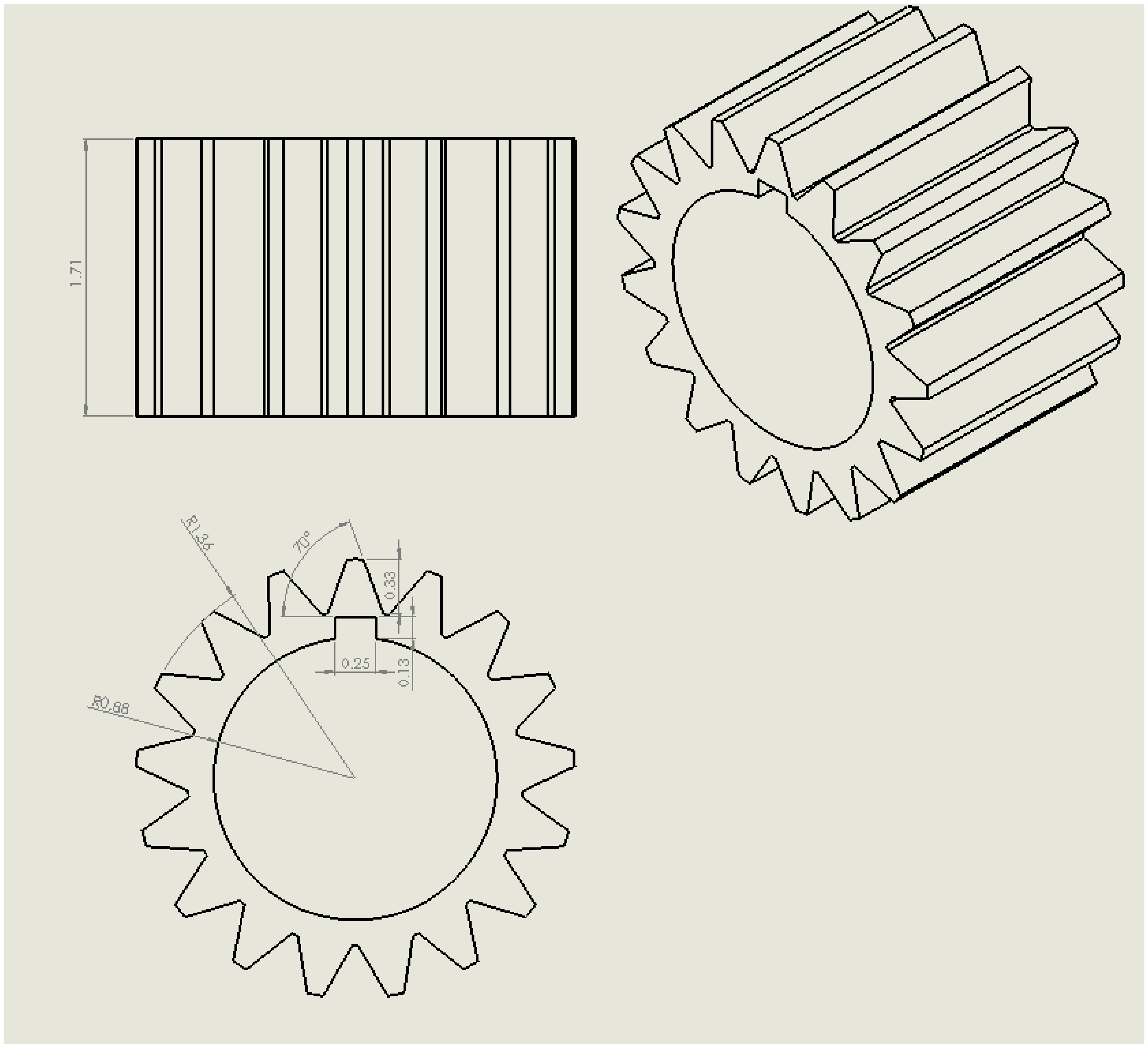


Transfer Shaft Zoomed in Views for Dimensions

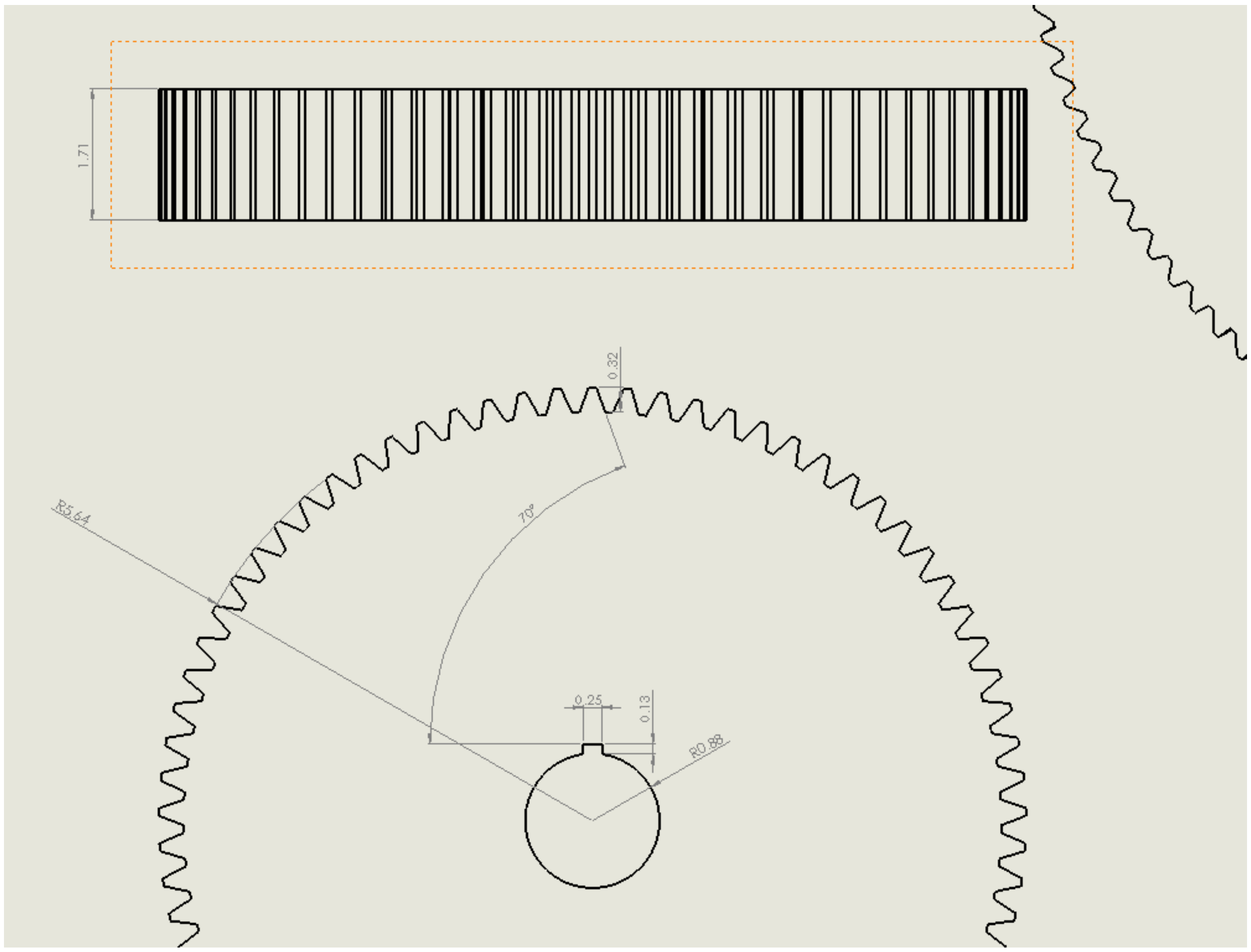




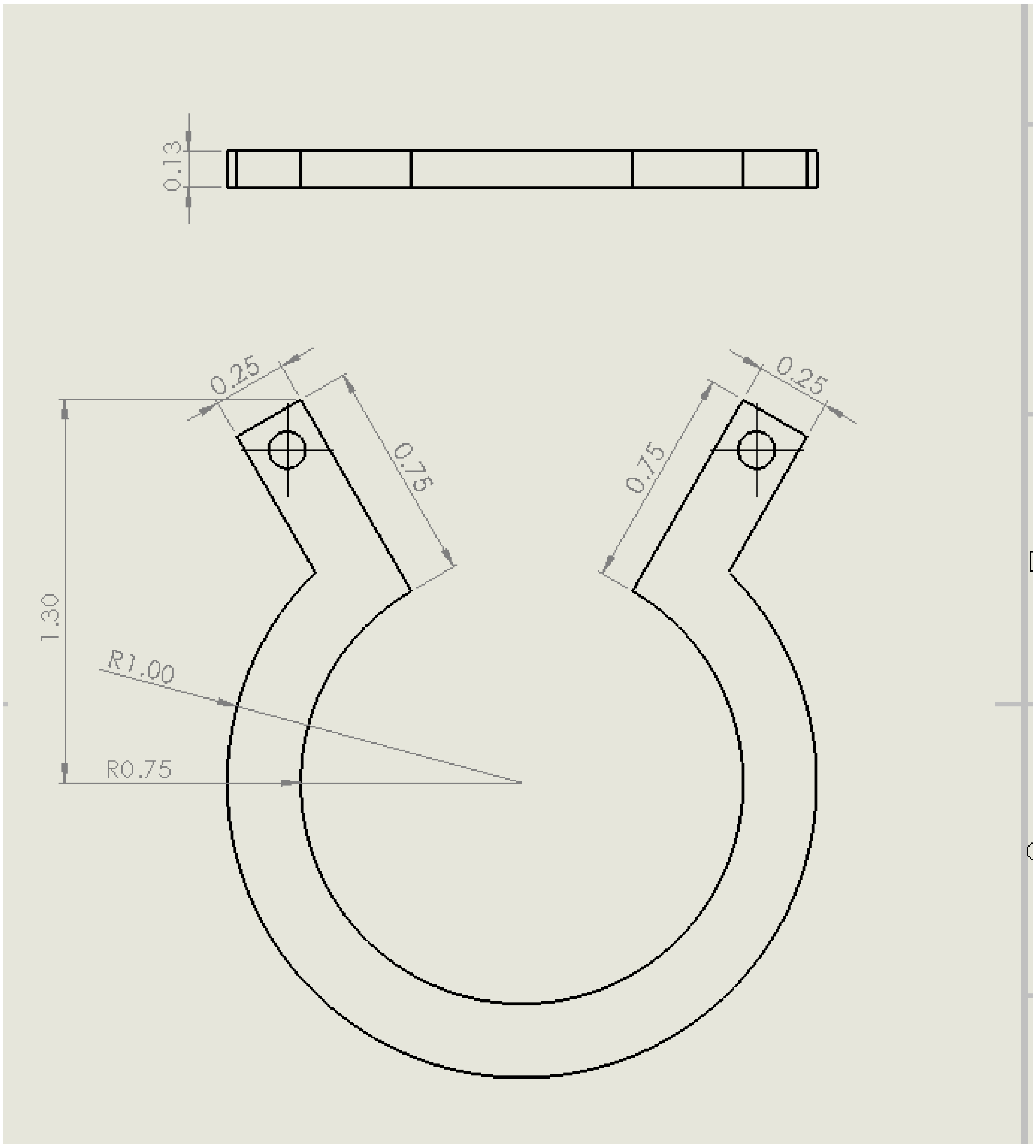
Transfer Shaft Pinion



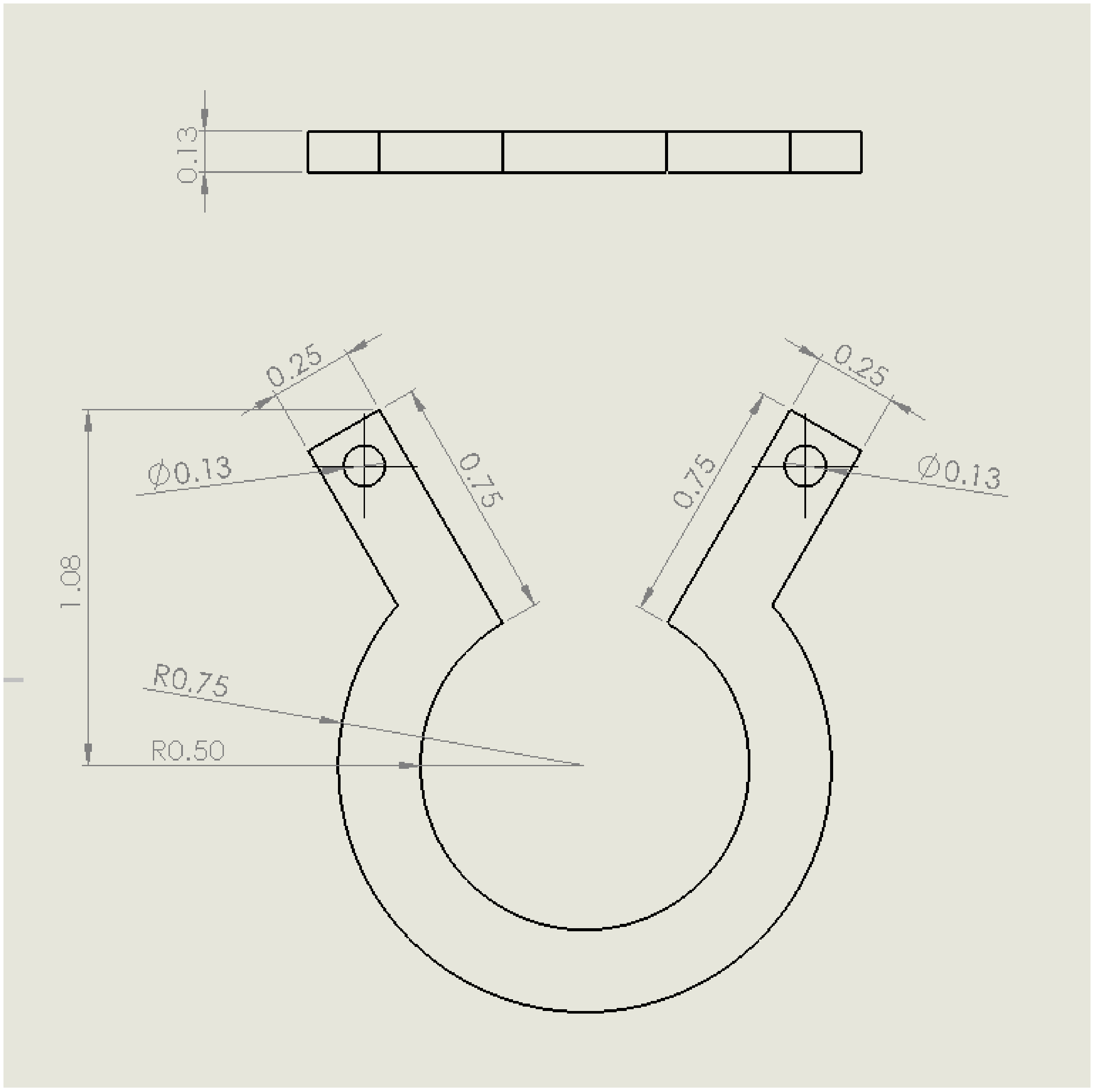
Transfer Shaft Gear



Retaining Ring for Gear / Pinion (2x)



Retaining Ring for attaching to arbitrary internal structures (2x)


0.13
0.25
0.25
0.75
0.75
⌀0.13
⌀0.13
1.08
R0.75
R0.50