MARMARA UNIVERSITY

FACULTY OF ENGINEERING

Computer-Aided Mechanical Design of Shafts

MAHMUT DEMİR

GRADUATION PROJECT REPORT

Department of Mechanical Engineering

Supervisor

Assoc. Prof. Dr. Mustafa ÖZDEMİR

ISTANBUL, 2022

MARMARA UNIVERSITY

FACULTY OF ENGINEERING

Computer-Aided Mechanical Design of Shafts

MAHMUT DEMİR

(150499981)

GRADUATION PROJECT REPORT

Department of Mechanical Engineering

Supervisor

Assoc. Prof. Dr. Mustafa ÖZDEMİR

ISTANBUL, 2022

MARMARA UNIVERSITY

FACULTY OF ENGINEERING

Computer Aided Mechanical Design of Shafts

by

Mahmut Demir

June 15, 2022, Istanbul

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

Signature of Author(s) ....Mahmut Demir

Department of Mechanical Engineering

Certified By ....

Project Supervisor, Department of Mechanical Engineering

Accepted By ....

Head of the Department of Mechanical Engineering

ACKNOWLEDGEMENT

First of all, I would like to thank my supervisor Assoc. Prof. Dr. Mustafa ÖZDEMİR, for the valuable guidance and advice on preparing this thesis and giving me moral and material support.

June, 2022 Mahmut Demir

CONTENTS

ABSTRACT

Computer Aided Mechanical Design of Shafts

Shafts are important machine elements. In the design of a shaft many calculations and iterations should be done analytically, including static and dynamic analysis, shaft deflection analysis and critical speed of the shaft. Changing the geometric layout, material type, size or loads on the shaft needs making the calculations and iterations repeatedly, which is a time consuming job. Computer-aided design offers a more efficient process, which is used successively for many years. In this project, a computer program is designed with a user-friendly graphical interface. After taking design parameters from the user, the program will perform mechanical analysis and design in accordance with the accepted standards in the field. The program executes all the calculations in a fully automated way. Discussion about used formulas is included.

SYMBOLS

A	: area
b	: shaft diameter
B_r	: sheave belt tight side to slack side ratio
D	: shaft diameter
D_{Gear}	: diameter of the gear
D_{Sheave}	: diameter of the sheave
E	: young’s modulus
I	: second moment of area
J	: polar second moment of area
k_a	: surface modification factor
k_e	: reliability modification factor
K_f	: fatigue-stress-concentration factor
K_{fs}	: fatigue-stress-concentration factor
K_t	: geometric stress-concentration factor
K_{ts}	: geometric stress conc. factor - shear stress
S_e	: fully corrected endurance limit
S_y	: yield strength
S_{ut}	: ultimate tensile strength
z_a	: transformation variate

σ_{uts}	: ultimate tensile strength
σ'	: von Mises stress
σ_a'	: alternating von Mises stress
σ_m'	: mean von Mises stress
τ	: shear stress
ω	: angular velocity

ABBREVIATIONS

ASME	**:** American Society of Mechanical Engineers
GUI	**:** Graphical User Interface

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Shaft is a machine element usually has a circular cross section that rotates and transmits power from a driving device through a machine. Shafts can carry gears, pulleys and sprockets to transmit rotary motion and power via mating gears, belts and chains. A shaft can also be stationary like the ones in automobiles. [1]

The size of the shaft can be determined by the stress analysis at a specific point on the shaft taking into account the shaft geometry at that point. Thus the geometry of the entire shaft is not needed. In design, critical areas should be located. These areas should be sized to meet the strength requirements. After that the rest of the shaft can be sized to meet the requirements of the shaft-supported elements. The deflection and slope analyses require defining the geometry of the entire shaft. We can say that deflection is a function of the entire geometry, however, the stress at a specific section is a function of local geometry. Hence, shaft design allows a consideration of stress first. After finding tentative values of the shaft dimensions, deflections and slopes analysis can be made. [2]

In this project we will use MATLAB software. Its app designer package will allow us to design a GUI for the program and the final program can be compiled as a standalone application.

2. SHAFT DESIGN

Shafts are used to transmit power and torque. Most of the motors or turbines use shaft to transfer the power. Bearings are required for the support of the shaft.

2.1. Shaft Design Considerations

The general layout of a shaft to accommodate shaft elements, which must be specified early in the design process in order to perform force analysis and to obtain shear-moment diagrams. [2]

Stepped diameters on shafts accommodate bearing mounts and are also shoulders for devices such as gears, sprockets and pulleys. [1]

Shaft design considerations include the following:

a) Size and spacing of components

b) Material selection

c) Deflection and rigidity (Bending and torsional deflection, slope at bearings, shear deflection.)

d) Stress and strength (Static strength, fatigue, reliability.)

e) Frequency response [1]

2.2. Shaft Design for Strength and Rigidity

The design procedure of the shaft can be summarized as in Figure 2.1.

- Determining the shaft rotational speed.
- Determining the power or torque to be transmitted by the shaft.
- Determining the dimensions of the components to be mounted on the shaft.
- Specifying the locations of the bearings to support the shaft.
- Determining the magnitude of the torques for the entire shaft.

- Determining the forces on the shaft.
- Determining the distribution of bending moments in the shaft.
- Determining the material of the shaft and type of shaft loading.
- Determining all the critical points on the shaft and finding the minimum diameter at each point to ensure a safe design.
- Determining the deflections of the shaft at critical points.
- Specifying the final dimensions of the shaft. [1]

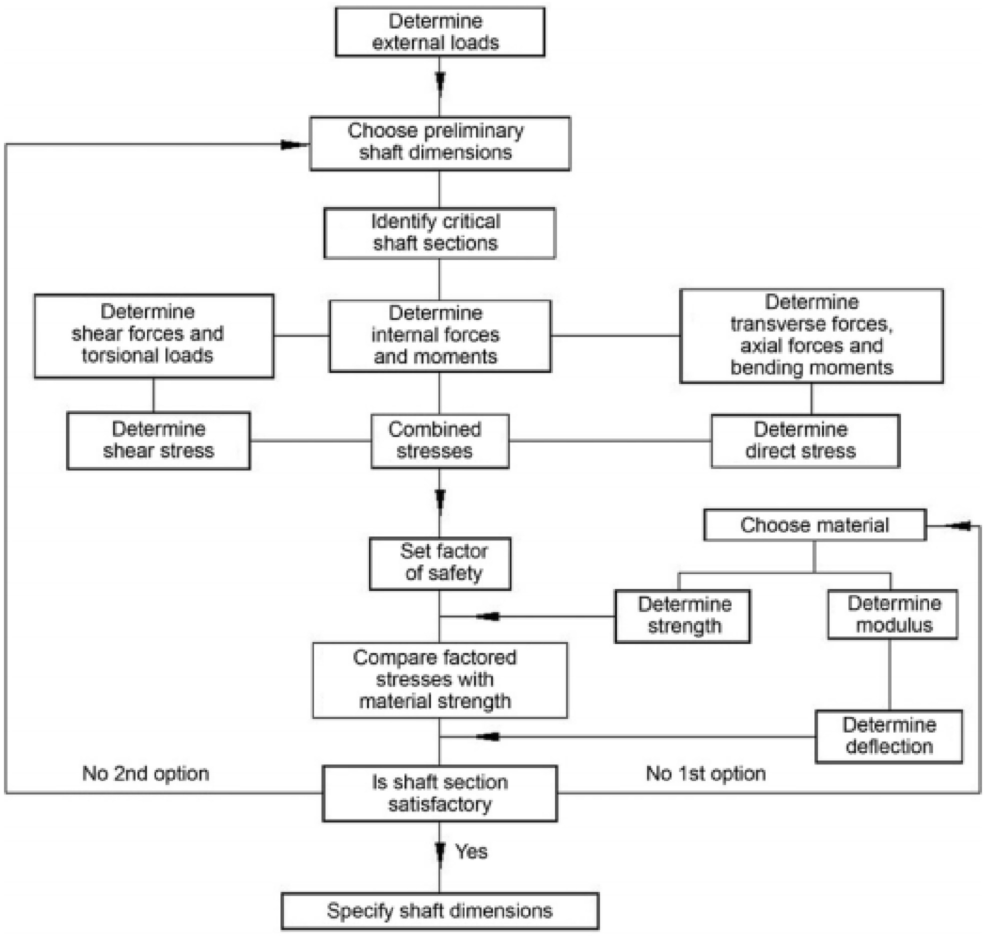

Determine external loads
Choose preliminary shaft dimensions
Identify critical shaft sections
Determine shear forces and torsional loads
Determine internal forces and moments
Determine transverse forces, axial forces and bending moments
Determine shear stress
Combined stresses
Determine direct stress
Choose material
Set factor of safety
Determine strength
Determine modulus
Compare factored stresses with material strength
Determine deflection
No 2nd option
Is shaft section satisfactory
No 1st option
Yes
Specify shaft dimensions

Figure 2.1 Design procedure flow chart of a shaft for strength and rigidity. [1]

3. LITERATURE REVIEW

3.1. Development of CAD Software for Shaft Under Various Loading Conditions

Adekunle et al [3] studied to design shaft under various loading conditions using Computer Aided Design. The Programming language used was Visual Studio C#.

Tasks their software claimed to solve

- Determine the right diameters of shafts to withstand the various loading conditions.
- Based on the preliminary design of the minimum diameter, technological and functional.
- Requirements, a design of the shaft shape can be made.
- Calculation of equivalent stress in the shaft.
- Coefficient of static safety of the shaft.
- Coefficient of dynamic safety of the shaft.

Required Inputs

- Power transmitted by the shaft (kW).
- Shaft speed (rpm).
- Torsional moment (Nm).
- Bending moment (Nm).
- Preliminary diameter (mm).
- The type of shaft load.
- Type of material to be used.

Their shaft design is based on deflection and rigidity & stress and strength. Strength is utilized with modified Goodman strength criteria. The software has a graphical user interface. A material selection field where the type of materials to be used and their properties would be selected. A field where shaft shape and dimensions can be selected. The result section consists of the torsional moment, bending moments, equivalent strength, and minimum shaft diameter, coefficient of static safety and coefficient of dynamic safety.

3.2. Design of Machine Shaft in Fatigue Loading by Using C++ Programming Language

Saradava et al [4] wrote a program in C++ for a shaft under fatigue load. They use ASME code for the design of transmission shaft and define the maximum permissible working stress in tension or compression accordingly. Their design of shaft based on strength as;

- Shaft subjected to twisting moment only
- Shaft subjected to bending moment only
- Shaft subjected to combined twisting and bending moment
- Shaft subjected to axial loads in addition to combined torsional and bending loads

They find diameters of the shaft due to bending and twisting moments and take the maximum diameter for the design of the shaft. They also included a systematic procedure and a C++ code in their study.

3.3. Development of a Computer Aided Software for Power Transmission Shaft Design with Multiple Criteria

Fadare et al [5] studied on a transmission shaft software with multiple criteria. The programming language used was Visual Basic 2008 and the program has an interface designed with an image design software Fireworks. They use five design criteria; strength, torsional rigidity, critical speed or vibration, Soderberg and lateral rigidity.

3.4. MITCalc Software

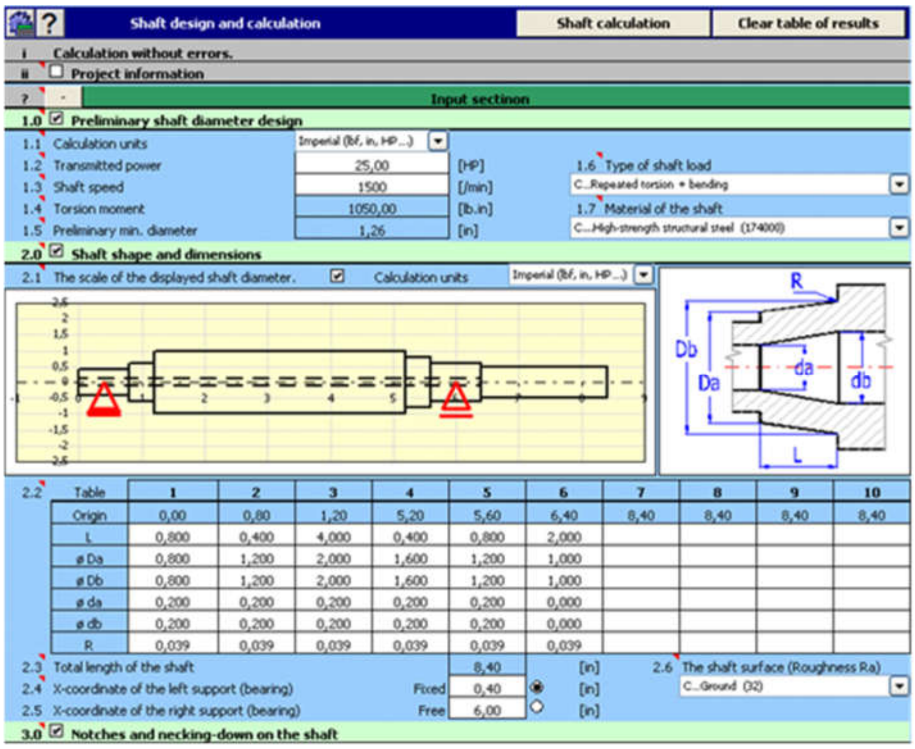
MITCalc is a commercial brand name of a software that has a shaft analysis module. The aim of the shaft module is analyzing geometrical designs and complex examinations of shafts. The software solves the following tasks:

- Simple definition of installed shafts, including hollow ones.
- Options of definitions of necking-down, recesses, grooves and calculation of the relevant coefficients of stress concentration.
- Simple definition of spatial shaft load.
- Calculation of reactions, courses of forces, moments, stress, deflection and bending angle of the shaft and others.
- Calculation of critical speed and safety coefficients (dynamic, static).

- Support of 2D and 3D CAD systems.

It demands from the user to enter values for the preliminary shaft diameter design.

It has an option for Imperial vs SI Units selection. After entering transmitted power and shaft speed, torsion moment and preliminary minimum diameter are found by the software. Material selection and different types of load selection are chosen by the user.



i Calculation without errors.

ii ☐ Project information

1.0 ☑ Preliminary shaft diameter design

1.1 Calculation units	Imperial (lbf, in, HP...)			
1.2 Transmitted power	25,00	[HP]	1.6 Type of shaft load	
1.3 Shaft speed	1500	[/min]	C...Repeated torsion + bending	
1.4 Torsion moment	1050,00	[lb.in]	1.7 Material of the shaft	
1.5 Preliminary min. diameter	1,26	[in]	C...High-strength structural steel (174000)	

2.0 ☑ Shaft shape and dimensions

2.1 The scale of the displayed shaft diameter. ☑ Calculation units Imperial (lbf, in, HP...)

2.2

Table	1	2	3	4	5	6	7	8	9	10
Origin	0,00	0,80	1,20	5,20	5,60	6,40	8,40	8,40	8,40	8,40
L	0,800	0,400	4,000	0,400	0,800	2,000				
ø Da	0,800	1,200	2,000	1,600	1,200	1,000				
ø Db	0,800	1,200	2,000	1,600	1,200	1,000				
ø da	0,200	0,200	0,200	0,200	0,200	0,000				
ø db	0,200	0,200	0,200	0,200	0,200	0,000				
R	0,039	0,039	0,039	0,039	0,039	0,039				

2.3 Total length of the shaft		8,40		[in]
2.4 X-coordinate of the left support (bearing)	Fixed	0,40	●	[in]
2.5 X-coordinate of the right support (bearing)	Free	6,00	○	[in]

2.6 The shaft surface (Roughness Ra) C...Ground (32)

3.0 ☑ Notches and necking-down on the shaft

Figure 3.1 MITCalc Software GUI. [6]

User then entered geometric dimensions of the shaft. There is also a selection for the shaft surface characteristic. In the notches and necking-down section, user can define geometric stress concentrations on the shaft. In the loading of the shaft section user defines loads on the shaft. If there are additional rotational masses, the user can define it. The software also has a database for materials and an advanced type of loading section. The user can also change values individually. Results section offers a wide variety of solutions. Reaction in the support, maximum deflection, angular deflection at bearings, maximum stress, minimum static safety, minimum dynamic safety, critical speed, results for X coordinate, and graphics.

4. PROGRAM DESIGN PROCEDURE

4.1. About MATLAB App Designer

MATLAB is a multilingual programming language and digital computing environment. [7] MATLAB App Designer lets user to develop apps interactively. It has the ability to pack finished app and share it as a standalone desktop app that can run without the need of MATLAB software.

4.2. Elements of the Shaft

The elements of a shaft can be gears, bearings and pulleys. In the app, shaft elements restricted to maximum number of two bearings, two gears and two sheaves.

4.3. Assumptions

The shaft assumed to be rotating between two bearings.

Gears are assumed spur gears where no axial force exerted along X-axis.

Belts on pulleys assumed to be wrapped at 180° angle.

Shaft mass is ignored.

Shaft is assumed to be a beam with left end is fixed and right end is free.

Assuming the torque is steady.

Assuming the torque is only present between the gears and/or pulleys. If the bearing on the right side is on between the gears and/or pulleys there is a checkmark to state the situation.

4.4. Input Loads on Elements of the Shaft

Gears and belts apply transverse forces on the shaft. These forces can be on different planes. X-axis taken as the axis along the length of the shaft, and Y-axis as vertical axis.

For gear loads, forces assumed about Z-axis, gear pressure angle asked from the user and weight of the gear is an option as an input if it is mandatory.

For pulley loads, belt force tight side to slack side ratio, weight of the pulley asked from the user.

4.5. Calculating Torque on the Shaft

The power transmitted through the shaft and shaft speed is asked from the user. Then the torque is calculated from the equation Eq. 4.1.

$$Torque = \frac{Power}{\omega} \qquad \text{(Eq. 4.1)}$$

The code used is;

```
% Rotational Velocity
app.RotVelocity = 2 * app.Speed.Value * pi / 60; % in radians / second
% Torque
app.Torque = (app.Power.Value * 1000) / app.RotVelocity; % in Newton-millimeters
```

Where;

app.Speed.Value: Input speed of the shaft.

app.Power.Value: Input power of the shaft.

4.6. Calculating Forces on Sheaves

The forces on sheaves that bends the shaft are calculated from Eq. 4.2.

$$Sheave\ Force = \frac{(B_r + 1)}{(B_r - 1)} \times \frac{Torque}{\frac{D_{Sheave}}{2}} \qquad \text{(Eq. 4.2)}$$

The code used is;

```
% Determining Forces on the Sheave 1
if (app.R_Sheave1.Value==0)
```

```
    app.Fsheave1_Z = 0;
elseif (app.R_Sheave1.Value==1)
    app.Fsheave1_Z = 0;
elseif (app.D_Sheave1.Value==0)
    app.Fsheave1_Z = 0;
else
app.Fsheave1_Z = ((app.R_Sheave1.Value + 1) / (app.R_Sheave1.Value - 1)) * app.Torque /
(app.D_Sheave1.Value / 2); % in Newtons
end
app.Fsheave1_Y = app.ForceSheave1Y.Value * 9.81;
```

Where;

app.R_Sheave1.Value: Pulley 1's belt tight side to slack side ratio. If user not entered the value, it is assumed that Pulley 1 is not used. If user entered the value as 1, force is calculated as zero not to stuck the app.

app.D_Sheave1.Value: Diameter of sheave 1.

app.ForceSheave1Y.Value: Weight of the pulley in kg.

4.7. Calculating Forces on Gears

The forces on gears about Z-axis are calculated from Eq. 4.3.

$$Gear\ Force\ on\ Z\ = \frac{Torque}{\frac{D_{Gear}}{2}} \qquad \text{(Eq. 4.3)}$$

The forces on the gears about Y-axis are calculated from Eq. 4.4.

$$Gear\ Force\ on\ Y\ = Gear\ Force\ on\ Z \times \tan(Pressure\ Angle) \qquad \text{(Eq. 4.4)}$$

The code used is;

```
% Determining Forces on Gear 1
if(app.D_Gear1.Value==0)
   app.Fgear1_Z=0;
   app.Fgear1_Y=0;
else
app.Fgear1_Z = app.Torque / (app.D_Gear1.Value / 2);
app.Fgear1_Y = app.Fgear1_Z * tand(app.PA_Gear1.Value)+app.ForceGear1Y.Value*9.81;
end
```

Where;

app.D_Gear1.Value: Diameter of gear 1.

app.ForceGear1Y.Value: Weight of the gear.

4.8. Calculating Moments at Critical Points Function

The function takes loads and their locations and find the reaction force on the right support bearing. Then evaluate the moment at the location of interest.

The code of the function;

```
% Function that computes moments on a shaft based on loads and their locations where;
%Inputs: X is the calculated moment location, Load is locations and amounts of the loads, R1 is the location of left support bearing and R2 is the location of the right support bearing.
%Outputs: Reaction1 is the reaction force at left bearing, Reaction2 is the reaction force at right bearing and M is the moment. %
function [M, Reaction1, Reaction2] = ShaftMoment(app, X, Load, R1, R2)
numberLoads = size(Load);        % number of loads
mmnt = 0;
% Finding the force at support 2
for i = 1: numberLoads(1)
     mmnt = mmnt + Load(i,2) * (Load(i,1) - R1);
end
Reaction2 = - mmnt / (R2 - R1);
%  Finding the force at support 1
```

```
Reaction1 = - Reaction2;
for i = 1: numberLoads(1)
    Reaction1 = Reaction1 - Load(i, 2);
end
% Finding moment at X
M = 0;
for i = 1: numberLoads(1)
    if (X > Load(i,1))
        M = M + Load(i,2) * (X - Load(i,1));
    end
end
%Additional moments from reactions
if (X > R1)
    M = M + Reaction1 * (X - R1);
end
if (X > R2)
    M = M + Reaction2 * (X - R2);
end
end
```

4.9. Calculating Moments at Critical Points

Loads and locations on both planes are taken inside matrixes and evaluated at the point of interest.

The code is;

```
% Moments at critical points in both planes
%Input Loads;
app.LoadZ = [app.Gear1Dist.Value, app.Fgear1_Z; app.Gear2Dist.Value, app.Fgear2_Z;
app.Pulley1XDist.Value, app.Fsheave1_Z; app.Pulley2XDist.Value, app.Fsheave2_Z];

app.LoadY = [app.Gear1Dist.Value, app.Fgear1_Y; app.Gear2Dist.Value, app.Fgear2_Y; app.Pulley1XDist.Value, app.Fsheave1_Y;app.Pulley2XDist.Value, app.Fsheave2_Y];

% Finding the Moment at the Position of Bearing 1

[app.ZMomentBear1, app.Reaction1, app.Reaction2] = ShaftMoment(app, 0, app.LoadZ, app.LSupport.Value, app.RSupport.Value);

[app.YMomentBear1, app.Reaction1, app.Reaction2] = ShaftMoment(app, 0, app.LoadY, app.LSupport.Value, app.RSupport.Value);

app.MomentBear1 = sqrt(app.ZMomentBear1 ^2 + app.YMomentBear1 ^2);
```

Where;

app.MomentBear1: Resultant moment on left bearing.

4.10. Calculating Endurance Limit Modifying Factors

Endurance limit of the specimens are tested in a laboratory environment with care, however, in reality it is less than the value founded. So, modification of the test results needed.

4.10.1. Surface Factor

Shafts are polished rotating beams and finishing methods change the strength value of the specimen. The surface factor is calculated from Eq. 4.5. [2]

$$k_a = a \times S_{ut}^{b} \qquad \text{(Eq. 4.5)}$$

The code to find surface factor is;

```
% Curve Fit Parameters for Surface Factor

app.surface1 = app.csurfinp.Value;

if strcmpi( app.surface1,'Machined')

    app.Ksurf= 4.51 * app.S_ut.Value ^ (-0.265);

    app.SurfaceFactorEditField.Value= app.Ksurf;

elseif strcmpi( app.surface1,'Hot Rolled')

    app.Ksurf= 57.7 * app.S_ut.Value ^ (-0.718);
    app.SurfaceFactorEditField.Value= app.Ksurf;
elseif strcmpi( app.surface1,'Ground')
    app.Ksurf= 1.58 * app.S_ut.Value ^ (-0.085);
    app.SurfaceFactorEditField.Value= app.Ksurf;
elseif strcmpi( app.surface1,'Forged')
     app.Ksurf= 272.0 * app.S_ut.Value ^ (-0.995);
     app.SurfaceFactorEditField.Value= app.Ksurf;
end
app.SurfaceFactorEditField.Value= app.Ksurf;
```

Where;

app.csurfinp.Value: Selected material finish.

app.S_ut.Value: Ultimate tensile strength of the material.

The code evaluates surface factor after the selection of material finish and shows on the input screen.

4.10.2. Loading Factor

Loading factor is taken as one for bending.

4.10.3. Reliability Factor Function

As same material is tested under the same conditions there occurred a scatter in the values by 8%. [8] Reliability factor is calculated from Eq. 4.6. [2]

$$k_e = 1 - 0.08z_a \qquad \text{(Eq. 4.6)}$$

```
% Calculating Reliability Factors
function [ kr ] = Kreliab(app, a )
if (a < 50)
   kr = 1;
elseif (a > 99.9999)
```

```
    kr = 0.620;
else
    y = [50 90 95 99 99.9 99.99 99.999 99.9999]';
    z = [1.000 0.897 0.868 0.814 0.753 0.702 0.659 0.620]';
    kr = interp1(y, z, a);
end
end
```

4.10.4. Calculating Endurance Limit Modifying Factors without Size Factor

The code is;

```
% Correction Factors without Size Factor
Kload = 1;  % for Bending
app.Kpart = Kload * app.Ksurf * Kreliab(app, app.Reliability.Value);
```

Where;

app.Reliability.Value: Reliability value.

4.11. Calculating Bending Stresses at Locations of Interest

Forces on gears and pulleys create bending stresses. Since the preliminary diameters used may change, at this point a while loop is also started.

The code is;

```
%  Loop for Dimensions Iteration
while (app.Iteration == 0)
%  Bending Stresses at locations of Interest
%  At Bearing 1
app.Inertia = 0;
app.Inertia = pi * app.DiaAtBearings.Value ^ 4 / 64;
app.StressBearing1 = app.MomentBear1 * (app.DiaAtBearings.Value / 2) / app.Inertia;
%  At Bearing 2
app.Inertia = 0;
app.Inertia = pi * app.DiaAtBearings.Value ^ 4 / 64;
app.StressBearing2 = app.MomentBear2 * (app.DiaAtBearings.Value / 2) / app.Inertia;
app.StressAltBearing2 = app.StressBearing2;
%  At Gear 1
app.Inertia = 0;
app.Inertia = pi * app.DiaAtGear1.Value ^ 4 / 64;
if(app.Inertia==0)
   app.StressGear1 =0;
else
   app.StressGear1 = app.MomentGear1 * (app.DiaAtGear1.Value / 2) / app.Inertia;
end
```

Where;

app.DiaAtBearings.Value: Preliminary diameter at bearings.

If shaft element is not exists on the location, inertia is calculated as zero not to stuck the app.

4.12. Calculating Shear Stresses at Locations of Interest

Torque on the shaft creates shear stresses. Below code is calculated shear forces at locations of interest.

The code is;

```
%  Shear Stresses at locations of Interest due to Torque
%  At Bearing 1 (Assuming No Torque)
app.ShearBearing1 = 0.0;
%  At Bearing 2
app.PInertia = 0;
app.PInertia = pi * app.DiaAtBearings.Value ^ 4 / 32;
app.ShearBearing2 = app.Torque * (app.DiaAtBearings.Value / 2) / app.PInertia;
%  At Gear 1
app.PInertia = 0;
app.PInertia = pi * app.DiaAtGear1.Value ^ 4 / 32;
if(app.PInertia==0)
   app.ShearGear1 = 0;
else
   app.ShearGear1 = app.Torque * (app.DiaAtGear1.Value / 2) / app.PInertia;
end
```

Where;

app.DiaAtBearings.Value: Preliminary diameter at bearings.

app.DiaAtGear1.Value: Preliminary diameter at gear 1.

If shaft element is not exists on the location, polar inertia is calculated as zero not to stuck the app.

4.13. Calculating the Size Factor Function

The size factor used is for bending and torsion. [2]

The code is;

```
% Size correction factor function for circular rotating disc
function [ ks ] = Ksize (app, Dia )
if (Dia < 2.79)
    ks = 1.0;
elseif (Dia < 51)
    ks = 1.24 * Dia^(-0.107);
else
    ks = 1.51 * Dia ^ (-0.157);
```

end

end

4.14. Calculating Fully Corrected Endurance Limit

Below is the code to calculate fully corrected endurance stresses at locations of interest.

The code is;

% Endurance Stresses at Locations of Interest

app.EnduranceAtBearings = 0.5 * app.S_ut.Value * app.Kpart * Ksize(app, app.DiaAtBearings.Value);

app.EnduranceAtGear1 = 0.5 * app.S_ut.Value * app.Kpart * Ksize(app, app.DiaAtGear1.Value);

app.EnduranceAtGear2 = 0.5 * app.S_ut.Value * app.Kpart * Ksize(app, app.DiaAtGear2.Value);

app.EnduranceAtSheave1 = 0.5 * app.S_ut.Value * app.Kpart * Ksize(app, app.DiaAtSheave1.Value);

app.EnduranceAtSheave2 = 0.5 * app.S_ut.Value * app.Kpart * Ksize(app, app.DiaAtSheave2.Value);

4.15. Function Calculating Static Stress Concentration Factors for Bending

Geometric stress-concentration factor Kt for a shaft with a shoulder fillet in bending is calculated from Eq. 4.7. [8]

$$K_t = A \times \left(\frac{r}{d}\right)^b \qquad \text{(Eq. 4.7)}$$

The code is;

% Stress concentration factor computing function. For bending, round disc with change in diameter.

function [kc] = Kscb(app, Diabig, Diasmall, Fillet)

%Diabig : Larger diameter

```
%Diasmall : Smaller diameter

%Fillet : Fillet on diameter change

D_table = [6.0 3.0 2.0 1.5 1.2 1.1 1.07 1.05 1.03 1.02 1.01]';

A_table = [.87868 .89334 .90879 .93836 .97098 .95120 .97527 .98137 .98061 .96048
.91938]';

b_table = [-.33243 -.30860 -.28598 -.25759 -.21796 -.23757 -.20958 -.19653 -.18381 -.17711
-.17032]';

% Interpolating A and b from the table

D_ratio = Diabig / Diasmall;

A = interp1 (D_table, A_table, D_ratio);

b = interp1 (D_table, b_table, D_ratio);

%  Stress concentration factor

kc = A * (Fillet /Diasmall) ^ b;

end
```

4.16. Function Calculating Static Stress Concentration Factors for Torsion

Geometric stress-concentration factor Kt for a shaft with a shoulder fillet in torsion is calculated from Eq. 4.7. [8]

The code is;

```
% Stress concentration factor computing function. For torsion, round disc with change in
diameter.

function [ kc ] = Ksct(app, Diabig, Diasmall, Fillet )

%Diabig : Larger diameter

%Diasmall : Smaller diameter

%Fillet : Fillet on diameter change

D_table = [2.0 1.33 1.20 1.09]';

A_table = [.86331 .84897 .83425 .90337]';
```

```
b_table = [-.23865 -.23161 -.21649 -.12692]';
% Interpolating A and b from the table
D_ratio = Diabig / Diasmall;
if (D_ratio > D_table(1))
    D_ratio = D_table(1);
end
if (D_ratio < D_table(4))
    D_ratio = D_table(4);
end
A = interp1 (D_table, A_table, D_ratio);
b = interp1 (D_table, b_table, D_ratio);
%  Stress concentration factor
kc = A * (Fillet /Diasmall) ^ b;
end
```

4.17. Calculating Static Stress Concentration Factors for Bending and Torsion

The code is;

```
% Static Stress Concentration Factors at Locations of Interest Computing for Bending and Shear
app.Kt_AtBearing1 = Kscb(app, (app.DiaAtBearings.Value+app.Fillet.Value*2), app.DiaAtBearings.Value, app.Fillet.Value);
app.Kts_AtBearing1 = Ksct(app, (app.DiaAtBearings.Value+app.Fillet.Value*2), app.DiaAtBearings.Value, app.Fillet.Value);
app.Kt_AtBearing2 = Kscb(app, (app.DiaAtBearings.Value+app.Fillet.Value*2), app.DiaAtBearings.Value, app.Fillet.Value);
app.Kts_AtBearing2 = Ksct(app, (app.DiaAtBearings.Value+app.Fillet.Value*2), app.DiaAtBearings.Value, app.Fillet.Value);
app.Kt_AtGear1 = Kscb(app, (app.DiaAtGear1.Value + app.Fillet.Value*2), app.DiaAtGear1.Value, app.Fillet.Value);
app.Kts_AtGear1 = Ksct(app, (app.DiaAtGear1.Value + app.Fillet.Value*2), app.DiaAtGear1.Value, app.Fillet.Value);
if(app.DiaAtGear2.Value==0)
   app.Kt_AtGear2=0;
else
   app.Kt_AtGear2 = Kscb(app, (app.DiaAtGear2.Value + app.Fillet.Value*2), app.DiaAtGear2.Value, app.Fillet.Value);


end
if(app.DiaAtGear2.Value==0)
   app.Kts_AtGear2=0;
else
   app.Kts_AtGear2 = Ksct(app, (app.DiaAtGear2.Value + app.Fillet.Value*2), app.DiaAtGear2.Value, app.Fillet.Value);


end
app.Kt_AtSheave1 = Kscb(app, (app.DiaAtSheave1.Value + app.Fillet.Value*2), app.DiaAtSheave1.Value, app.Fillet.Value);
app.Kts_AtSheave1 = Ksct(app, (app.DiaAtSheave1.Value + app.Fillet.Value*2), app.DiaAtSheave1.Value, app.Fillet.Value);
if(app.DiaAtSheave2.Value==0)
   app.Kt_AtSheave2 = 0;
else
   app.Kt_AtSheave2 = Kscb(app, (app.DiaAtSheave2.Value + app.Fillet.Value*2), app.DiaAtSheave2.Value, app.Fillet.Value);
end
```

```
if(app.DiaAtSheave2==0)
   app.Kts_AtSheave2 = 0;
else
   app.Kts_AtSheave2 = Ksct(app, (app.DiaAtSheave2.Value + app.Fillet.Value*2), app.DiaAtSheave2.Value, app.Fillet.Value);
end
```

4.18. Function for Reducing Stress Concentration Factor for Bending

Below function reduces the stress concentration factor using the Neuber equation. [2]

The code is;

```
function [ Kf ] = Kfsr(app, Kc, Fillet, S_ut )
Sc = S_ut / 6.8948; % in Kpsi
r = Fillet / 25.4;  % in inch
%  Neubers Constant for Bending
a = 0.246 - 3.08e-3*Sc + 1.51e-5*Sc^2 - 2.67e-8*Sc^3;
Kf = 1 + (Kc - 1) / (1 + a / sqrt(r));
end
```

4.19. Function for Reducing Stress Concentration Factor for Torsion

Below function reduces the stress concentration factor using the Neuber equation. [2]

The code is;

```
function [ Kf ] = Kfsrt(app, Kc, Fillet, S_ut)
Sc = S_ut / 6.8948; % in Kpsi
r = Fillet / 25.4;  % in inch
%  Neubers Constant for Torsion
a = 0.190 - 2.51e-3*Sc + 1.35e-5*Sc^2 - 2.67e-8*Sc^3;
Kf = 1 + (Kc - 1) / (1 + a / sqrt(r));
```

end

4.20. Reducing Static Stress Concentration Factors for Bending

The code is;

```
% Reducing Stress Concentration Factors for Fatigue at Locations of Interest
app.Kf_AtBearing1 = Kfsr(app, app.Kt_AtBearing1, app.Fillet.Value, app.S_ut.Value);
app.Kfs_AtBearing1 = Kfsrt(app, app.Kts_AtBearing1, app.Fillet.Value, app.S_ut.Value);
app.Kf_AtBearing2 = Kfsr(app, app.Kt_AtBearing2, app.Fillet.Value, app.S_ut.Value);
app.Kfs_AtBearing2 = Kfsrt(app, app.Kts_AtBearing2, app.Fillet.Value, app.S_ut.Value);
app.Kf_AtGear1 = Kfsr(app, app.Kt_AtGear1, app.Fillet.Value, app.S_ut.Value);
app.Kfs_AtGear1 = Kfsrt(app, app.Kts_AtGear1, app.Fillet.Value, app.S_ut.Value);
app.Kf_AtGear2 = Kfsr(app, app.Kt_AtGear2, app.Fillet.Value, app.S_ut.Value);
app.Kfs_AtGear2 = Kfsrt(app, app.Kts_AtGear2, app.Fillet.Value, app.S_ut.Value);
app.Kf_AtSheave1 = Kfsr(app, app.Kt_AtSheave1, app.Fillet.Value, app.S_ut.Value);
app.Kfs_AtSheave1 = Kfsrt(app, app.Kts_AtSheave1, app.Fillet.Value, app.S_ut.Value);
app.Kf_AtSheave2 = Kfsr(app, app.Kt_AtSheave2, app.Fillet.Value, app.S_ut.Value);
app.Kfs_AtSheave2 = Kfsrt(app, app.Kts_AtSheave2, app.Fillet.Value, app.S_ut.Value);
```

4.21. Function Calculating Von Mises Alternating Stresses

For bending, torsional and axial stresses, von Mises alternating stress is found from Eq. 4.8. [2]

$$\sigma_a' = \left\{\left[\left(K_f\right)_{bending}(\sigma_a)_{bending} + \left(K_f\right)_{axial}\frac{(\sigma_a)_{axial}}{0.85}\right]^2 + 3\left[\left(K_{fs}\right)_{torsion}(\tau_a)_{torsion}\right]^2\right\}^{\frac{1}{2}} \qquad \text{(Eq. 4.8)}$$

The code is;

```
% Calculating Von Mises Alternating Stresses
function [ Salt_VonMisses ] = VonMisses_Alt(app, Kf_BND, Sa_BND, Kf_Ax, Sa_Ax,
Kfs_Tor, T_Alt)

Salt_VonMisses = sqrt((Kf_BND*Sa_BND + Kf_Ax * Sa_Ax/0.85)^2 + 3*(Kfs_Tor*T_Alt)^2);

end
```

4.22. Function Calculating Von Mises Mean Stresses

For bending, torsional and axial stresses, von Mises mean stress is found from Eq. 4.9. [2]

$$\sigma'_m = \left\{\left[\left(K_f\right)_{bending}(\sigma_m)_{bending} + \left(K_f\right)_{axial}(\sigma_m)_{axial}\right]^2 + 3\left[\left(K_{fs}\right)_{torsion}(\tau_m)_{torsion}\right]^2\right\}^{\frac{1}{2}} \qquad \text{(Eq. 4.9)}$$

The code is;

```
% Calculating Von Mises Mean Stresses

function [ Smean_VonMisses ] = VonMisses_Mean(app, Kf_BND, Sm_BND, Kf_Ax, Sm_Ax, Kfs_Tor, T_Mean)

Smean_VonMisses = sqrt((Kf_BND*Sm_BND + Kf_Ax*Sm_Ax)^2 + 3*(Kfs_Tor*T_Mean)^2);

end
```

4.23. Calculating Von Mises Stresses

The code is;

```
% Checking If Torsion Exists on Bearing2

if(app.TCheckBearing2.Value ==1)

   app.ShearAltBearing2 = app.ShearBearing2;

   app.Kfs_TorsionBearing2 = app.Kfs_AtBearing2;

else

   app.ShearAltBearing2 = 0;

   app.Kfs_TorsionBearing2 = 1;

end
```

```
% Von Mises Stresses at Locations of Interest
app.VonMisses_Alt_Bearing1 = VonMisses_Alt(app, app.Kf_AtBearing1,
app.StressBearing1, 1.0, 0.0, 1.0, 0);
app.VonMisses_Mean_Bearing1 = VonMisses_Mean(app, app.Kf_AtBearing1, 0, 1.0, 0.0,
1.0, app.ShearBearing1);
app.VonMisses_Alt_Bearing2 = VonMisses_Alt(app, app.Kf_AtBearing2,
app.StressAltBearing2, 1.0, 0.0, 1.0, 0);
app.VonMisses_Mean_Bearing2 = VonMisses_Mean(app, app.Kf_AtBearing2, 0, 1.0, 0.0,
app.Kfs_TorsionBearing2, app.ShearBearing2);
app.VonMisses_Alt_Gear1 = VonMisses_Alt(app, app.Kf_AtGear1, app.StressGear1, 1.0,
0.0, 1.0, 0);
app.VonMisses_Mean_Gear1 = VonMisses_Mean(app, app.Kf_AtGear1, 0, 1.0, 0.0,
app.Kfs_AtGear1, app.ShearGear1);
app.VonMisses_Alt_Gear2 = VonMisses_Alt(app, app.Kf_AtGear2, app.StressGear2, 1.0,
0.0, 1.0, 0);
app.VonMisses_Mean_Gear2 = VonMisses_Mean(app, app.Kf_AtGear2, 0, 1.0, 0.0,
app.Kfs_AtGear2, app.ShearGear2);
app.VonMisses_Alt_Sheave1 = VonMisses_Alt(app, app.Kf_AtSheave1, app.StressSheave1,
1.0, 0.0, 1.0, 0);
app.VonMisses_Mean_Sheave1 = VonMisses_Mean(app, app.Kf_AtSheave1, 0, 1.0, 0.0,
app.Kfs_AtSheave1, app.ShearSheave1);
app.VonMisses_Alt_Sheave2 = VonMisses_Alt(app, app.Kf_AtSheave2, app.StressSheave2,
1.0, 0.0, 1.0, 0);
app.VonMisses_Mean_Sheave2 = VonMisses_Mean(app, app.Kf_AtSheave2, 0, 1.0, 0.0,
app.Kfs_AtSheave2, app.ShearSheave2);
```

4.24. Function for Calculating Langer's Safety Factor

Langer safety factor is calculated from Eq. 4.10. [2]

$$n = \frac{S_y}{\sigma_a + \sigma_m} \qquad \text{(Eq. 4.10)}$$

The code is;

```
% Static Safety Factor - Langer
function [ N ] = SSF_Langer(app, S_Alt, S_Mean, S_y)
N = S_y / (S_Mean + S_Alt);
end
```

4.25. Function for Calculating Modified Goodman Safety Factor

Modified Goodman safety factor is calculated from Eq. 4.11. [2]

$$n = \frac{1}{\left(\frac{\sigma_a}{S_e} + \frac{\sigma_m}{S_{ut}}\right)} \qquad \text{(Eq. 4.11)}$$

The code is;

```
% Safety Factor - Modified Goodman
function [ N ] = M_Goodman(app, S_Alt, S_Mean, S_E, S_Ut)
N = 1 / (S_Alt / S_E + S_Mean / S_Ut);
End
```

4.26. Function for Selecting Next Larger Size

Below function is to select next larger size of diameter on location of interest if Langer safety factor or modified Goodman safety factor is less than allowable user safety factor.

The code is;

```
app.AllowableShaftSize = [10, 12, 14, 15, 16, 17, 18, 20, 22, 25, 27, 30, 31, 32, 33, 34, 35,
36, 37, 38, 40, 42, 45, 48, 50, 52, 55, 58, 60, 62, 65, 68, 70, 72, 75, 78, 80, 82, 85, 88, 90, 92,
95, 98, 100];
% Loop for the Available Larger Size
function [Diameter] = New_Size(app, Previous_Diameter, Allowable_Diameter)
```

```
[a, ArrayLength] = size(Allowable_Diameter);
Done = 0;
for i = 1: ArrayLength
   Diameter = Allowable_Diameter (i);
   if (Diameter > Previous_Diameter)
Done = 1;
break;
   end
end
end
```

4.27. Calculating Langer & Modified Goodman Safety Factor

The code is;

```
% At Bearing 1
app.Langer_N_Bearing1 = SSF_Langer(app, app.VonMisses_Alt_Bearing1, app.VonMisses_Mean_Bearing1, app.S_y.Value);
app.Goodman_N_Bearing1 = M_Goodman(app, app.VonMisses_Alt_Bearing1, app.VonMisses_Mean_Bearing1, app.EnduranceAtBearings, app.S_ut.Value);
app.SF_NBearing1 = min(app.Langer_N_Bearing1, app.Goodman_N_Bearing1);
if (app.SF_NBearing1 < app.Min_SF.Value)
   [app.DiaAtBearings.Value] = New_Size(app, app.DiaAtBearings.Value, app.AllowableShaftSize);
   continue;
end
% At Bearing 2
app.Langer_N_Bearing2 = SSF_Langer(app, app.VonMisses_Alt_Bearing2, app.VonMisses_Mean_Bearing2, app.S_y.Value);
app.Goodman_N_Bearing2 = M_Goodman(app, app.VonMisses_Alt_Bearing2,
app.VonMisses_Mean_Bearing2, app.EnduranceAtBearings, app.S_ut.Value);
app.SF_NBearing2 = min(app.Langer_N_Bearing2, app.Goodman_N_Bearing2);
if (app.SF_NBearing2 < app.Min_SF.Value)
   [app.DiaAtBearings.Value]        =        New_Size(app,        app.DiaAtBearings.Value,
app.AllowableShaftSize);
   continue;
end
%  At Gear 1
if (app.Gear1Dist.Value~=0)
   app.Langer_N_Gear1        =        SSF_Langer(app,        app.VonMisses_Alt_Gear1,
app.VonMisses_Mean_Gear1, app.S_y.Value);
   app.Goodman_N_Gear1        =        M_Goodman(app,        app.VonMisses_Alt_Gear1,
app.VonMisses_Mean_Gear1, app.EnduranceAtGear1, app.S_ut.Value);
   app.SF_NGear1 = min(app.Langer_N_Gear1, app.Goodman_N_Gear1);
   if (app.SF_NGear1 < app.Min_SF.Value)
[app.DiaAtGear1.Value] = New_Size(app, app.DiaAtGear1.Value, app.AllowableShaftSize);
      continue;
   end
end
%  At Gear 2
if(app.Gear2Dist.Value~=0)
   app.Langer_N_Gear2        =        SSF_Langer(app,        app.VonMisses_Alt_Gear2,
app.VonMisses_Mean_Gear2, app.S_y.Value);
   app.Goodman_N_Gear2        =        M_Goodman(app,        app.VonMisses_Alt_Gear2,
app.VonMisses_Mean_Gear2, app.EnduranceAtGear2, app.S_ut.Value);
   app.SF_NGear2 = min(app.Langer_N_Gear2, app.Goodman_N_Gear2);
   if (app.SF_NGear2 < app.Min_SF.Value)
```

```
        [app.DiaAtGear2.Value] = New_Size(app, app.DiaAtGear2.Value, app.AllowableShaftSize);
        continue;
    end
end
% At Sheave 1
if(app.Pulley1XDist.Value~=0)
    app.Langer_N_Sheave1 = SSF_Langer(app, app.VonMisses_Alt_Sheave1, app.VonMisses_Mean_Sheave1, app.S_y.Value);
    app.Goodman_N_Sheave1 = M_Goodman(app, app.VonMisses_Alt_Sheave1, app.VonMisses_Mean_Sheave1, app.EnduranceAtSheave1, app.S_ut.Value);
    app.SF_NSheave1 = min(app.Langer_N_Sheave1, app.Goodman_N_Sheave1);
    if (app.SF_NSheave1 < app.Min_SF.Value)
        [app.DiaAtSheave1.Value] = New_Size(app, app.DiaAtSheave1.Value, app.AllowableShaftSize);
        continue;
    end
end
% At Sheave 2
if(app.Pulley2XDist.Value~=0)
    app.Langer_N_Sheave2 = SSF_Langer(app, app.VonMisses_Alt_Sheave2, app.VonMisses_Mean_Sheave2, app.S_y.Value);
    app.Goodman_N_Sheave2 = M_Goodman(app, app.VonMisses_Alt_Sheave2, app.VonMisses_Mean_Sheave2, app.EnduranceAtSheave2, app.S_ut.Value);
    app.SF_NSheave2 = min(app.Langer_N_Sheave2, app.Goodman_N_Sheave2);
    if (app.SF_NSheave2 < app.Min_SF.Value)
        [app.DiaAtSheave2.Value] = New_Size(app, app.DiaAtSheave2.Value, app.AllowableShaftSize);
        continue;
    end
end
app.Iteration = 1;
```

4.28. Function for Calculating Gerber Safety Factor

The code is;

```
% Safety Factor – Gerber [2]
function [ N ] = SF_Gerber(app, S_Alt, S_Mean, S_E, S_Ut )
if (S_Mean == 0)
    N = S_E / S_Alt;
else
    G_A = sqrt(1 + (2*S_Mean*S_E/(S_Ut*S_Alt))^2) - 1;
    N = 0.5 * ((S_Ut / S_Mean)^2)* (S_Alt / S_E) * G_A;
end
end
```

4.29. Function for Calculating ASME Safety Factor

The code is;

```
% Safety Factor - ASME [2]
function [ N ] = SF_ASME(app, S_Alt, S_Mean, S_E, S_y )
Asme_A = ((S_Alt / S_E)^2) + ((S_Mean / S_y)^2);
N = sqrt( 1 / Asme_A);
End
```

4.30. Calculating Gerber Safety Factor

The code is;

```
% Iteration Control
app.Iteration = 1;
% Gerber Safety Factor
app.Gerber_N_Bearing1 = SF_Gerber(app, app.VonMisses_Alt_Bearing1,
app.VonMisses_Mean_Bearing1, app.EnduranceAtBearings, app.S_ut.Value);
app.Gerber_N_Bearing2 = SF_Gerber(app, app.VonMisses_Alt_Bearing2,
app.VonMisses_Mean_Bearing2, app.EnduranceAtBearings, app.S_ut.Value);
app.Gerber_N_Gear1 = SF_Gerber(app, app.VonMisses_Alt_Gear1,
app.VonMisses_Mean_Gear1, app.EnduranceAtGear1, app.S_ut.Value);
app.Gerber_N_Gear2 = SF_Gerber(app, app.VonMisses_Alt_Gear2,
app.VonMisses_Mean_Gear2, app.EnduranceAtGear2, app.S_ut.Value);
app.Gerber_N_Sheave1 = SF_Gerber(app, app.VonMisses_Alt_Sheave1,
app.VonMisses_Mean_Sheave1, app.EnduranceAtSheave1, app.S_ut.Value);
app.Gerber_N_Sheave2 = SF_Gerber(app, app.VonMisses_Alt_Sheave2,
app.VonMisses_Mean_Sheave2, app.EnduranceAtSheave2, app.S_ut.Value);
```

4.31. Calculating ASME Safety Factor

The code is;

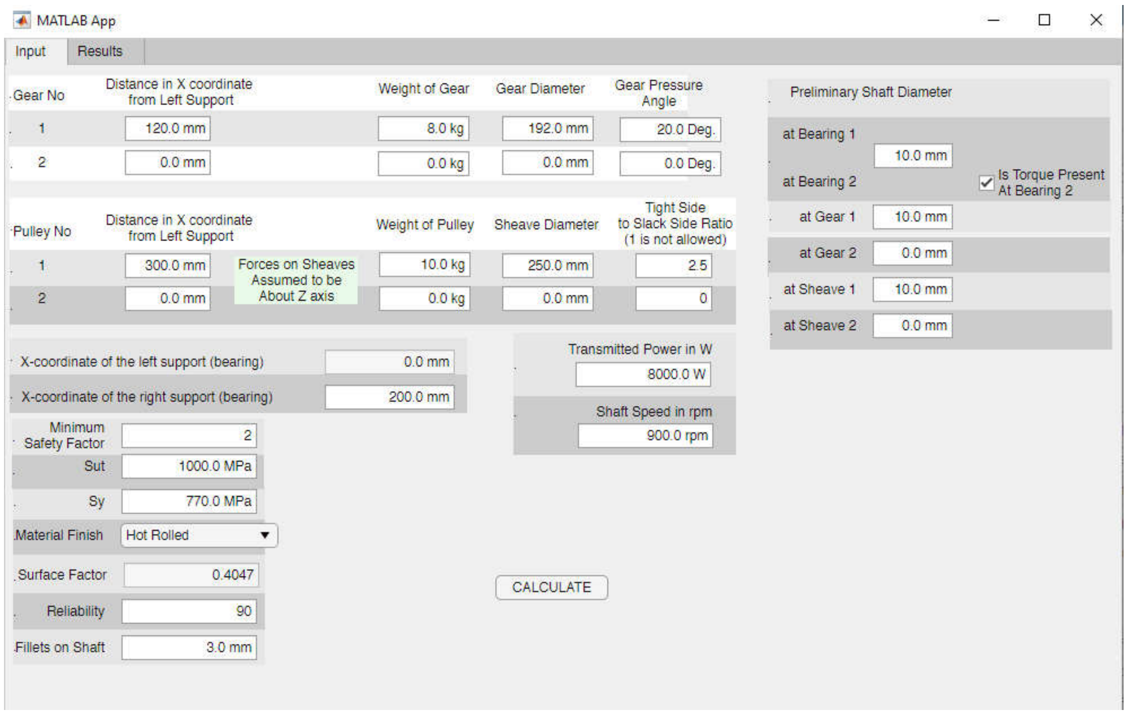
```
% ASME Safety Factor
app.Asme_N_Bearing1 = SF_ASME(app, app.VonMisses_Alt_Bearing1,
app.VonMisses_Mean_Bearing1, app.EnduranceAtBearings, app.S_ut.Value);
app.Asme_N_Bearing2 = SF_ASME(app, app.VonMisses_Alt_Bearing2,
app.VonMisses_Mean_Bearing2, app.EnduranceAtBearings, app.S_ut.Value);
app.Asme_N_Gear1 = SF_ASME(app, app.VonMisses_Alt_Gear1,
app.VonMisses_Mean_Gear1, app.EnduranceAtGear1, app.S_ut.Value);
app.Asme_N_Gear2 = SF_ASME(app, app.VonMisses_Alt_Gear2,
app.VonMisses_Mean_Gear2, app.EnduranceAtGear2, app.S_ut.Value);
app.Asme_N_Sheave1 = SF_ASME(app, app.VonMisses_Alt_Sheave1,
app.VonMisses_Mean_Sheave1, app.EnduranceAtSheave1, app.S_ut.Value);
app.Asme_N_Sheave2 = SF_ASME(app, app.VonMisses_Alt_Sheave2, app.VonMisses_Mean_Sheave2, app.EnduranceAtSheave2, app.S_ut.Value);

%  Ending While Loop

End
```

5. RESULTS AND DISCUSSION

Software is verified with a sample problem.

Problem: Determine a sensible minimum nominal diameter for the drive shaft consisting of a mid-mounted spur gear and overhung pulley wheel. The shaft is to be manufactured using 817M40 hot-rolled alloy steel with σ_{uts} = 1000MPa, σ_y = 770MPa and Brinell Hardness approximately 220 BHN. The radius of the fillets at the gear and pulley shoulders is 3mm. The power to be transmitted is 8 kW at 900rpm. The pitch circle diameter of the 20-degree pressure angle spur gear is 192mm and the pulley diameter is 250mm. The masses of the gear and pulley are 8kg and 10kg, respectively. The ratio of belt tensions should be taken as

2.5. Profiled keys are used to transmit torque through the gear and pulley. A shaft nominal reliability of 90% is desired. A nominal diameter of 32mm can be assumed for the shaft. Gear is 120mm apart from left bearing, right bearing is 200mm apart from left bearing, and pulley is 300mm apart from left bearing. [1]


MATLAB App

Input | Results

Gear No	Distance in X coordinate from Left Support	Weight of Gear	Gear Diameter	Gear Pressure Angle
1	120.0 mm	8.0 kg	192.0 mm	20.0 Deg.
2	0.0 mm	0.0 kg	0.0 mm	0.0 Deg.

Pulley No	Distance in X coordinate from Left Support		Weight of Pulley	Sheave Diameter	Tight Side to Slack Side Ratio (1 is not allowed)
1	300.0 mm	Forces on Sheaves Assumed to be About Z axis	10.0 kg	250.0 mm	2.5
2	0.0 mm		0.0 kg	0.0 mm	0

Preliminary Shaft Diameter	
at Bearing 1	10.0 mm
at Bearing 2	✓ Is Torque Present At Bearing 2
at Gear 1	10.0 mm
at Gear 2	0.0 mm
at Sheave 1	10.0 mm
at Sheave 2	0.0 mm

X-coordinate of the left support (bearing): 0.0 mm

X-coordinate of the right support (bearing): 200.0 mm

Transmitted Power in W: 8000.0 W

Shaft Speed in rpm: 900.0 rpm

Minimum Safety Factor: 2

Sut: 1000.0 MPa

Sy: 770.0 MPa

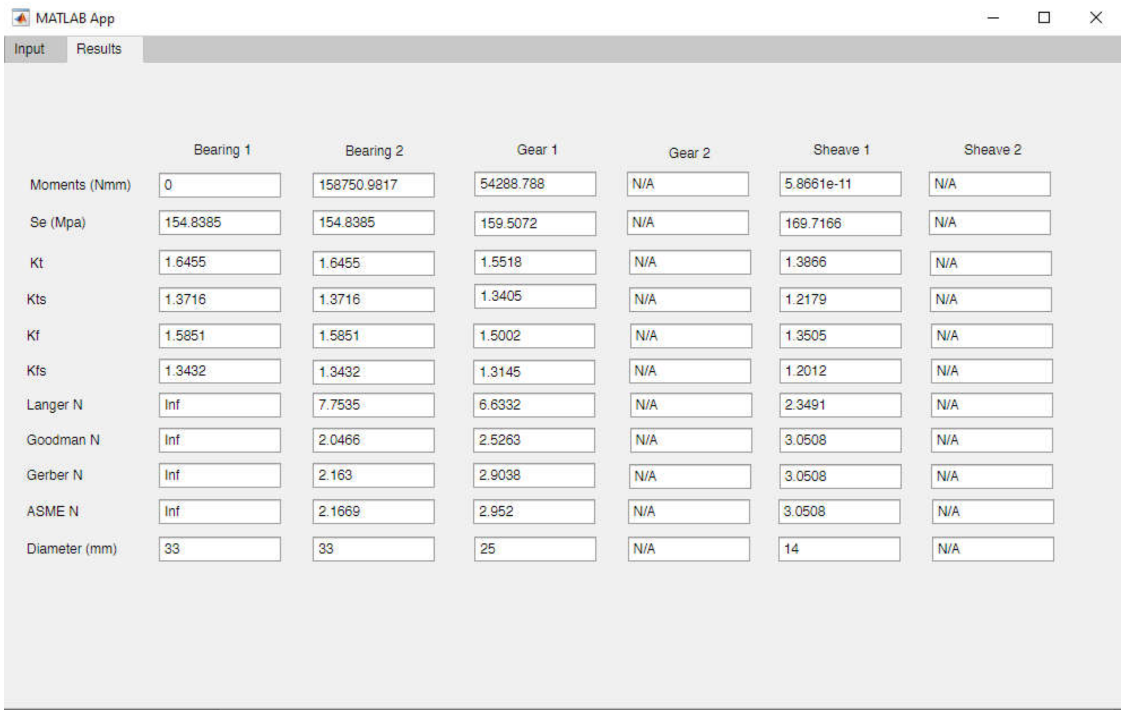
Material Finish: Hot Rolled

Surface Factor: 0.4047

Reliability: 90

Fillets on Shaft: 3.0 mm

Figure 5.1: Proposed software inputs of the problem


MATLAB App

Input | Results

	Bearing 1	Bearing 2	Gear 1	Gear 2	Sheave 1	Sheave 2
Moments (Nmm)	0	158750.9817	54288.788	N/A	5.8661e-11	N/A
Se (Mpa)	154.8385	154.8385	159.5072	N/A	169.7166	N/A
Kt	1.6455	1.6455	1.5518	N/A	1.3866	N/A
Kts	1.3716	1.3716	1.3405	N/A	1.2179	N/A
Kf	1.5851	1.5851	1.5002	N/A	1.3505	N/A
Kfs	1.3432	1.3432	1.3145	N/A	1.2012	N/A
Langer N	Inf	7.7535	6.6332	N/A	2.3491	N/A
Goodman N	Inf	2.0466	2.5263	N/A	3.0508	N/A
Gerber N	Inf	2.163	2.9038	N/A	3.0508	N/A
ASME N	Inf	2.1669	2.952	N/A	3.0508	N/A
Diameter (mm)	33	33	25	N/A	14	N/A

Figure 5.2: The output interface of the proposed software for shaft design

Table 1: A comparison between the proposed software and book results

	Moment Bearing 2 (Nm)	Moment Gear 1 (Nm)	Se Bearing 2 (MPa)	Diameter Bearing 2 (mm)
Proposed Software	158.75	54.29	159.51	33
Book Results	158.8	54.27	155.5	33

Figure 3 illustrates input parameters entered in the proposed software. After calculation button pressed result page can be seen in Figure 4. We can see book results and proposed software results in Table 1. Main difference occurred in modified endurance limit value. However, the proposed diameter for bearing 2 is same.

6. CONCLUSION

In this project, we focused on an app that will aid the user for the mechanical design of a shaft. Matlab app designer used as programing software that has a user friendly GUI and ability to share as standalone app. Proposed app analyzes the shaft based on dynamic loads. Optimal shaft diameter at the locations of interest are calculated by fatigue stress analysis. Langer, and modified Goodman safety factors are used to check next available diameter then Gerber and ASME safety factors are computed. Verification is also done with a problem from Mechanical Design Engineering Handbook. Result for proposed diameters matches in both solutions.

The author is aware of the limitations of the app. Only shoulders are used for notch type. Shaft elements are limited to two bearings, two gears and two pulleys. However, the core of the design is solid and can be used for a most sophisticated design of a shaft.

REFERENCES

[1] Childs, Peter R. N., (2019) Mechanical Design Engineering Handbook, 2nd edition, Butterworth Heinemann, Oxford, United Kingdom.

[2] Budynas, R. G. and Nisbett, J. K., (2011) Shigley's Mechanical Engineering Design, 9th edition, McGraw-Hill Companies, New York.

[3] Adekunle, A.A, Adejuyigbe, S.B., Arulogun O.T., (2012) Development of CAD Software for Shaft Under Various Loading Conditions, Procedia Engineering, 38, pp. 1962-1983.

[4] Saradava, Ketan D, Mandaliya, Piyush J., Parsania, Pratik P., (2016) Design of Machine Shaft in Fatigue Loading by Using C++ Programming Language, Trends in Machine Design, Volume 3, Issue 1, pp. 75-81.

[5] Fadare, D. A., Akanbi, O., (2010) Development Of A Computer Aided Software For Power Transmission Shaft Design With Multiple Criteria, Ife Journal of Technology, 19(1), pp. 25-33.

[6] MITCalc, Computer-Aided Shaft Design Software, https://www.mitcalc.com/doc/shafts/help/en/shaft.htm, January 16, 2022.

[7] What Is MATLAB? 2015: https://ch.mathworks.com/discovery/what-ismatlab.html

[8] Robert L. Norton, (2010) Machine Design An Integrated Approach, 4th edition, Prentice Hall, New Jersey.