close all

## **Reading Excel Files**

fileName = 'boltSpecsAndMaterialProperties.xlsx';  
  
SAEBoltStrength = readtable(fileName, 'PreserveVariableNames', true);  
  
aluminumAlloys = readtable(fileName,'SHEET','ALUMINUM ALLOYS', ...  
 'PreserveVariableNames', true);  
  
boltDimensions = readtable(fileName, 'SHEET', 'BOLT DIMENSIONS', ...  
 'PreserveVariableNames', true);  
  
% Must be presented in increasing inner diameter order  
bearingDimensions = readtable(fileName, 'SHEET', 'BEARING DIMENSIONS', ...  
 'PreserveVariableNames', true);

## **Defining Design Factor and Crash Factor**

% nd  
designFactor = 2.5;  
% Kcrash  
crashFactor = 3;

## **Determining Bolt Size**

% Defining dBolt as a symbolic variable  
syms DBolt  
  
% Minimum yield strenght (kpsi) of bolts  
minYieldStrengthBolt = table2array(SAEBoltStrength(1:end, 5))';  
  
% Finding vector sum of reaction forces on Lateral Support  
pLug = sqrt(rxns(3,1)^2 + rxns(4,1)^2);  
  
Theta = atand(rxns(4,1) ./ rxns(3,1));  
  
% Standard bolt sizes  
stdBoltDia = table2array(boltDimensions(1:end, 2))';  
  
% Initializing all size changing vectors  
vecLength = length(minYieldStrengthBolt);  
Sy = zeros(1, vecLength);  
dBolt = zeros(1, vecLength);  
dBoltStd = zeros(1, vecLength);  
  
for ii = 1:length(minYieldStrengthBolt)  
  
 % Yield strength of bolt  
 Sy(ii) = minYieldStrengthBolt(ii) \* 1000; % TEMP VAL (READ FROM EXCEL)  
  
 % Shear stress on lugbolt  
 tauBolt = pLug \* crashFactor ./ (pi \* (DBolt.^2) ./ 4);  
  
 % Design factor equation w/ shear stress subbed in  
 materialSelection = Sy(ii) \* .5 ./ tauBolt == designFactor;  
  
 % Solving for Dbolt  
 boltDiaIdx = solve(materialSelection, DBolt);  
  
 % Taking positive value and storing in dBolt array  
 dBolt(ii) = boltDiaIdx(1);  
  
 % Determining standard bolt size  
 for jj = 1: length(dBolt)  
  
 potentialDia = stdBoltDia(stdBoltDia > dBolt(jj));  
  
 dBoltStd(jj) = min(potentialDia);  
 end  
  
end  
  
% Identifying unique std bolt sizes  
uniqueStdSizes = unique(dBoltStd);  
Undefined function 'rxns' for input arguments of type 'double'.  
  
Error in lugDesign (line 33)  
pLug = sqrt(rxns(3,1)^2 + rxns(4,1)^2);

## **Determining Bearing Dimensions**

% Extracting Bearing Inner Diameter (in)  
bearingIDs = table2array(bearingDimensions(1:end, 2))';  
  
% Extracting bearing outer diameter (in)  
bearingODs = table2array(bearingDimensions(1:end, 3))';  
  
% Extracting bearing width (in)  
bearingWidths = table2array(bearingDimensions(1:end, 4))';  
  
% Initializing all size changing vectors  
bearingID = zeros(1, length(dBoltStd));  
bearingWidth = zeros(1, length(dBoltStd));  
bearingOD = zeros(1, length(dBoltStd));  
  
for ii = 1:length(dBoltStd)  
  
 % Inner diameter of bearing  
 potentialBearings = bearingIDs(bearingIDs >= dBoltStd(ii));  
 bearingID(ii) = min(potentialBearings);  
  
 % Bearing width  
 negativeIndex = length(potentialBearings);  
 bearingWidth(ii) = bearingWidths(end - negativeIndex + 1);  
  
 % Outer diameter of bearinf  
 bearingOD(ii) = bearingODs(end - negativeIndex + 1);  
  
end

## **Determining Depth and Width of Clevis**

% ----- Geometry -----  
% Thickness of bushing diameter  
bushingThickness = .03;  
  
% Bushing flange thickness  
bushingFlange = .031;  
  
% Allowable gap between face of the bushing flanges and bearing  
linkTol = .01;  
  
  
% Initializing all size changing vectors  
Dlug = zeros(1, length(dBoltStd));  
Rlink = zeros(1, length(dBoltStd));  
minClevisDepth = zeros(1, length(dBoltStd));  
widthClevis = zeros(1, length(dBoltStd));  
  
for ii = 1:length(bearingID)  
  
 % Lug Diameter  
 Dlug(ii) = dBoltStd(ii) + (2 \* bushingThickness);  
  
 % Radius of link  
 Rlink(ii) = .8 \* bearingOD(ii);  
  
 % Defining minumum clevis depth  
 minClevisDepth(ii) = Rlink(ii);  
  
 % Defining clevis width  
 widthClevis(ii) = (bushingFlange \* 2 ) + bearingWidth(ii) + ...  
 (2 \*linkTol);  
  
end

## **Defining Lug Geometry**

Wlug = 1.25; % DEPENEDS ON DESIGN  
Rlug = 2; % DEPENEDS ON DESIGN  
% Theta = 0; % DEPENEDS ON DESIGN

## **Design Curves**

% ----- R/D Design Curves ------  
% R/D Ratio  
RoDplot = linspace(.4, 3, 1000);  
RoD = Rlug ./ Dlug;  
  
% R/D design curve equation  
rd = @(RoDplot) (8\*RoDplot-4) ./ (3+2\*RoDplot);  
  
% ----- W/D Design Curves -----  
% W/D ratio  
WoDplot = linspace(.4 ,3 ,1000);  
WoD = Wlug ./ Dlug;  
  
  
% W/D Equation 1  
wd1 = @(WoDplot) -1.1241624334181 + 1.26260602679792\*(WoDplot) - ...  
 0.134109402804349\*(WoDplot).^2;  
  
% W/D Equation 2  
wd2 = @(WoDplot) -1.05128778218923 + 1.11850813948953\*(WoDplot) - ...  
 0.064994276922263\*(WoDplot).^2;  
  
% W/D Equation 3  
wd3 = @(WoDplot) -1.04780311931278 + 1.09937838959058\*(WoDplot) - ...  
 0.050356226727686\*(WoDplot).^2;  
  
% W/D Equation 4  
wd4 = @(WoDplot) -1.05695777389161 + 1.09458793219562\*(WoDplot) - ...  
 0.039868567021969\*(WoDplot).^2;  
  
% W/D Equation 5  
wd5 = @(WoDplot) -1.04007802184702 + 1.06368514278838\*(WoDplot) ...  
 - 0.024983837942521\*(WoDplot).^2;  
  
% ----- Offset Loading Design Curve -----  
% Offset Loading Curve  
offsetLoading = @(Theta) 0.9999125874117 - ...  
 0.000129506604504392\*(Theta)- 0.000091754079254061.\*(Theta).^2 + ...  
 6.94250194250276E-07.\*(Theta).^3;  
  
% Creating new figure  
figure(1)  
  
% Allowing for multiple plots  
hold on  
% Turning grid on  
grid on  
% Defining grid size  
grid minor  
  
% PLotting all W/D Curves  
plot(wd1(linspace(0,3, 1000)), linspace(0, 3, 1000));  
plot(wd2(linspace(0,3, 1000)), linspace(0, 3, 1000));  
plot(wd3(linspace(0,3, 1000)), linspace(0, 3, 1000));  
plot(wd4(linspace(0,3, 1000)), linspace(0, 3, 1000));  
plot(wd5(linspace(0,3, 1000)), linspace(0, 3, 1000));  
  
% PLotting R/D curve  
plot(rd(linspace(0, 3, 1000)), linspace(0, 3, 1000));  
  
% Defining axis limits  
xlim([0 2.2]);  
ylim([.4 3.0]);  
  
% Plot Descriptors  
xlabel('\emph {$F\_{Br}$ / $F\_{Tu}$}', 'fontsize', 14, 'Interpreter', ...  
 'latex');  
ylabel('\emph {$\frac{R}{D}$ $\&$ $\frac{W}{D}$}', 'fontsize', 14, ...  
 'Interpreter', 'latex');  
title('\emph {Lug Design Chart}', 'fontsize', 16, 'Interpreter', 'latex');  
legend('W/D - 1','W/D - 2','W/D - 3','W/D - 4','W/D - 5','R/D', ...  
 'location','northwest');  
  
% PLOT OFFSTE LOADING CURVE  
% Creating new figure  
figure(2)  
  
% X-Vals  
ThetaPlot = linspace(0, 90, 1000);  
% Y-Vals  
plot(ThetaPlot, offsetLoading(ThetaPlot));  
  
%Plot adjustemnts  
grid on  
grid minor  
  
% Plot Descriptors  
ylabel('\emph {$P\_{0}$ / $P\_{\theta}$}', 'fontsize', 14, 'Interpreter', ...  
 'latex');  
xlabel('\emph {$\theta$ (Degrees)}', 'fontsize', 14, ...  
 'Interpreter', 'latex');  
title('\emph {Allowable Lateral Lug Loads}', 'fontsize', 16, ...  
 'Interpreter', 'latex');  
legend('Off-Axis Corrective Factor','location','northeast');

## **Bearing Strength**

% Extracting ultimate strenghts from aluminum alloys table (kpsi)  
ultimateStrength = table2array(aluminumAlloys(1:end, 3))';  
  
% Ultimate strenght of aluminum alloy for clevis (psi)  
Sut = ultimateStrength \* 1000;  
  
% Defining Sbr as symbolic variable  
syms Sbr  
  
bearingStrength = zeros(length(ultimateStrength), length(dBoltStd));  
  
for ii = 1:length(ultimateStrength)  
 for jj = 1: length(dBoltStd)  
  
 % Bearing strength equaiton  
 bearingStrenghtEq = Sbr == Sut(ii) \* wd5(WoD(jj)) \* ...  
 offsetLoading(Theta);  
  
 % Solving for bearing strength  
 bearingStrength(ii, jj) = double(solve(bearingStrenghtEq, Sbr));  
  
 end  
end  
  
  
% INCLUDE IF STATEMENT TO PICK R/D vs. W/D

## **Calculating Thickness of Lug**

% Defini tLug as symbolic variable  
syms tLug  
  
sigmaLug = zeros(length(ultimateStrength), length(dBoltStd));  
lugThickness = zeros(length(ultimateStrength), length(dBoltStd));  
  
for ii = 1:length(ultimateStrength)  
 for jj = 1:length(dBoltStd)  
  
 % Lug stress equation  
 sigmaLug(ii, jj) = bearingStrength(ii, jj) ./ designFactor;  
  
 % Lug thickness equation  
 lugThicknessEq = sigmaLug(ii, jj) == (.6 \* pLug \* crashFactor) ...  
 ./ (Dlug(jj) \* tLug);  
  
 % Solving for lug thickness  
 lugThickness(ii, jj) = double(solve(lugThicknessEq, tLug));  
  
 end  
end