
```

function springCalculatorGUI
    % Create the main figure
    fig = uifigure('Name', 'Spring Calculator', 'Position', [400, 360, 400,
400]); % left, bottom, width, height

    % Create UI components

    % END TYPE
    endTypeLabel = uilabel(fig, 'Text', 'End Type:', 'Position', [20, 330,
80, 22]);
    endTypeDropDown = uidropdown(fig, 'Items', {'', 'Plain', 'Plain and
ground', 'Squared or closed', 'Squared and ground' }, 'Position', [120, 330,
120, 22]);

    % MATERIAL
    materialLabel = uilabel(fig, 'Text', 'Material:', 'Position', [20, 300,
80, 22]);
    materialDropDown = uidropdown(fig, 'Items', {'', 'Music wire A228', 'Hard-
drawn wire A227', 'Chrome-vanadium wire A232', 'Chrome-silicon wire A401',
'302 stainless wire A313', 'Phosphor-bronze wire B159'}, 'Position', [120,
300, 200, 22]);

    % Units Dropdown
    unitsLabel = uilabel(fig, 'Text', 'Units:', 'Position', [20, 270, 50,
22]);
    unitsDropDown = uidropdown(fig, 'Items', {'Metric', 'English'},
'Position', [80, 270, 80, 22]);

    % WIRE DIAMETER
    wireDiameterLabel = uilabel(fig, 'Text', 'Wire Diameter (mm/in):',
'Position', [20, 240, 120, 22]);
    wireDiameterEdit = uieditfield(fig, 'numeric', 'Position', [150, 240, 70,
22]);

    % OUTER DIAMETER
    outerDiameterLabel = uilabel(fig, 'Text', 'Outer Diameter (mm/in):',
'Position', [20, 210, 120, 22]);
    outerDiameterEdit = uieditfield(fig, 'numeric', 'Position', [150, 210,
70, 22]);

    % FREE LENGTH
    freeLengthLabel = uilabel(fig, 'Text', 'Free Length (mm/in):',
'Position', [20, 180, 120, 22]);
    freeLengthEdit = uieditfield(fig, 'numeric', 'Position', [150, 180, 70,
22]);

    % SOLID LENGTH
    solidLengthLabel = uilabel(fig, 'Text', 'Solid Length (mm/in):',
'Position', [20, 150, 120, 22]);
    solidLengthEdit = uieditfield(fig, 'numeric', 'Position', [150, 150, 70,
22]);

```

```

    % Input Fmin and Fmax
    fMinLabel = uilabel(fig, 'Text', 'Fmin [N/lbf] (write 0 if static load is
applied):', 'Position', [20, 120, 250, 22]);
    fMaxLabel = uilabel(fig, 'Text', 'Fmax [N/lbf]:', 'Position', [20, 90,
120, 22]);
    fMinEdit = uieditfield(fig, 'numeric', 'Position', [250, 120, 70, 22]);
    fMaxEdit = uieditfield(fig, 'numeric', 'Position', [150, 90, 70, 22]);

    % Peening Status
    peeningStatusLabel = uilabel(fig, 'Text', 'Peening Status:', 'Position',
[20, 60, 120, 22]);
    peeningStatusDropDown = uidropdown(fig, 'Items', {'', 'Peened',
'Unpeened'}, 'Position', [150, 60, 100, 22]);

    calculateButton = uibutton(fig, 'push', 'Text', 'Calculate', 'Position',
[250, 20, 100, 40], ...
    'ButtonPushedFcn', @(btn, event) calculateSpring(...
        endTypeDropDown.Value, materialDropDown.Value,
unitsDropDown.Value, wireDiameterEdit.Value, ...
        outerDiameterEdit.Value, freeLengthEdit.Value, solidLengthEdit.Value,
...
        fMinEdit.Value, fMaxEdit.Value, peeningStatusDropDown.Value));

end

```

calculate spring properties

Callback function to calculate spring properties

```

function calculateSpring(endType, material, units, wireDiameter,
outerDiameter, freeLength, solidLength, fMin, fMax, peenedStatus)

```

Quality control

Check if end type and material are selected

```

if isempty(endType) && isempty(material)
    errordlg('Please select End Type and Material', 'Error', 'modal');
    return; % Exit the function if not selected

```

```

elseif isempty(endType)
    errordlg('Please select End Type', 'Error', 'modal');
    return; % Exit the function if not selected
elseif isempty(material)
    errordlg('Please select Material', 'Error', 'modal');
    return; % Exit the function if not selected
end

% Verification check for numeric values
if ~isnumeric(wireDiameter) || ~isnumeric(outerDiameter) ||
~isnumeric(freeLength) || ~isnumeric(solidLength)
    errordlg('Please enter numeric values for diameters and lengths',
'Error', 'modal');
    return;
end

% Wire diameter verification
if wireDiameter < 0
    errordlg('Invalid wire diameter. Please enter a positive value.',
'Error', 'modal');
    return;
elseif wireDiameter == 0
    errordlg('Invalid wire diameter. Please enter a positive, non-zero
value.', 'Error', 'modal');
    return;
end

% Outer diameter verification
if outerDiameter < 0
    errordlg('Invalid outer diameter. Please enter a positive value.',
'Error', 'modal');
    return;
elseif outerDiameter == 0
    errordlg('Invalid outer diameter. Please enter a positive, non-zero
value.', 'Error', 'modal');
    return;
end

% Free length verification
if freeLength <= solidLength || freeLength < 0
    errordlg('Invalid Free Length. Please enter a positive value greater
than Solid Length.', 'Error', 'modal');
    return;
elseif freeLength == 0
    errordlg('Invalid Free Length. Please enter a non-zero, positive
value greater than Solid Length', 'Error', 'modal');
    return;
end

% Solid length verification
if solidLength < 0
    errordlg('Invalid Solid Length. Please enter a positive value.',
'Error', 'modal');
    return;
end

```

```

elseif solidLength == 0
    errordlg('Invalid Solid Length. Please enter a non-zero, positive
value.', 'Error', 'modal');
    return;
end

%peened or unpeened verification
if fMin ~= 0 && isempty(peenedStatus)
    errordlg('If Fmin is not zero, peened or unpeened must be selected.',
'Error');
    return;
elseif fMin == 0 && ~isempty(peenedStatus)
    errordlg('If Fmin is zero, neither peened nor unpeened should be
selected.', 'Error');
    return;
end

```

Check if the units are in English or Metric

```

if strcmp(units, 'English')
    % Convert to mm from inches and pounds based on Michael's test case
    convertToMetric_length = @(value) value * 25.4; % from inches to mm
    convertToMetric_force = @(value) value * 4.44822; %from pound force
to newtons

    wireDiameter = convertToMetric_length(wireDiameter);
    solidLength = convertToMetric_length(solidLength);
    freeLength = convertToMetric_length(freeLength);
    outerDiameter = convertToMetric_length(outerDiameter);
    fMax = convertToMetric_force(fMax);
    fMin = convertToMetric_force(fMin);
    % Add any other conversions if needed
end
% Now user inputs are either in metric or remain unchanged

```

Call other functions to calculate and display results

```

%use round to ensure the outcome is an integer
totalCoils = round(calculateTotalCoils(endType, wireDiameter,
solidLength));
activeCoils = round(calculateActiveCoils(endType, totalCoils));
pitch = calculatePitch(activeCoils, freeLength, wireDiameter,
endType);
springRate = calculateSpringRate(wireDiameter, outerDiameter,
activeCoils, material);
force = calculateForce(freeLength, solidLength, springRate);
force_FOS = calculateStaticFOS(material, wireDiameter, force,
outerDiameter); % fos calculated from force needed to compress the spring to
max length

```

```

        %calculate FOS - verify if static or inf life
        if fMin == 0
            fos = calculateStaticFOS(material, wireDiameter, fMax,
outerDiameter);
        else
            fos = calculateInfFOS(fMin, fMax, outerDiameter, wireDiameter,
peenedStatus, material);
        end

        % Display the results in a new figure
        displayResultsFigure(totalCoils, activeCoils, pitch, springRate,
force, fMin, fos, force_FOS, units);

    end

```

calculate total coils - Shigley Table 10-1

```

function totalCoils = calculateTotalCoils(endType, d, Ls)
    % Default value, in case the end type is not recognized
    totalCoils = 0;

    % Use a switch statement to handle different end types
    switch endType
        case 'Plain'
            totalCoils = Ls / d - 1;

        case 'Squared or closed'
            totalCoils = Ls / d - 1;

        case 'Plain and ground'
            totalCoils = Ls / d;

        case 'Squared and ground'
            totalCoils = Ls / d;

    end
end

```

calculate active coils - Shigley Table 10-1

```

function activeCoils = calculateActiveCoils(endType, Nt)
    % Default value, in case the end type is not recognized
    activeCoils = 0;

    % Use a switch statement to handle different end types
    switch endType
        case 'Plain'
            activeCoils = Nt;

        case 'Plain and ground'
            activeCoils = Nt - 1;

        case 'Squared or closed'
            activeCoils = Nt - 2;
    end
end

```

```

        case 'Squared and ground'
            activeCoils = Nt - 2;
        end
    end
end

```

calculate pitch - Shigley Table 10-1

```

function pitch = calculatePitch(Na, L0, d, endType)
    % Default value, in case the end type is not recognized
    pitch = 0;

    % Use a switch statement to handle different end types
    switch endType
        case 'Plain'
            pitch = (L0-d)/Na;

        case 'Plain and ground'
            pitch = L0/(Na+1);

        case 'Squared or closed'
            pitch = (L0 - 3*d)/Na;

        case 'Squared and ground'
            pitch = (L0 - 2*d)/Na;
    end
end

```

calculate spring rate k

```

function springRate = calculateSpringRate(d, Do, Na, material)
    %springRate = 0;

    %Do = outer diameter
    D = Do - d; %mean diameter of the spring

    % Use a switch statement to handle different materials
    switch material
        case 'Music wire A228'
            %Table 10-5 Shigley
            if d < 0.8128 % converted to mm
                G = 82.7; %Gpa
            elseif d < 1.6256
                G = 81.7; %GPa
            elseif d <= 3.175
                G = 81.0; %GPa
            elseif d > 3.175
                G = 80.0; %GPa
            end

        case 'Hard-drawn wire A227'
            %Table 10-5 Shigley
            if d < 0.8128

```

```

        G = 80.7; %Gpa
    elseif d < 1.6256
        G = 80.0; %GPa
    elseif d <= 3.175
        G = 79.3; %GPa
    elseif d > 3.175
        G = 78.6; %GPa
    end

    case 'Chrome-vanadium wire A232'
        %Table 10-5 Shigley
        G = 77.2; %GPa

    case 'Chrome-silicon wire A401'
        %Table 10-5 Shigley
        G = 77.2; %GPa

    case '302 stainless wire A313'
        %Table 10-5 Shigley
        G = 69.0; %GPa

    case 'Phosphor-bronze wire B159'
        %Table 10-5 Shigley
        G = 41.4; %GPa
    end

    springRate = (d/1000)^4 * (G*10^9) / (8 * (D/1000)^3 * Na); %k, N/m

end

```

calculate force - The force needed to compress the spring to its solid length

```

function force = calculateForce(L0, Ls, k)
    force = k * ((L0 - Ls)/1000); %F=kx, newtons
end

```

calculate factor of safety - STATIC

```

function fos_static = calculateStaticFOS(material, d, F, D)
    % Use a switch statement to handle different materials
    switch material
        case 'Music wire A228'
            %Table 10-4 Shigley
            A = 2211; %MPa * mm^m
            m = 0.145; %exponent
            elasticpercent = .45;

        case 'Hard-drawn wire A227'
            %Table 10-4 Shigley
            A = 1783; %MPa * mm^m

```

```

        m = 0.190; %exponent
        elasticpercent = .45;

    case 'Chrome-vanadium wire A232'
        %Table 10-4 Shigley
        A = 2005; %MPa * mm^m
        m = 0.168; %exponent
        elasticpercent = .65;

    case 'Chrome-silicon wire A401'
        %Table 10-4 Shigley
        A = 1974; %MPa * mm^m
        m = 0.108; %exponent
        elasticpercent = .65;

    case '302 stainless wire A313'
        %Table 10-4 Shigley
        if d < 2.5
            A = 1867;
            m = 0.146;
        elseif d < 5
            A = 2065;
            m = 0.263;
        elseif d <= 10
            A = 2911;
            m = 0.478;
        end
        elasticpercent = .45;

    case 'Phosphor-bronze wire B159'
        %Table 10-4 Shigley
        if d < 0.6
            A = 1000;
            m = 0;
        elseif d < 2
            A = 913;
            m = 0.028;
        elseif d <= 7.5
            A = 932;
            m = 0.064;
        end
        elasticpercent = .45;
end

S_ut = A / (d^m);
S_sy = elasticpercent*S_ut;
tao = 8*F*D/(pi*d^3) + 4*F/(pi*d^2);
fos_static = S_sy / tao;
end

```

FOS inf life

```
function fos_inf = calculateInfFOS(Fmin, Fmax, D, d, peenedStatus,
material)
    Fa = (Fmax-Fmin)/2;
    Fm = (Fmax+Fmin)/2;

    switch material
        case 'Music wire A228'
            %Table 10-4 Shigley
            A = 2211; %MPa * mm^m
            m = 0.145; %exponent

        case 'Hard-drawn wire A227'
            %Table 10-4 Shigley
            A = 1783; %MPa * mm^m
            m = 0.190; %exponent

        case 'Chrome-vanadium wire A232'
            %Table 10-4 Shigley
            A = 2005; %MPa * mm^m
            m = 0.168; %exponent

        case 'Chrome-silicon wire A401'
            %Table 10-4 Shigley
            A = 1974; %MPa * mm^m
            m = 0.108; %exponent

        case '302 stainless wire A313'
            %Table 10-4 Shigley
            if d < 2.5
                A = 1867;
                m = 0.146;
            elseif d < 5
                A = 2065;
                m = 0.263;
            elseif d <= 10
                A = 2911;
                m = 0.478;
            end

        case 'Phosphor-bronze wire B159'
            %Table 10-4 Shigley
            if d < 0.6
                A = 1000;
                m = 0;
            elseif d < 2
                A = 913;
                m = 0.028;
            elseif d <= 7.5
                A = 932;
                m = 0.064;
            end
    end
end
```

```

end
C = D / d;
Kb = (4*C + 2)/(4*C - 3);
tao_a = Kb * (8*Fa*D)/(pi*d^3);
tao_m = Kb * (8*Fm*D)/(pi*d^3);
S_ut = A / (d^m);
S_su = 0.67*S_ut;
%add S_se based on peened or unpeened
switch peenedStatus
    case 'Peened'
        S_sa = 398; %MPa
        S_sm = 534; % MPa

    case 'Unpeened'
        S_sa = 241; %MPa
        S_sm = 379; % MPa

end

S_se = S_sa / (1 - S_sm/S_su);

middleStep = (tao_a / S_se + tao_m / S_su);
fos_inf = 1/middleStep;

%fos_inf = (tao_a / S_se + tao_m / S_su)^(-1);

end

```

Display the results in a new figure

```

function displayResultsFigure(totalCoils, activeCoils, pitch, springRate,
force, fMin, fos, force_FOS, units)
    % Create a new figure
    resultsFig = uifigure('Name', 'Spring Results', 'Position', [600, 300,
800, 150]);

    % Create uicontrols to display the results
    uicontrol(resultsFig, 'Style', 'text', 'Position', [20, 120, 300, 20],
'String', ['Total Coils, Nt: ' num2str(totalCoils)]);
    uicontrol(resultsFig, 'Style', 'text', 'Position', [20, 90, 300, 20],
'String', ['Active Coils, Na: ' num2str(activeCoils)]);

    % Check if the units are in English and perform conversions if needed
    if strcmp(units, 'English')
        convertToEnglish_length = @(value) value / 25.4; % from m to inches
        convertToEnglish_springRate = @(value) value * 0.00571; % from N/m to
lbf/in
        convertToEnglish_force = @(value) value * 0.224809; %from newtons to
pound force

        % Convert results to English units (replace with your conversion
logic)
    end

```

```

pitch = convertToEnglish_length(pitch);
springRate = convertToEnglish_springRate(springRate);
force = convertToEnglish_force(force);

    uicontrol(resultsFig, 'Style', 'text', 'Position', [20, 60, 300, 20],
'String', ['Pitch, p [in]: ' sprintf('%.4f', pitch)]);
    uicontrol(resultsFig, 'Style', 'text', 'Position', [20, 30, 300, 20],
'String', ['Spring Rate, k [lbf/in]: ' sprintf('%.3f', springRate)]);
    uicontrol(resultsFig, 'Style', 'text', 'Position', [400, 120, 400,
20], 'String', ['Force needed to compress spring, F [lbf]: ' sprintf('%.3f',
force)]);
    else
        uicontrol(resultsFig, 'Style', 'text', 'Position', [20, 60, 300, 20],
'String', ['Pitch, p [mm]: ' sprintf('%.4f', pitch)]);
        uicontrol(resultsFig, 'Style', 'text', 'Position', [20, 30, 300, 20],
'String', ['Spring Rate, k [N/m]: ' sprintf('%.3f', springRate)]);
        uicontrol(resultsFig, 'Style', 'text', 'Position', [400, 120, 400,
20], 'String', ['Force needed to compress spring, F [N]: ' sprintf('%.3f',
force)]);
    end

    uicontrol(resultsFig, 'Style', 'text', 'Position', [400, 90, 400,
20], 'String', ['Associated FOS with Force needed to compress spring: '
sprintf('%.1f', force_FOS)]);

    % Display Factor of Safety with one decimal place
    if fMin == 0
        uicontrol(resultsFig, 'Style', 'text', 'Position', [400, 60, 400,
20], 'String', ['Factor of Safety (static): ' sprintf('%.1f', fos)]);
    else
        uicontrol(resultsFig, 'Style', 'text', 'Position', [400, 60, 400,
20], 'String', ['Factor of Safety (inf life): ' sprintf('%.1f', fos)]);
    end

end

```

Published with MATLAB® R2023b