

University of Wales Trinity Saint David

Wales Institute of Science and Art



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Analytical Assessment of Engine Internal Components

Contemporary Powertrain Design | ACHA6006

Joao Ferreira

1902427

March 2025

Table of Contents

Table of Figures	2
1 Introduction	3
2 Assessment of Current Engine Component Design	4
2.1 Data Collection	4
2.1.1 Base Line Data Gathering	4
2.1.2 Component Geometry	5
2.2 First Principles Calculations.....	5
3 Connecting Rod Analysis	11
3.1 Original Connecting Rod Stresses.....	11
3.2 Recalculate Safety Factors at Increased RPM	12
3.3 New Connecting Rod Force Calculations	14
3.4 Connecting Rod After Treatment	15
4 Gudgeon Pin Analysis.....	16
4.1 Original Stresses	16
4.2 New Connecting Gudgeon Pin Stresses.....	17
4.3 Gudgeon Pin After Treatment.....	18
5 Bolt Analysis.....	19
5.1 Original Stresses	19
6 Determine Optimal Crankshaft Configuration	21
7 References	23

Table of Figures

Figure 1 - Ford 1.6 GTDI EcoBoost (Ford Motor Company, no date)	3
Figure 2 - Data Import into MATLAB	4
Figure 3 - Piston Kinematics Diagram (Hugo et al., 2017)	5
Figure 4 - Connecting Rod Angle Calculation Code	5
Figure 5 - Distance From TDC Code	6
Figure 6 - Distance from TDC vs Crank Angle	6
Figure 7 - Piston Velocity Code for Calculations and Plot	7
Figure 8 - Piston Position vs Piston Velocity @ 6000 RPM	7
Figure 9 - Piston Acceleration Calculation Code and Plot	8
Figure 10 - Piston Acceleration vs Crank Angle and RPM	8
Figure 11 - Reciprocating Mass Calculation Code	9
Figure 12 - Piston Inertia Force Calculation Code	9
Figure 13 - Gas Force Calculation Code	9
Figure 14 - Net Force Calculation Code	9
Figure 15 - Rod Force Calculation Code	9
Figure 16 - Side Force Calculation Code	10
Figure 17 - Side Friction Calculation Code	10
Figure 18 - Rod Stress Calculation Code and Display Results	11
Figure 19 - Original Rod Stresses and Factor of Safety	11
Figure 20 - Interpolation of Cylinder Pressures and Verification Plot	12
Figure 21 - Pressure vs Crank Angle at 6000, 7000, 8000 and 9000 RPM	13
Figure 22 - New Rod Stress Calculations at 9000 RPM	14
Figure 23 - New Rod Stresses and Factor of Safety at 9000 RPM	14
Figure 24 - Quenched and Tempered Rod Stress Calculations at 9000 RPM	15
Figure 25 - Quenched and Tempered Rod Stress Results at 9000 RPM	15
Figure 26 - Code for Original Gudgeon Pin Stresses	16
Figure 27 - Original Stresses on Gudgeon Pin Results	16
Figure 28 - Gudgeon Pin Stresses Code for 9000 RPM	17
Figure 29 - Original Stresses on Gudgeon Pin Results at 9000 RPM	17
Figure 30 - Rod Cap Stress Calculation Code	19
Figure 31 - Rod Cap Stress Results	19
Figure 32 - Bolt Stresses Code	20
Figure 33 - Bolt Stress Results	20
Figure 34 - Torque Calculation Code Part 1	21
Figure 35 - Torque Calculation Code Part 2	21

1 Introduction

The internal components of an engine play a crucial role in its performance, efficiency, and durability. In modern powertrain design, understanding the mechanical stresses and forces acting upon these components is essential to optimize performance and reliability. This report provides an analytical assessment of key internal components within an internal combustion engine, specifically focusing on the Ford EcoBoost 1.6 GTDI.



Figure 1 - Ford 1.6 GTDI EcoBoost (Ford Motor Company, no date)

The analysis includes the treatment of supplied data and stress calculations for critical components such as the connecting rod, gudgeon pin, and bolts. Using first-principles calculations and MATLAB simulations, the study evaluates the original design of these components under various operational conditions, including increased RPM scenarios. The results allow for an assessment of potential design improvements and material treatment methods, such as quenching and tempering, to enhance durability and safety margins.

Additionally, the report investigates the implications of different crankshaft configurations—flat-plane versus cross-plane—on engine performance, discussing their respective advantages and limitations in high-performance applications. The findings aim to provide insights into optimizing internal engine components for improved efficiency, longevity, and mechanical resilience.

2 Assessment of Current Engine Component Design

2.1 Data Collection

2.1.1 Base Line Data Gathering

To gain a deeper understanding of the original specifications of the Ford 1.6 engine, the provided data was first imported into MATLAB. This allowed for more detailed analysis and visualization of the engine's performance characteristics.

Ford Sigma Data Analysis

Contemporary Powertrain Design - ACHA6006

Joao Ferreira - 1902427

Load Given Data:

```
% Load the data from the MAT-file
loadedData = load('SigmaPressureData.mat');

% Extract the table (adjust the field name if necessary)
sigmaPressureData = loadedData.sigmaPressureData;

% Rename table columns for clarity
sigmaPressureData.Properties.VariableNames = {'Crank_Angle', '1000rpm', '2000rpm', '3000rpm', '4000rpm', '5000rpm', '6000rpm'};

% Define physical constants
Piston_Mass = 258 / 1000; % Piston Mass in kg
Rod_Mass = 372/1000;
Ring_Mass = 9 / 1000; % Piston Ring Mass in kg
Pin_Mass = 80 / 1000; % Pin Mass in kg
Clips_Mass = 5 / 1000; % Clips Mass in kg
Rotat_Mass = 313 / 1000; % Rotational Mass in kg
Bore_Dia = 79 / 1000; % Bore Diameter in m
Piston_Area = (pi * (Bore_Dia^2))/4; % Piston Surface Area in m^2
Crank_Rad = 40.7 / 1000; % Crank Radius in m
Rod_Length = 134 ./ 1000; % Rod Length in m
Pist_Exposed_Mass = 23.28/1000;
Rod_CSAmmsq = 139.65;
L1 = 7.75/1000;
L2 = 7/1000;
D1 = 12.3/1000;
D2 = 21/1000;
Cap_Mass = 105/1000;
Bolt_Mass = 35/1000;
Head_Dia = 12.9/1000;
Cap_Int_CSAmmsq = 210;
Mod_Elas_Cap = 170000;
Rod_Int_Bolt = 25/1000;
Bolt_CSAmmsq = 42.43;
Mod_Elas_Bolt = 210000;
Lengh_Sec_Thread = 27.26/1000;
Thread_Lenght = 32/1000;

% Extract Crank Angle and Pressure Data
Crank_Angle = sigmaPressureData(:, 1)'; % Crank Angle in degrees
RPM_Values = [1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000]'; % RPM values corresponding to columns 2 to 7
PressureData = sigmaPressureData(:, 2:7)'; % Transpose to align dimensions for plotting
```

Figure 2 - Data Import into MATLAB

As shown in *Figure 2*, the initial section of code was designed to import all the provided .csv data files into MATLAB. This included average cylinder pressure data, RPM values, and the constant values specified in the original dataset. Importing this data established the foundation for subsequent analysis and processing.

2.1.2 Component Geometry

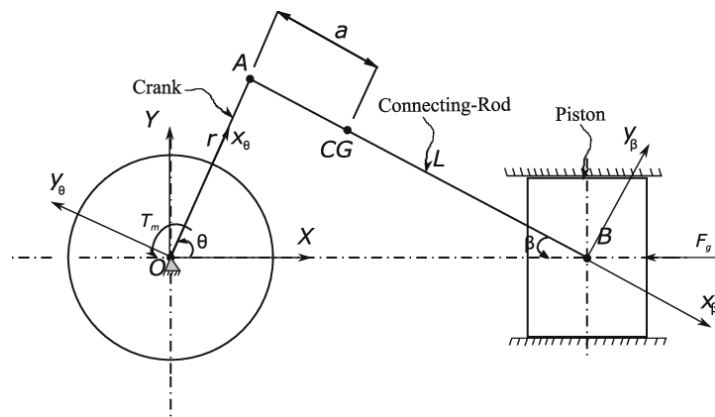


Figure 3 - Piston Kinematics Diagram (Hugo et al., 2017)

To calculate the distance from top dead centre (TDC) as the engine crank rotates, it is essential to first understand the relationship between the crank angle and the resulting piston position.

At TDC, the connecting rod is fully aligned with the gudgeon pin and the crankshaft centreline, resulting in maximum piston height, which equals the sum of the crankshaft radius and the connecting rod length.

As the crankshaft rotates, the distance from TDC becomes the hypotenuse of a triangle formed by the crankshaft radius and the connecting rod length. The direct distance between the crank centreline and the gudgeon pin is no longer a simple sum of the radius and rod length but is instead governed by a trigonometric relationship involving the crankshaft angle.

To compute this distance accurately, the rod angle must first be calculated. This requires resolving the triangle formed by the crankshaft, connecting rod, and piston position using trigonometric functions based on the crankshaft angle and the lengths of the crankshaft radius and connecting rod.

2.2 First Principles Calculations

Rod Angle

```
Rod_Angle = asind((Crank_Rad * sind(Crank_Angle)) ./ Rod_Length);
```

Figure 4 - Connecting Rod Angle Calculation Code

Once the rod angle has been determined for each crankshaft angle, the distance from TDC can be calculated. This is achieved by resolving the trigonometric relationship between the crankshaft radius,

the connecting rod length, and the rod angle. This calculation allows for an accurate determination of the piston's position relative to TDC throughout the crankshaft's rotation.

Distance From TDC

```
Dist_From_TDC = (Crank_Rad + Rod_Length) - (Crank_Rad * cosd(Crank_Angle) + Rod_Length * cosd(Rod_Angle));  
  
Dist_From_TDC_mm = Dist_From_TDC/1000;  
  
% Plot Dist_From_TDC vs Crank_Angle  
figure;  
plot(real(Crank_Angle), real(Dist_From_TDC), 'LineWidth', 1.5);  
xlabel('Crank Angle (Degrees)');  
ylabel('Distance from TDC (m)');  
title('Piston Distance from TDC vs. Crank Angle');  
grid on;  
  
% Set x-axis limits from 0 to 720 degrees  
xlim([-360 360]);
```

Figure 5 - Distance From TDC Code

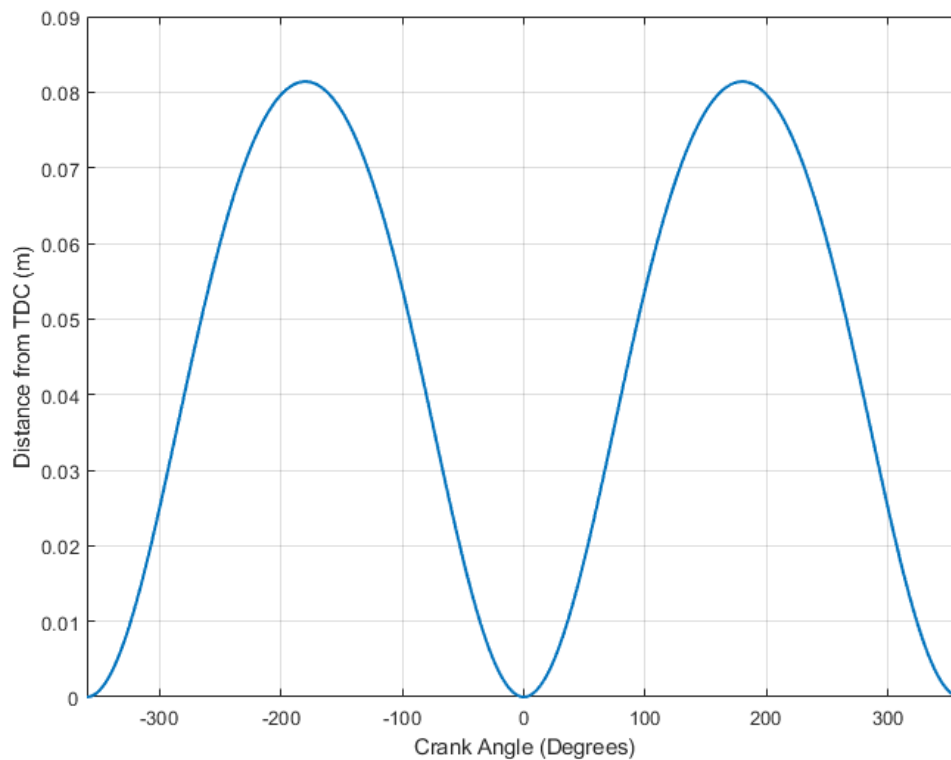


Figure 6 - Distance from TDC vs Crank Angle

This data can then be plotted against the crankshaft angle, as shown in **Error! Reference source not found..** This visualization provides a clear representation of the piston's position relative to TDC throughout the crankshaft's rotation.

Using this information and the RPM values, the piston velocity can be calculated and plotted.

Velocity

```

Crank_Ang_Vel = RPM_Values .* ((2 .* pi) ./ 60);

Piston_Velocity = Crank_Rad * Crank_Ang_Vel * sind(Crank_Angle) + ((Crank_Rad^2 * Crank_Ang_Vel) / (2 * Rod_Length)) * sind(2 * Crank_Angle);

Piston_Vel_6000RPM = (Piston_Velocity(6, :)) .* 1000;

% Create Plot with Dual Y-Axes
figure;

% Left Y-Axis: Piston Distance from TDC
yyaxis left
plot(Crank_Angle, Dist_From_TDC_mm, 'LineWidth', 2, 'Color', get(gca, 'YColor'));
ylabel('Piston Distance From TDC (mm)');

% Right Y-Axis: Piston Velocity at 6000 RPM
yyaxis right
plot(Crank_Angle, Piston_Vel_6000RPM, 'LineWidth', 2, 'Color', get(gca, 'YColor'));
ylabel('Piston Velocity (mm/s)');

% Set right y-axis to scientific notation (10^4)
ax = gca;
ax.YAxis(2).Exponent = 4;
ytickformat('%1f');

% X-Axis: Crank Angle
xlabel('Crank Angle (degrees)');

% Title and Grid
title('Piston Position vs Piston Velocity @ 6000 RPM');
grid on;

% Set x-axis limits from 0 to 720 degrees
xlim([-360 360]);

```

Figure 7 - Piston Velocity Code for Calculations and Plot

After calculating the piston velocity across the RPM range and plotting it against the distance from TDC, it becomes evident that the velocity peaks just before the piston reaches TDC. Following this peak, the velocity rapidly transitions into the negative direction as the piston accelerates toward BDC, as seen in *Figure 8*. This behaviour reflects the dynamic nature of the crank-rod mechanism and the influence of crankshaft rotation on piston motion.

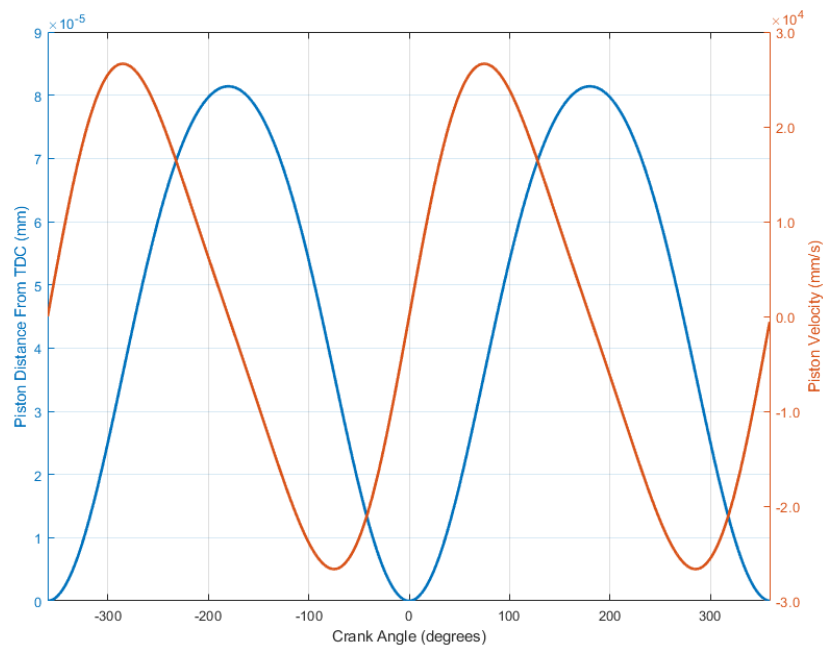


Figure 8 - Piston Position vs Piston Velocity @ 6000 RPM

Piston acceleration can then be determined by taking the derivative of the previously calculated piston velocity. This provides insight into the forces acting on the piston throughout the crankshaft cycle, helping to analyse the dynamic loading and stress on the engine components.

Acceleration

```
Piston_Accel = Crank_Rad .* Crank_Ang_Vel.^2 .* cosd(Crank_Angle) + ...
    ((Crank_Rad.^2 / Rod_Length) .* Crank_Ang_Vel.^2 .* cosd(2 .* Crank_Angle));

% Create meshgrid for Crank_Angle and RPM_Values
[Crank_Angle_Grid, RPM_Grid] = meshgrid(Crank_Angle, RPM_Values);

% 3D Surface Plot
figure;
surf(RPM_Grid, Crank_Angle_Grid, Piston_Accel, 'EdgeColor', 'none');
colormap('jet');
colorbar; % Show color scale
xlabel('RPM');
ylabel('Crank Angle (°)');
zlabel('Acceleration (mm/s^2)');
title('Piston Acceleration vs Crank Angle and RPM');
grid on;
view(135, 30); % Adjust view angle if desired

% Invert the RPM (x-axis) direction
set(gca, 'XDir', 'reverse');

% Set RPM steps of 1000
xticks(1000:1000:max(RPM_Values));
```

Figure 9 - Piston Acceleration Calculation Code and Plot

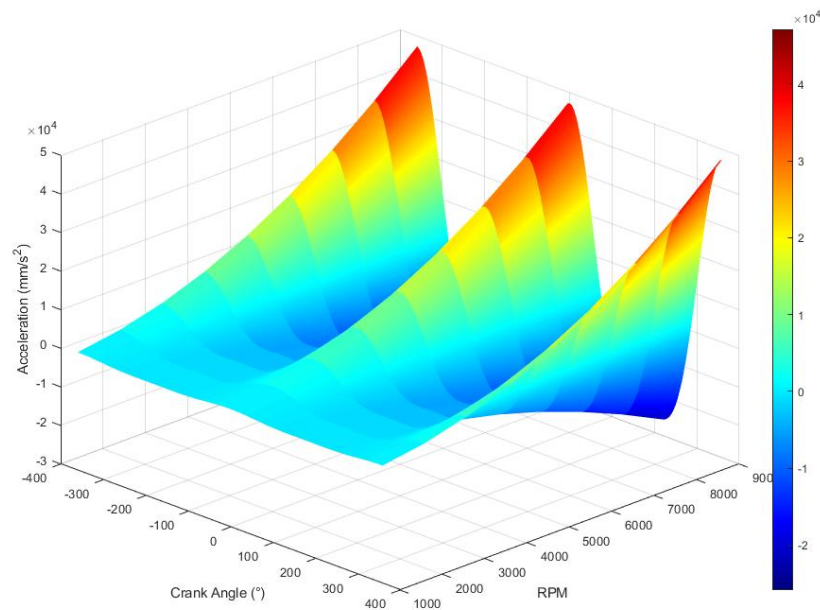


Figure 10 - Piston Acceleration vs Crank Angle and RPM

Reciprocating Mass

```
Recip_Mass = Piston_Mass + Pin_Mass + (Rod_Mass/3);
```

Figure 11 - Reciprocating Mass Calculation Code

The reciprocating mass can be calculated by summing the masses of the piston, the gudgeon pin, and one-third of the connecting rod mass. This value is essential for determining the piston's inertia force, which can be calculated using the product of the reciprocating mass and the previously obtained piston acceleration.

Piston Inertia Force

```
Piston_Inertia_Force = Recip_Mass .* Piston_Accel;
```

Figure 12 - Piston Inertia Force Calculation Code

The gas force can be calculated as the product of the cylinder pressure data and the piston crown area. This allows for the calculation of the net force acting on the piston by combining the gas force with the inertia force. From this, the rod force can be determined, providing a complete picture of the forces transmitted through the connecting rod during the engine cycle.

Gas Force

```
Gas_Force = PressureData .* Piston_Area;
```

Figure 13 - Gas Force Calculation Code

Net Force

```
Piston_Inertia_Force_Resized = reshape(Piston_Inertia_Force(1:(6*720)), 6, 720);  
Net_Force = Gas_Force - Piston_Inertia_Force_Resized;
```

Figure 14 - Net Force Calculation Code

Rod Force

```
Rod_Force = Net_Force ./ cosd(Rod_Angle);
```

Figure 15 - Rod Force Calculation Code

To calculate the forces exerted on the sides of the bore walls, the rod angle and the net force acting on the piston must be considered. The lateral or side force can be determined by resolving the net force into components along the axis of the connecting rod and perpendicular to the cylinder wall.

Side Force

```
Side_Force = Net_Force .* tand(Rod_Angle)
```

Figure 16 - Side Force Calculation Code

Once the side force is known, the frictional force can be estimated using the coefficient of friction between the piston and the bore wall. This provides insight into the mechanical losses within the engine and helps identify areas for potential efficiency improvements.

Side Friction

```
Side_Friction = 0.1 .* Side_Force
```

Figure 17 - Side Friction Calculation Code

3 Connecting Rod Analysis

3.1 Original Connecting Rod Stresses

To calculate the connecting rod stresses in the original specification engine, the forces acting on the rod are divided by its cross-sectional area to determine the stress values. From this the minimum (compressive) and maximum (tensile) stresses can be extracted.

Rod Stress

```
Rod_Stress = Rod_Force ./ Rod_CSAmmsq

% Find minimum (compressive) stress and its position
[Compressive_Stress, minIndex] = min(Rod_Stress(:));
[rowMin, colMin] = ind2sub(size(Rod_Stress), minIndex);

% Find maximum (tensile) stress and its position
[Tensile_Stress, maxIndex] = max(Rod_Stress(:));
[rowMax, colMax] = ind2sub(size(Rod_Stress), maxIndex);

% Calculate stress range and mean stress
Stress_Range = Tensile_Stress - Compressive_Stress;
Stress_Mean = (Tensile_Stress + Compressive_Stress) / 2;

% Display results
fprintf('Minimum Compressive Stress: %.2f at Row %d, Column %d\n', Compressive_Stress, rowMin, colMin);
fprintf('Maximum Tensile Stress: %.2f at Row %d, Column %d\n', Tensile_Stress, rowMax, colMax);
fprintf('Stress Range: %.2f\n', Stress_Range);
fprintf('Mean Stress: %.2f\n', Stress_Mean);

UTS = 750;

EndLin = 0.45 .* UTS;

Stress_Allow = 2 .* EndLin .* (1 - (Stress_Mean ./ UTS));

FSF = Stress_Allow ./ Stress_Range;

fprintf('Factor of Safety (FSF): %.2f\n', FSF);
```

Figure 18 - Rod Stress Calculation Code and Display Results

From these values, the stress range and mean stress can be calculated. By comparing these stresses to the ultimate tensile strength (UTS) and the endurance limit of the material, fatigue safety factors can be determined. This analysis helps assess the connecting rod's ability to withstand cyclic loading and ensures that the design meets the required durability and performance standards.

```
Minimum Compressive Stress: -155.93 at Row 3, Column 2
Maximum Tensile Stress: 87.72 at Row 3, Column 239
Stress Range: 243.65
Mean Stress: -34.10
Factor of Safety (FSF): 2.90
```

Figure 19 - Original Rod Stresses and Factor of Safety

The fatigue safety factor of the original connecting rod was calculated to be 2.9. This indicates that the connecting rod can withstand cyclic stresses up to 2.9 times higher than the expected operational stresses before reaching failure, providing a reasonable margin of safety against fatigue.

3.2 Recalculate Safety Factors at Increased RPM

To assess how the connecting rod would react to increased engine speeds, we need to calculate the pressure data for the extended RPM range, as this data is not directly provided. This can be accomplished by mathematically extrapolating the original pressure data, assuming that the pressure follows a predictable trend relative to changes in engine speed.

Interpolation of Pressure Data to 9000rpm

```
% --- Given Variables (assumed loaded in your workspace) ---
% Crank_Angle: Crank angle in degrees from the first column of sigmaPressureData
Crank_Angle = sigmaPressureData(:, 1)';

% RPM_Values: Original RPM values corresponding to columns 2 to 7 (6 values)
RPM_Values = [1000, 2000, 3000, 4000, 5000, 6000]';

% PressureData: Extract pressure data from sigmaPressureData
% Extract from columns 2 to 7 and transpose so rows correspond to crank angles
PressureData = sigmaPressureData(:, 2:7)';
PressureData = PressureData'; % Now, size(PressureData) is [length(Crank_Angle) x 6]

% --- Pressure at 6000 RPM (directly from original data) ---
% Multiply by 1e5 to convert to Pascals
Pressure_6000 = PressureData(:, 6) * 1e5;

% --- Define new RPM values for extrapolation: 7000, 8000, and 9000 RPM ---
newRPM = [7000, 8000, 9000]';

% Preallocate matrix for interpolated pressures for new RPM values
Pressure_new = zeros(length(Crank_Angle), length(newRPM));

% Interpolate pressure data for each crank angle across the new RPM values
for i = 1:length(Crank_Angle)
    Pressure_new(i, :) = interp1(RPM_Values, PressureData(i, :), newRPM, 'linear', 'extrap');
end

% Convert interpolated pressures to Pascals
Pressure_new_Pa = Pressure_new * 1e5;

% --- Extract pressures for each new RPM ---
Pressure_7000 = Pressure_new_Pa(:, 1);
Pressure_8000 = Pressure_new_Pa(:, 2);
Pressure_9000 = Pressure_new_Pa(:, 3);

% --- Plot Pressure vs Crank Angle for 6000, 7000, 8000, and 9000 RPM ---
figure;
plot(Crank_Angle, Pressure_6000, 'b-', 'Linewidth', 2); hold on;
plot(Crank_Angle, Pressure_7000, 'r--', 'Linewidth', 2);
plot(Crank_Angle, Pressure_8000, 'g-.', 'Linewidth', 2);
plot(Crank_Angle, Pressure_9000, 'k:', 'Linewidth', 2);
xlabel('Crank Angle (degrees)');
ylabel('Pressure (Pa)');
legend('6000 RPM', '7000 RPM', '8000 RPM', '9000 RPM');
title('Pressure vs Crank Angle at 6000, 7000, 8000 and 9000 RPM');
grid on;
```

Figure 20 - Interpolation of Cylinder Pressures and Verification Plot

To verify the data obtained through extrapolation, a comparison plot (Figure 21) can be created between the original test data at 6000 RPM and the newly calculated values for 7000, 8000, and 9000 RPM. This plot visually demonstrates how the extrapolated data compares to the original test data, helping to assess the accuracy and reliability of the extrapolation method. By analysing the trends and deviations, we can ensure that the extrapolated pressure values are reasonable and reflect the expected behaviour of the engine at higher speeds.

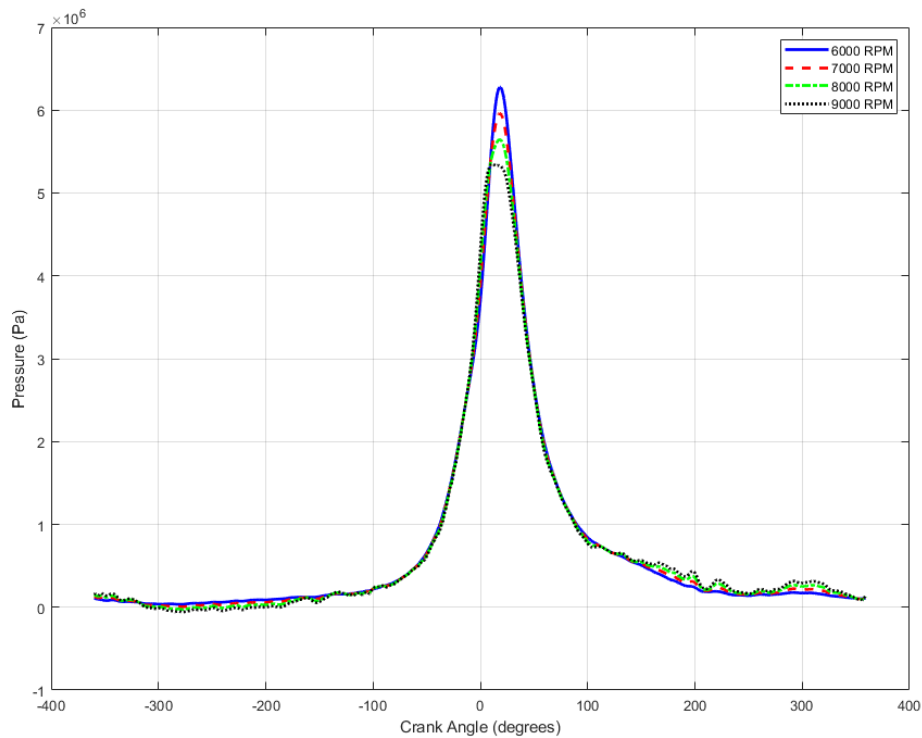


Figure 21 - Pressure vs Crank Angle at 6000, 7000, 8000 and 9000 RPM

At higher engine speeds, the time available for combustion decreases, leading to less efficient combustion and a reduction in peak cylinder pressure. This trend aligns with findings, which highlight that increased RPM can cause changes in dynamics and the balance between compression and expansion, ultimately reducing peak pressures at higher speeds.

For instance, research indicates that as engine speed increases, the maximum in-cylinder pressure decreases. This phenomenon is attributed to the combustion process, which becomes less synchronized with increasing RPM, leading to reduced pressure levels. (Anggono et al., 2019)

Therefore, the observed decrease in peak cylinder pressure at elevated RPMs aligns with established engine performance characteristics.

3.3 New Connecting Rod Force Calculations

By applying the new pressure and engine speed data to the same code as before, we obtain the following result. This allows for the analysis of how the updated pressure values at higher RPMs impact the forces acting on the connecting rod.

New Rod Stress

```
Combined_PressureData = [PressureData, Pressure_new_Pa];
Gas_Force_new = Combined_PressureData .* Piston_Area;
Piston_Inertia_Force_Inv = Piston_Inertia_Force.';
New_Net_Force = Gas_Force_new - Piston_Inertia_Force_Inv;
New_Rod_Force = New_Net_Force ./ cosd(Rod_Angle');
New_Side_Force = New_Net_Force .* tand(Rod_Angle');
New_Side_Friction = 0.1 .* New_Side_Force;
New_Rod_Stress = New_Rod_Force ./ Rod_CSAmmsq;

% Find minimum (compressive) stress and its position
[NewCompressive_Stress, minIndex] = min(New_Rod_Stress(:));
[rowMin, colMin] = ind2sub(size(New_Rod_Stress), minIndex);

% Find maximum (tensile) stress and its position
[NewTensile_Stress, maxIndex] = max(New_Rod_Stress(:));
[rowMax, colMax] = ind2sub(size(New_Rod_Stress), maxIndex);

% Calculate stress range and mean stress
New_Stress_Range = NewTensile_Stress - NewCompressive_Stress;
New_Stress_Mean = (NewTensile_Stress + NewCompressive_Stress) / 2;

New_Stress_Allow = 2 .* EndLim .* (1 - (New_Stress_Mean ./ UTS));
New_FSF = New_Stress_Allow ./ New_Stress_Range;

% Display results
fprintf('9000 RPM Minimum Compressive Stress: %.2f at Row %d, Column %d\n', NewCompressive_Stress, rowMin, colMin);
fprintf('9000 RPM Maximum Tensile Stress: %.2f at Row %d, Column %d\n', NewTensile_Stress, rowMax, colMax);
fprintf('9000 RPM Stress Range: %.2f\n', New_Stress_Range);
fprintf('9000 RPM Mean Stress: %.2f\n', New_Stress_Mean);
fprintf('9000 RPM Factor of Safety (FSF): %.2f\n', New_FSF);
```

Figure 22 - New Rod Stress Calculations at 9000 RPM

```
9000 RPM Minimum Compressive Stress: -152.28 at Row 718, Column 9
9000 RPM Maximum Tensile Stress: 124.23 at Row 381, Column 7
9000 RPM Stress Range: 276.51
9000 RPM Mean Stress: -14.02
9000 RPM Factor of Safety (FSF): 2.49
```

Figure 23 - New Rod Stresses and Factor of Safety at 9000 RPM

As seen in Figure 23, the new safety factor of 2.49 compares relatively well with the original value of 2.90. However, there is an inconsistency in the results. The minimum compressive stress has decreased, even though it should have remained the same as before, since this occurred on Row 3 (3000 RPM). This discrepancy raises doubts about the accuracy of the code used, as the original data is still present and, therefore, the minimum compressive stress should have been identical. Further analysis is required to verify the integrity of the code and ensure that the results accurately reflect the expected behaviour.

3.4 Connecting Rod After Treatment

There are many methods of post-treating these parts to improve their ultimate tensile strength. The method chosen for this study was quenching and tempering of the connecting rods, as research suggests that this process could increase the ultimate tensile strength by as much as 50%. (Howeyze et al., 2021)

This heat treatment process involves heating the material to a high temperature and then rapidly cooling it (quenching), followed by reheating it to a lower temperature (tempering) to achieve a balance between hardness and ductility, ultimately enhancing the component's strength and fatigue resistance.

Quenching and Tempering Rods

```
Combined_PressureData = [PressureData, Pressure_new_Pa];
Gas_Force_new = Combined_PressureData .* Piston_Area;
Piston_Inertia_Force_Inv = Piston_Inertia_Force.^-1;
New_Net_Force = Gas_Force_new - Piston_Inertia_Force_Inv;
New_Rod_Force = New_Net_Force ./ cosd(Rod_Angle);
New_Side_Force = New_Net_Force .* tand(Rod_Angle);
New_Side_Friction = 0.1 .* New_Side_Force;
New_Rod_Stress = New_Rod_Force ./ Rod_CSAmmsq;

% Find minimum (compressive) stress and its position
[NewCompressive_Stress, minIndex] = min(New_Rod_Stress(:));
[rowMin, colMin] = ind2sub(size(New_Rod_Stress), minIndex);

% Find maximum (tensile) stress and its position
[NewTensile_Stress, maxIndex] = max(New_Rod_Stress(:));
[rowMax, colMax] = ind2sub(size(New_Rod_Stress), maxIndex);

% Calculate stress range and mean stress
New_Stress_Range = NewTensile_Stress - NewCompressive_Stress;
New_Stress_Mean = (NewTensile_Stress + NewCompressive_Stress) / 2;

New_UTS = UTS .* 1.5;
New_EndLim = 0.45 .* New_UTS;
New_Stress_Allow = 2 .* New_EndLim .* (1 - (New_Stress_Mean ./ New_UTS));
New_FSF = New_Stress_Allow ./ New_Stress_Range;

% Display results
fprintf('Quenching and Tempering Minimum Compressive Stress: %.2f at Row %d, Column %d\n', NewCompressive_Stress, rowMin, colMin);
fprintf('Quenching and Tempering Maximum Tensile Stress: %.2f at Row %d, Column %d\n', NewTensile_Stress, rowMax, colMax);
fprintf('Quenching and Tempering Stress Range: %.2f\n', New_Stress_Range);
fprintf('Quenching and Tempering Mean Stress: %.2f\n', New_Stress_Mean);
fprintf('Quenching and Tempering Factor of Safety (FSF): %.2f\n', New_FSF);
```

Figure 24 - Quenched and Tempered Rod Stress Calculations at 9000 RPM

```
Quenching and Tempering Minimum Compressive Stress: -152.28 at Row 718, Column 9
Quenching and Tempering Maximum Tensile Stress: 124.23 at Row 381, Column 7
Quenching and Tempering Stress Range: 276.51
Quenching and Tempering Mean Stress: -14.02
Quenching and Tempering Factor of Safety (FSF): 3.71
```

Figure 25 - Quenched and Tempered Rod Stress Results at 9000 RPM

Figure 25 shows the increase in the factor of safety to 3.71 after the quenching and tempering process. This improvement, which represents a 49% increase from 2.49 the 9000 RPM figure and a 28% increase on the original value of 2.90, demonstrates the significant impact of heat treatment on enhancing the material's strength and durability, leading to a higher margin of safety and better resistance to fatigue under operational conditions.

4 Gudgeon Pin Analysis

4.1 Original Stresses

In this section, the stresses at the gudgeon pin are calculated using a series of steps that involve determining the forces acting on the piston assembly and applying standard stress formulas. The goal is to assess the bending, torsional, and resultant stresses at the gudgeon pin, which are crucial for determining the safety and durability of the engine components.

Gudgeon Pin Stresses

```
% Step 1: Inertia Force Calculation
FPin_Inertia = Pist_Exposed_Mass .* Piston_Accel;
MPin_Inertia = FPin_Inertia .* L1;

FPin_Piston = Mass_Piston_Parts .* Piston_Accel;

% Reshape arrays to fit (6, 720) without if statements
FPin_Piston_Resized = reshape(FPin_Piston(1:(6*720)), 6, 720);
MPin_Inertia_Resized = reshape(MPin_Inertia(1:(6*720)), 6, 720);

F_Net = FPin_Piston_Resized - Gas_Force;

% Step 2: Net Moment Calculation
MNet_Force = (F_Net ./ 2) .* L2;
MPin_Inertia_Resized = reshape(MPin_Inertia(1:(6*720)), 6, 720);
M = MNet_Force + MPin_Inertia_Resized;

% Step 3: Bending Stress Calculation
Stress_Bending = (M .* L1) ./ ((pi/64) .* (D2^4 - D1^4));

% Step 4: Torsional Stress Calculation
T = ((F_Net ./ 2) + FPin_Inertia_Resized) ./ ((pi/4) .* (D2^3 - D1^3));

% Step 5: Resultant Stress Calculation
Pin_Result_Stress = sqrt((Stress_Bending).^2 + (2 .* T).^2);

% Flip sign if both bending and torsional stress are negative
idx = (Stress_Bending < 0) & (T < 0);
Pin_Result_Stress(idx) = Pin_Result_Stress(idx) * -1;
Pin_Result_Stress_Correct = Pin_Result_Stress;

% Step 6: Find Extremes
value = Pin_Result_Stress_Correct(6, 390);

[Pin_Stress_Max, linearIndexMax] = max(Pin_Result_Stress_Correct(:));
[rowMax, colMax] = ind2sub(size(Pin_Result_Stress_Correct), linearIndexMax);

[Pin_Stress_Min, linearIndexMin] = min(Pin_Result_Stress_Correct(:));
[rowMin, colMin] = ind2sub(size(Pin_Result_Stress_Correct), linearIndexMin);

% Step 7: Calculate Allowable Stress and FSF
Pin_UTS = 1000;
Pin_EndLim = 0.45 * Pin_UTS;

Pin_Stress_Mean = (Pin_Stress_Max + Pin_Stress_Min) / 2;
Pin_Stress_Range = Pin_Stress_Max - Pin_Stress_Min;

Pin_Stress_Allow = 2 * Pin_EndLim * (1 - (Pin_Stress_Mean / Pin_UTS));
Pin_FSF = Pin_Stress_Allow / Pin_Stress_Range;

% Display Results
fprintf('Value at Row 6, Column 390: %.2f\n', value);
fprintf('Maximum Stress: %.2f at Row %d, Column %d\n', Pin_Stress_Max, rowMax, colMax);
fprintf('Minimum Stress: %.2f at Row %d, Column %d\n', Pin_Stress_Min, rowMin, colMin);
fprintf('Pin Stress Allowable: %.2f\n', Pin_Stress_Allow);
fprintf('Factor of Safety (FSF): %.2f\n', Pin_FSF);
```

Figure 26 - Code for Original Gudgeon Pin Stresses

```
Value at Row 6, Column 390: -37252341.56
Maximum Stress: 10280059.48 at Row 3, Column 2
Minimum Stress: -56399774.64 at Row 6, Column 324
Pin Stress Allowable: -20879228.18
Factor of Safety (FSF): -0.13
```

Figure 27 - Original Stresses on Gudgeon Pin Results

The results obtained from the stress analysis indicate significant issues, with a negative factor of safety (FSF) of -0.13 and an allowable stress that is also negative. These values suggest that the gudgeon pin

would fail under operating conditions, which is inconsistent with expected engineering tolerances for a well-designed component.

Given the magnitude and direction of the stresses, it is evident that there is an error in the code used in the calculations which was not tracked during this simulation.

4.2 New Connecting Gudgeon Pin Stresses

Gudgeon Pin Stresses at 9000 RPM

```
% Step 1: Inertia Force Calculation
FPin_Inertia9K = Pist_Exposed_Mass .* Piston_Accel;
MPin_Inertia9K = FPin_Inertia9K .* L1;

FPin_Piston9K = Mass_Piston_Parts .* Piston_Accel;

F_Net9K = FPin_Piston9K - Gas_Force_new;

% Net moment calculation
MNet_Force9K = (F_Net9K ./ 2) .* L2;
M9K = MNet_Force9K + MPin_Inertia9K;

% Step 2: Bending Stress Calculation
Stress_Bending9K = (M9K .* L1) ./ ((pi/64) .* (D2^4 - D1^4));

% Step 3: Torsional Stress Calculation
T9K = ((F_Net9K ./ 2) + FPin_Inertia9K) ./ ((pi/4) .* (D2^2 - D1^2));

% Step 4: Resultant Stress Calculation
Pin_Result_Stress9K = sqrt((Stress_Bending9K.^2 + (2 .* T9K).^2));

% Step 5: Stress Extremes and Positions
value9K = Pin_Result_Stress9K(6, 390);

[Pin_Stress_Max9K, linearIndexMax9K] = max(Pin_Result_Stress9K(:));
[rowMax9K, colMax9K] = ind2sub(size(Pin_Result_Stress9K), linearIndexMax9K);

[Pin_Stress_Min9K, linearIndexMin9K] = min(Pin_Result_Stress9K(:));
[rowMin9K, colMin9K] = ind2sub(size(Pin_Result_Stress9K), linearIndexMin9K);

% Step 6: Calculate Allowable Stress and FSF
Pin_Stress_Mean9K = (Pin_Stress_Max9K + Pin_Stress_Min9K) / 2;
Pin_Stress_Range9K = Pin_Stress_Max9K - Pin_Stress_Min9K;

Pin_Stress_Allow9K = 2 .* Pin_EndLim .* (1 - (Pin_Stress_Mean9K ./ Pin_UTS));
Pin_FSF9K = Pin_Stress_Allow9K ./ Pin_Stress_Range9K;

fprintf('Value at Row 6, Column 390: %.2f\n', value9K);
fprintf('Maximum Stress: %.2f at Row %d, Column %d\n', Pin_Stress_Max9K, rowMax9K, colMax9K);
fprintf('Minimum Stress: %.2f at Row %d, Column %d\n', Pin_Stress_Min9K, rowMin9K, colMin9K);
fprintf('Pin Stress Allowable: %.2f\n', Pin_Stress_Allow9K);
fprintf('Factor of Safety (FSF): %.2f\n', Pin_FSF9K);
```

Figure 28 - Gudgeon Pin Stresses Code for 9000 RPM

This code calculates the stresses on the gudgeon pin at 9000 RPM, incorporating both bending and torsional stresses to determine the resultant stress. However, if there are inconsistencies or unrealistic values in these inputs, the results may lead to incorrect or non-physical outputs, as seen in the previous analysis.

```
Value at Row 6, Column 390: 36289967.24
Maximum Stress: 182633658.96 at Row 7, Column 381
Minimum Stress: 4134.34 at Row 1, Column 76
Pin Stress Allowable: -46186183.39
Factor of Safety (FSF): -0.45
```

Figure 29 - Original Stresses on Gudgeon Pin Results at 9000 RPM

The values in the stress analysis are significantly high, with the maximum stress reaching 102,633,650.96 N/mm² and the allowable stress remaining negative. These elevated values suggest that the inputs or calculations might be off in a way that leads to unrealistic results. For example, the force values, mass parameters, or acceleration data could be incorrectly scaled or interpreted, leading to unreasonably high stress values. While the FSF has reduced slightly, it is still negative, reinforcing the idea that something is wrong upstream. These unusually high stress values indicate that the underlying assumptions or input data need to be carefully reviewed and corrected to ensure realistic and accurate results.

4.3 Gudgeon Pin After Treatment

The simulation of material treatments, such as nitriding, was not included for the gudgeon pin, as the upstream calculations currently yield unrealistic results. The stress values are excessively high, and the factor of safety remains negative, suggesting potential errors or inconsistencies in the input data or assumptions. Until these fundamental issues are resolved, simulating a material treatment would not provide reliable or meaningful results, as the baseline calculations themselves are not yet accurate.

5 Bolt Analysis

5.1 Original Stresses

The following code was used to calculate the stresses acting on the rod cap bolts.

Rod Cap

```
Rot_Mass = Piston_Mass + Rod_Mass + Ring_Mass + Pin_Mass + Clips_Mass + Cap_Mass + 2.* Bolt_Mass;

FWhip = Rot_Mass .* Crank_Rad .* Crank_Ang_Vel.^2 .* cosd(Crank_Angle);

ForceInertia = (Recip_Mass .* Crank_Rad .* (Crank_Ang_Vel.^2) .* (cosd(Crank_Angle)) + ...
    (((Crank_Rad^2) ./ Rod_Length) .* (Crank_Ang_Vel.^2) .* cosd(2 .* Crank_Angle)));

FTotalCap = ForceInertia + FWhip;

FTotalCap_MaxValue = max(FTotalCap(:));

MaxStressCap = FTotalCap_MaxValue ./ Cap_Int_CSAmmsq;

CapStressMean = (0 + MaxStressCap)/2;

CapStressRange = MaxStressCap - 0;

CapStressAllowable = 2 * EndLim * (1 - (CapStressMean / UTS));

Cap_FSF = CapStressAllowable / CapStressRange;

% Display the results
fprintf('Maximum Stress in Cap: %.2f\n', MaxStressCap);
fprintf('Mean Stress in Cap: %.2f\n', CapStressMean);
fprintf('Stress Range in Cap: %.2f\n', CapStressRange);
fprintf('Factor of Safety (FSF) for Cap: %.2f\n', Cap_FSF);
```

Figure 30 - Rod Cap Stress Calculation Code

```
Maximum Stress in Cap: 185.71
Mean Stress in Cap: 92.85
Stress Range in Cap: 185.71
Factor of Safety (FSF) for Cap: 3.18
```

Figure 31 - Rod Cap Stress Results

Surprisingly, the results for this are much more in line with what is expected, unlike the gudgeon pin results. The maximum and mean stresses appear reasonable for a rod cap under typical engine loading conditions. The stress range is consistent with anticipated cyclic loading, and the factor of safety (FSF) of 3.18 indicates a comfortable margin against failure. These results suggest that the rod cap is structurally sound and not over-stressed during operation.

Bolt

```

Bolt_UTS = 1100;

Yield_Bolt = 0.9 .* Bolt_UTS;

EndLim_Bolt = 0.45 .* Bolt_UTS;

Bolt_Axial_Stress = 0.6 .* Yield_Bolt;

Bolt_Axial_Force = Bolt_Axial_Stress .* Bolt_CSAmmsq;

Effect_Radius = 0.5 .* ((8 + Head_Dia)/2);

Head_Friction_Torque = (0.12 .* Bolt_Axial_Force .* Effect_Radius) / 1000;

Thread_Friction = 0.18 .* Bolt_Axial_Force;

Torque_Friction_Thread = (Thread_Friction .* 4) ./ 1000;

Total_Torque = Torque_Friction_Thread + Head_Friction_Torque;

Bolt_InertiaForce = Recip_Mass .* (Crank_Rad .* (Crank_Ang_Vel.^2) .* cosd(Crank_Angle)) + ...
    (Rotat_Mass .* (Crank_Rad .* (Crank_Ang_Vel.^2) .* cosd(2 .* Crank_Angle)));

Bolt_InertiaForce_perBolt = Bolt_InertiaForce ./ 2;

Cap_Stiffness = Mod_Elas_Cap .* (Cap_Int_CSAmmsq ./ Rod_Int_Bolt);

Bolt_Stiffness = Mod_Elas_Bolt .* (Bolt_CSAmmsq ./ Lengh_Sec_Thread);

Extra_BoltForce = (Bolt_InertiaForce_perBolt .* Bolt_Stiffness) ./ (Bolt_Stiffness + Cap_Stiffness);

[Max_Extra_BoltForce, idx] = max(Extra_BoltForce(:));
[row, col] = ind2sub(size(Extra_BoltForce), idx);

NewBoltForce = Max_Extra_BoltForce + Bolt_Axial_Force;

BoltStress_Min = Bolt_Axial_Force ./ Bolt_CSAmmsq;

BoltStress_Max = NewBoltForce ./ Bolt_CSAmmsq;

BoltStress_Mean = (BoltStress_Max + BoltStress_Min) / 2;

BoltStress_Range = BoltStress_Max - BoltStress_Min;

BoltStress_Allow = 2 .* EndLim_Bolt .* (1 - (BoltStress_Mean ./ Bolt_UTS));

BoltStress_FSF = BoltStress_Allow ./ BoltStress_Range;

% Display the results
fprintf('Total Bolt Torque: %.2f Nm\n', Total_Torque);
fprintf('Maximum Stress in Bolt: %.2f\n', BoltStress_Max);
fprintf('Mean Stress in Bolt: %.2f\n', BoltStress_Mean);
fprintf('Stress Range in Bolt: %.2f\n', BoltStress_Range);
fprintf('Factor of Safety (FSF) for Bolt: %.2f\n', BoltStress_FSF);

```

Figure 32 - Bolt Stresses Code

```

Total Bolt Torque: 33.95 Nm
Maximum Stress in Bolt: 655.50
Mean Stress in Bolt: 624.75
Stress Range in Bolt: 61.50
Factor of Safety (FSF) for Bolt: 6.96

```

Figure 33 - Bolt Stress Results

The results for the conrod bolts demonstrate a notably high factor of safety (FSF) of 6.96, which is significantly greater than the FSF observed for other components like the connecting rod and the cap. This high FSF suggests that the conrod bolts are subjected to considerably lower stress levels relative to their designed capacity, indicating a conservative design with a substantial safety margin. The maximum stress of 655.50 MPa and the mean stress of 624.75 MPa are within acceptable limits for high-strength fasteners, reflecting that the bolts are not operating near their material limits. The higher FSF could be attributed to the critical nature of conrod bolts in maintaining engine integrity under dynamic loading conditions, where failure could lead to catastrophic engine damage. Therefore, the bolts are likely overengineered to ensure long-term durability and to accommodate potential variations in load and operating conditions.

6 Determine Optimal Crankshaft Configuration

To analyse the optimal crankshaft configuration, the first step is to calculate the crank torque behaviour. Crank torque is a critical parameter that influences the engine's overall performance, including power output, rotational stability, and efficiency. It is influenced by several factors such as crank radius, angular velocity, piston mass, and gas forces. Figure 34 and Figure 35 outline the approach to calculating the crank torque behaviour.

Torque

```
% Calculate Inertial Torque
Torque_Inertia = (ForceInertia .* tand(Rod_Angle) .* Crank_Rad .* cosd(Rod_Angle)) ...
    - (ForceInertia .* Crank_Rad .* sind(Rod_Angle));

% Calculate Gas Torque
Torque_Gas = (Gas_Force_new' .* tand(Rod_Angle) .* Crank_Rad .* cosd(Rod_Angle)) ...
    + (Gas_Force_new' .* Crank_Rad .* sind(Rod_Angle));

% Total Crank Torque
Crank_Torque = Torque_Gas + Torque_Inertia;

% Average Torque per Cycle (assuming 720 degrees per cycle)
Torque_Avg = sum(Crank_Torque') / 720;

% Total Engine Torque (assuming a 4-cylinder engine)
Engine_Torque = Torque_Avg * 4;

% Calculate Ring Friction Forces
Ring_Force = Combined_PressureData' .* pi .* Bore_Dia .* 1.5;
Ring_Friction = Ring_Force * 0.1; % Assuming a friction coefficient of 0.1

% External Gas Forces
Gas_Force_Ext = Combined_PressureData .* Piston_Area;

% Net External Forces
Net_Force_Ext = Gas_Force_Ext' - ForceInertia;

% Rod Forces
Rod_Force_Ext = Net_Force_Ext ./ cosd(Rod_Angle);

% Side Forces
Side_Force_Ext = Net_Force_Ext .* tand(Rod_Angle);

% Side Friction Forces
Side_Friction_Ext = 0.1 * Side_Force_Ext; % Assuming a friction coefficient of 0.1
```

Figure 34 - Torque Calculation Code Part 1

```
% Work Done by Ring Friction
Stroke_Length = max(Dist_From_TDC) - min(Dist_From_TDC);
Work_RingFriction = mean(Ring_Friction, 2) * Stroke_Length;

% Work Done by Side Friction
Work_SideFriction = mean(Side_Friction_Ext, 2) * Stroke_Length;

% Total Friction Work Losses
Ring_FrictionTotal = sum(Work_RingFriction);
Work_Loss_Total = Ring_FrictionTotal + Work_SideFriction;

% Work per Torque
Work_Torque = Crank_Torque(:, 720);

% Extract RPM Values (assuming RPM_Grid is a column vector)
RPM_Values1 = RPM_Grid(:, 1);

% Ensure Work_Torque is a row vector
Work_Torque1 = Work_Torque(:)';

% Calculate Brake Torque
Work_Loss_Total = Ring_FrictionTotal + Work_SideFriction; % Keep this part intact
Brake_Torque = max(0, Work_Torque1' - Work_Loss_Total); % Ensure no negative brake torque

% Display intermediate results for debugging
disp('Work_Torque at each RPM:');
disp(Work_Torque1');
disp('Work_Loss_Total at each RPM:');
disp(Work_Loss_Total);

% Calculate Powers
Indicated_Power = Work_Torque1' .* Crank_Ang_Vel; % Indicated Power
Friction_Power = Work_Loss_Total .* Crank_Ang_Vel; % Friction Power
Brake_Power = Indicated_Power - Friction_Power; % Brake Power
```

Figure 35 - Torque Calculation Code Part 2

Engine Torque at various RPMs:

RPM_Values1	Var2
1000	0.00061681
2000	0.00085458
3000	0.00086025
4000	0.00087936
5000	0.00089261
6000	0.0008408
7000	78.899
8000	73.718
9000	68.538

Brake Power at various RPMs:

RPM_Values1	Brake_Power
1000	-6.8417e+05
2000	-1.3683e+06
3000	-2.0525e+06
4000	-2.7367e+06
5000	-3.4209e+06
6000	-4.105e+06
7000	-4.7909e+06
8000	-5.4752e+06
9000	-6.1595e+06

The code used for calculating the crank torque did not work properly, with torque values peaking below 70 Nm, which is much lower than expected. Additionally, in the 1000 to 6000 RPM range, the torque values were close to zero, as seen in Figure 35. This suggests that the torque behaviour in this RPM range was not accurately captured, likely due to errors in force or moment calculations. The issue could stem from incorrect data inputs or mistakes in the torque calculation formula, leading to unrealistic results.

Due to the inability to empirically analyse the crank torque behaviour, it was not possible to fully assess the differences between the flat-plane and cross-plane crankshaft configurations.

Flat-plane crankshafts are often favoured in high-performance engines due to their lighter weight and reduced rotational inertia, which facilitate quicker engine acceleration and higher RPM capabilities. Their design also enhances exhaust scavenging efficiency, contributing to improved engine performance at elevated speeds.

However, flat-plane crankshafts can introduce increased vibrations due to secondary imbalances, necessitating additional engineering solutions to mitigate these effects.

In contrast, cross-plane crankshafts are associated with smoother operation and higher torque at lower RPMs, but their increased rotational mass can limit high-speed performance. (Edtengineers.com, 2022)

For high-RPM applications such as this engine with the raised 9000 RPM, flat-plane crankshafts are generally more suitable.

7 References

Anggono, W., Ikoma, W., Chen, H., Liu, Z., Ichiyanagi, M. and Suzuki, T. (2019). Effect of Various Supercharger Boost Pressure to in-Cylinder Pressure and Heat Release Rate Characteristics of Direct Injection Diesel Engine at Various Engine Rotation. *E3S Web of Conferences*, [online] p.01036. doi: <https://doi.org/10.1051/e3sconf/201913001036>

Edtengineers.com. (2022). *EDT Engineers*. [online] Available at: <https://www.edtengineers.com/blog-post/gearhead-quandary-be-flat-or-be-crossed>.

Ford Motor Company (no date) *1.6 GTDI EcoBoost Datasheet*. <https://www.ford.co.uk/content/dam/guxeu/uk/documents/home/experience-ford/about-ford/ford-component-sales/specification-details-data-sheets/16-ecoboost-np.pdf> (Accessed: March 13, 2025)

Howeyze, M., Eivani, A.R., Ghosh, M. and Jafarian, H.R. (2021). Effects of annealing temperature on microstructure and mechanical behavior of conventionally and severely deformed Fe–24Ni steel by accumulative roll bonding. *Journal of Materials Research and Technology*, [online] pp.2428–2440. doi: <https://doi.org/10.1016/j.jmrt.2021.07.108>

Hugo, A., Pedreira, J.A., Kurka, P.R.G. and Bittencourt, M.L. (2017). *Modeling and Simulation of Torsional Vibrations in Two-Stage Reciprocating Compressors Driven by Electrical Motor*. [online] Available at: https://www.researchgate.net/publication/324953418_Modeling_and_Simulation_of_Torsional_Vibrations_in_Two-Stage_Reciprocating_Compressors_Driven_by_Electrical_Motor