# MME540 Graduate Project

Kyler Limata

May 8th, 2025

## Contents



## Introduction

This report covers the design of a piston connecting rod for an Otto cycle engine, including the stress analysis, material selection, and finite element analysis. The piston as a whole has a bore of 80 mm, a stroke of 96 mm a connecting rod length of 200 mm, a compression ratio of 10, a pin diameter of 45 mm, and a bearing diameter of 70 mm.

## 1 Design and Parameters Overview

The following design parameters are introduced for the design process:

$d_{out}$: The outside diameter of the ring around the pin

$w_{beam}$: The width of the central beam section

$t_{beam}$: The thickness of the central beam section

$w_{web}$: The width of the beam web

$t_{web}$: The thickness of the beam web

$w_{base}$: The width of the base

$r_{basefillet}$: The radius of the fillet where the beam meets the base

$r_{webfillet}$: The radius of the fillet inside the slot formed by the web

One thing to note is that the base is perfectly square, including the bottom part omitted from the design, so the distance from the center of the bearing to the top of the base is half the width. Additionally, the slot formed by the web spans from the edge of the base fillet to $d_{out} + 15mm$ below the center of the pin.

# 2 Stress Analysis

As shown in the following figure, stress analysis will be performed at four different points; three near the base of the piston beam section and one at the side of the rings.

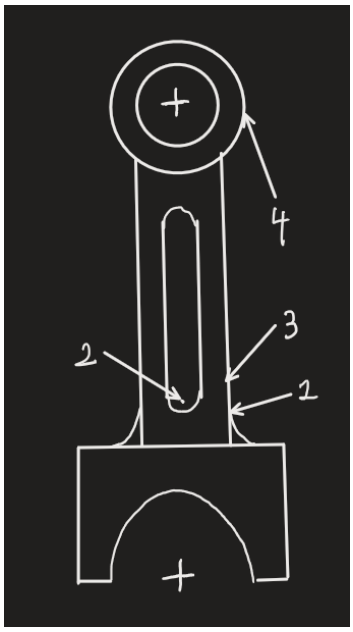



Figure 1: Four stress analysis points

Only the x and y directions will be considered in this stress analysis as loading in the z direction should be negligable.

## 2.1 Stress Equations and Assumptions

At point one, there is a stress concentration due to the fillet connecting the base to the "beam" section of the rod. We can treat this as a bar with a shoulder fillet, giving a stress concentration factor for the bending stress in x and normal stress in y, giving us the following equations:

$$\begin{aligned} M_1 &= F_x(r_{rod} - 0.5w_{base} \\ \sigma_x &= k_{t,bending,1}\frac{6M_1}{t_{beam}w_{beam}^2} \\ \sigma_y &= k_{t,axial,1}\frac{Fy}{w_{beam}t_{beam}} \\ \tau_{xy} &= \frac{3}{2}\frac{F_x}{w_{beam}t_{web}} \end{aligned} \tag{1}$$

Point two is a little more complicated; it can be approximated as a bar with a shoulder fillet, but the catch is that if the bar is the web of the "beam" section, then the small bar cross-section does not make up the entire cross-section of the beam. After some experimentation, it was found that multiplying the load by the percentage of the total cross-section the web cross-section takes up in addition to the stress concentration factor produces results close to the FEA results later shown

in **Section 4**. However, since the end of the slot is round and not square, this approximation isn't entirely accurate, so it was decided to analyze at the center of the beam to avoid inaccuracies in bending stresses. With all that, the equations for point 2 are as follows.

$$\begin{aligned} A_{web} &= w_{web}t_{web} \\ A_{total} &= w_{beam}t_{beam} - w_{web}(t_{beam} - t_{web}) \\ \eta &= \frac{A_{web}}{A_{total}} \\ \sigma_x &= 0 \\ \sigma_y &= k_{t,axial,2}\frac{\eta F y}{w_{web}t_{web}} \\ \tau_{xy} &= \frac{3}{2}\frac{F_x}{w_{beam}t_{web}} \end{aligned} \tag{2}$$

Originally, it was attempted to find a stress concentration factor for point three by treating it as a combined plate with hole and solid cross-section, using a similar approximation to point 2. However, it was found that using no stress concentration factor produced results closest to the FEA. The equations for point 3 are shown below:

$$\begin{aligned} A_{webfillet} &= (1 - \frac{\pi}{4})r_{basefillet}^2 \\ A_{cross,3} &= w_{beam}t_{beam} - w_{web}(t_{beam} - t_{web}) + 4A_{webfillet} \\ M_3 &= F_x(r_{rod} - 0.5w_{base} - r_{basefillet} - 0.5w_{web}) \\ I_3 &= \frac{w_{web}t_{web}^3}{12} + \frac{t_{beam}^3}{12}(w_{beam} - w_{web}) \\ \sigma_x &= \frac{M_3 w_{beam}}{2I_3} \\ \sigma_y &= \frac{Fy}{A_{cross,3}} \\ \tau_{xy} &= \frac{3}{2}\frac{F_x}{A_{cross,3}} \end{aligned} \tag{3}$$

For point four, it was chosen to approximate the top of the rod as an infinite plate with a hole. Despite it actually being a ring shape, the approximation seems to have held up during FEA. Since we're at the top of the rod, there is no bending stress here. The equations for point four are shown below.

$$\begin{aligned} A_{cross,4} &= t_{beam}(d_{out} - d_{pin}) \\ \sigma_x &= 0 \\ \sigma_y &= \frac{Fy}{A_{cross,4}} \\ \tau_{xy} &= \frac{3}{2}\frac{F_x}{A_{cross,4}} \end{aligned} \tag{4}$$

Since we're only looking at stresses in the two-dimensional case, we can use the following equation to find the maximum principal stresses:

$$\sigma_1, \sigma_2 = \frac{\sigma_x + \sigma_y}{2} \pm \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2} \tag{5}$$

Since we only need $\sigma_1$, we can resolve to the case where the square root is added.

## 2.2 Numerical Simulation

In the interest of performing a thorough fatigue analysis with a non-zero mean, a numerical simulation of the stress equations for each point was developed. This simulation calculated the loading conditions over the compression and power strokes of the ideal auto cycle - the intake and exhaust strokes were ignored as the stresses during those strokes would be minimal but reduce the mean stress at each point by a non-negligible amount. The full source code for the simulation and calculations can be found in **Appendix A**.

For the simulation, the design parameters below were chosen - all values are in millimeters.

Table 1: Design Parameters

| $d_{out}$ | $w_{beam}$ | $t_{beam}$ | $w_{web}$ | $t_{web}$ | $w_{base}$ | $r_{basefillet}$ | $r_{webfillet}$ |
|---|---|---|---|---|---|---|---|
| 60 | 50 | 30 | 30 | 15 | 100 | 10 | 4.5 |

With these parameters and the design requirements, the stress concentration factors for points 1 and 2 could be found using the charts in chapter 4.12 of *Fundamentals of Machine Component Design* (Juvinall et al.) [2].

Table 2: Stress concentration factors

| Point | 1 | 2 |
|---|---|---|
| $k_{t,axial}$ | 1.8 | 1.65 |
| $k_{t,bending}$ | 1.5 | None |

The results of the simulation are shown below.

Python was also used to compute the maximum and average principal stress at each point; these values are shown in the table below.

Table 3: Maximum and average stresses

| Point | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $\sigma_m$ (Mpa) | 46.682 | 2.679 | 258.066 | 6.618 |
| $\sigma_{max}$ (Mpa) | 230.271 | 11.377 | 1277.819 | 28.227 |

## 2.3 Fatigue Analysis and Safety Factor

Now that the max and average principal stresses are known, a fatigue analysis can be conducted at each point to find the safety factor. Let's start by assuming the rod will be made from grade 80-55-06 grey nodular cast iron, which has a tensile strength of $s_u = 974\,Mpa$ and a yield strength of $s_y = 864\,Mpa$. Chapter 5-6 of *Machine Elements in Mechanical Design* (Mott et al.) has a chart in Figure 5-11 for finding the endurance limit as a function of the tensile strength [1]. Assuming a machined surface, the endurance limit for grade 80-55-06 nodular iron is $s_n = 350\,Mpa$. From there, we can use the book's equation for estimating the actual endurance limit:

$$s_n^{'} = s_n(C_m)(C_{st})(C_R)(C_s) \tag{6}$$

)

Where the C factors are the material factor, stress factor, reliability factor, and the size factor. Per the book's recommendation, we will choose $C_m = 0.66$ for ductile (nodular) iron, $C_{st} = 0.8$ for axial tension, and $C_R = 0.75$ for 99.9% reliability. Size factor is chosen based on the diameter of the cross-sectional area; since none of the cross sections are circular, we will instead use the equivalent diameter, or the diameter of a circle having the same area as the cross-section. After $s_n^{'}$ is found at each point, the safety factor can be calculated using the Soderberg Criterion:

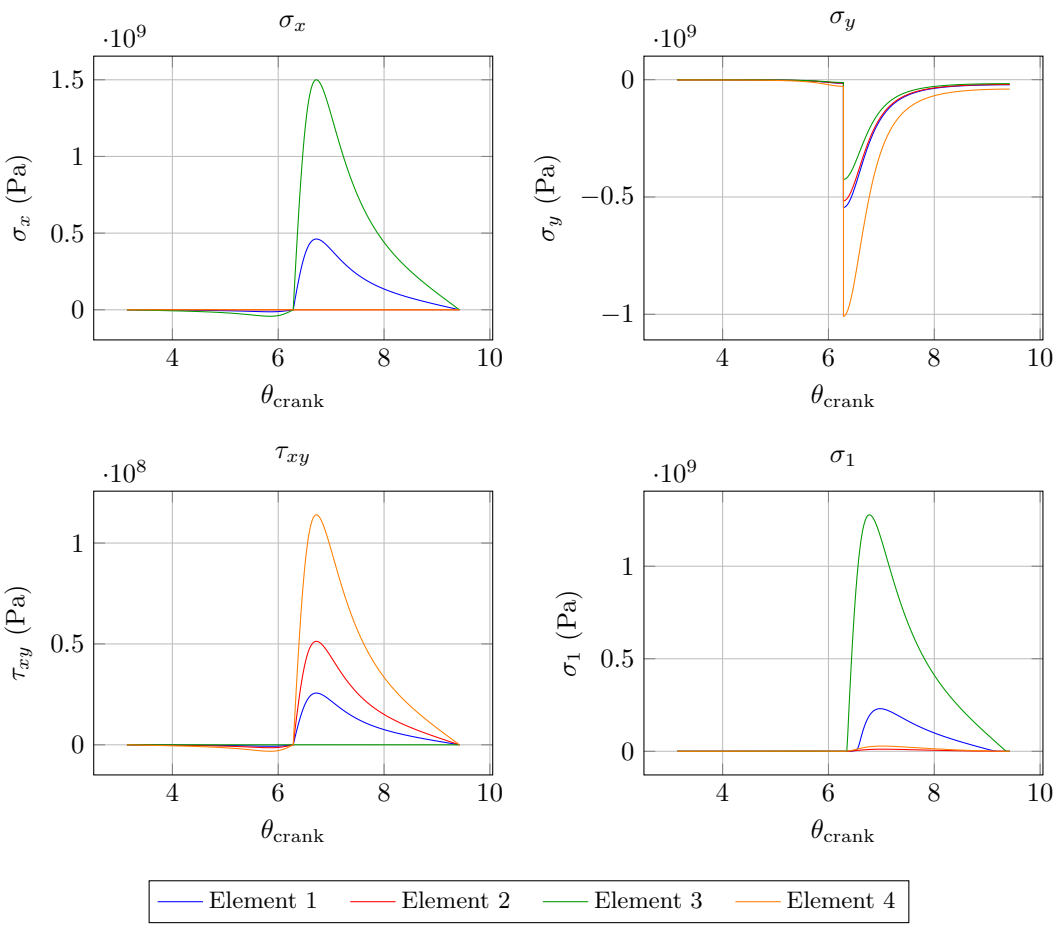



Figure 2: Stress components vs. Crank Angle for four elements

$$\frac{k_t(\sigma_{max} - \sigma_m)}{s_n^{'}} + \frac{\sigma_m}{s_u} = \frac{1}{SF}$$
$$SF = \frac{1}{\frac{k_t(\sigma_{max} - \sigma_m)}{s_n^{'}} + \frac{\sigma_m}{s_u}} \tag{7}$$

The Soderberg criterion will give a more conservative estimate, which is good especially since the loading cycle does not follow a sinusoidal pattern. With all that, the results of the calculations can be found in the table below.

Table 4: Maximum and average stresses

| Point | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $D_{eq}\,(mm)$ | 43 | 23.9 | 36.6 | 23.9 |
| $C_s$ | 0.81 | 0.87 | 0.83 | 0.87 |
| $s_n^{'}\,(Mpa)$ | 112.8 | 120.5 | 115 | 120.5 |
| Safety Factor | 0.59 | 0.63 | 0.61 | 0.63 |

# 3 FEA Validation

In an attempt to verify the accuracy of the stress equations show in **Section 2.1**, finite element analysis was performed using the ANSYS software. Due to difficulties with the transient structural analysis, static structural analysis was performed at $\theta_{crank} = 6.377907$ - shortly after the start of the power stroke. In the table below are the stress values calculated by the numerical simulation in Pascals. This is followed by ANSYS views of each type of stress with annotations added at the four nodes closest to the four points analyzed.

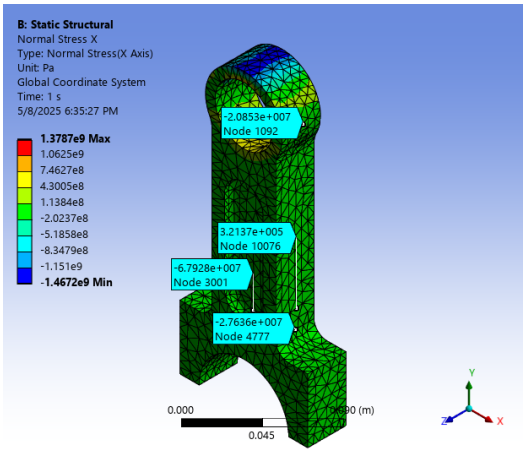
For the normal stresses in the y direction, the approximations hold up really well, especially at point 2. Where they sort of fall apart is with the x and shear stresses, which means that the normal stresses are off as well. This is likely due to how bulky the connecting rod is, allowing those "force lines" to spread apart as they move through the piston, which is not at all reflected in how the stress analysis treats the force as a point load rather than a distributed load. In a more slender rod, those approximations would likely be a lot more accurate.

# 4 Materials Selection and Manufacturing Process

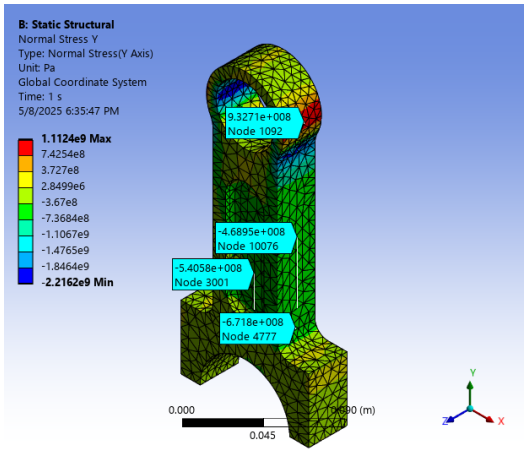
As previously discussed in **Section 2.3**, the connecting rod will be made from grade 80-55-06 gray nodular cast iron. Due to the concavity of the design, the casting will be done with a mold - sand casting will be used here since the surface finish doesn't need to be perfect. The pin hole will be absent from the mold. After casting, the surface that touches the bearing and the pin hole will be machined via milling, producing a high-quality surface finish.

| Stress | $\sigma_x$ | $\sigma_y$ | $\tau_{xy}$ | $\sigma_1$ |
|---|---|---|---|---|
| Point 1 | $1.79 \cdot 10^8$ | $-5.3 \cdot 10^{-8}$ | $9.95 \cdot 10^6$ | $2.85 \cdot 10^5$ |
| Point 2 | 0 | $-5 \cdot 10^{-8}$ | $1.99 \cdot 10^7$ | $7.9 \cdot 10^5$ |
| Point 3 | $5.82 \cdot 10^8$ | $-4.1 \cdot 10^{-8}$ | $4.48 \cdot 10^3$ | $1.71 \cdot 10^8$ |
| Point 4 | 0 | $-9.7 \cdot 10^{-8}$ | $4.42 \cdot 10^7$ | $2 \cdot 10^6$ |

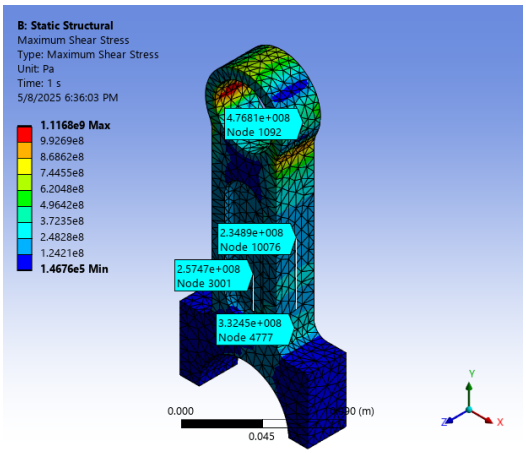
Table 5: Caption



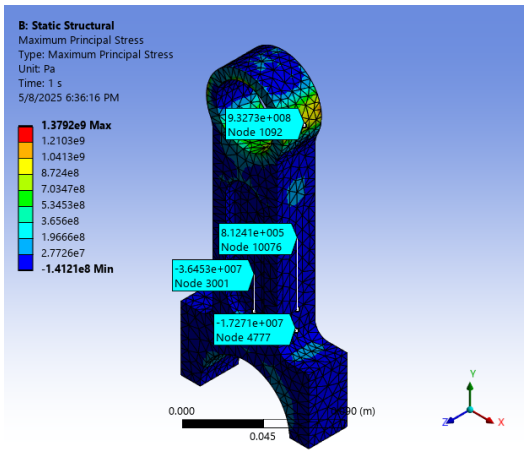

(a) Normal $x$ stress




(b) Normal $y$ stress




(c) Shear stress $\tau_{xy}$




(d) Principal stress $\sigma_1$

Figure 3: Finite Element Analysis Results for Various Stress Components

# A Python Source Code

Below is the source code used to perform the stress calculations at each of the four points.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

def piston_kinematics(B, S, r, theta_crank):
    """
    Calculates the displacement volume and connecting
    rod angle for the given piston geometry at the
    passed values of theta.

    Parameters
    ----------
    B : Piston Bore
    S : Piston Stroke
    r : Connecting Rod Length
    theta_crank : The crank shaft angles

    Returns
    -------

    """
    ## Find the displacement volume
    a = S/2 # Crank Offset
    s = a*np.cos(theta_crank) + np.sqrt(r**2 - (a**2)*(np.sin(theta_crank)**2)) #
        Displacement
    Vd = (r + a - s)*(np.pi/4)*(B**2) # Displacement Volume

    ## Find the angle of the connecting rod from the y-axis
    inside = (s**2 + r**2 - a**2)/(2*s*r)
    inside_clipped = np.clip(inside, -1.0, 1.0)
    theta_rod = np.arccos(inside_clipped)

    return {'Vd': Vd, 'theta_rod': theta_rod }

def simulate_connecting_rod(params, funcs, npoints):
    """
    Performs a numerical simulation of the loading on a
    piston connecting rod over the compression and
    power strokes of the ideal otto cycle.

    Uses the passed functions to find the normal and
    shear stresses at any given points and computes
    the maximum principle stress at each and the
    average of the maximum principle stresses.

    2D stress conditions are assumed.

    Parameters
    ----------
    params: Piston Parameters
    funcs : A list of functions for the normal and shear stresses at each point
    npoints : Number of points to simulate at for each stroke

    Returns
    -------
    Dictionary : Computed principle stresses
    """

    results = {} # Outpput data

    ## Unpack parameters
    B = params['B'] # Bore
    S = params['S'] # Stroke
    r = params['r'] # Connecting rod length
```

```
    CR = params['CR'] # Compression Ratio
    T1 = params['T1'] # Temperature at 1
    P1 = params['P1'] # Pressure at 1
    T3 = params['T3'] # Temperature at 3
    Vc = (1/(CR - 1))*(np.pi*S*B**2)/4 # Clearance volume

    ## Calculate kinematics for compression and power stroke
    theta_crank_compression = np.linspace(np.pi, 2*np.pi, npoints)
    kinematics_data_compression = piston_kinematics(B, S, r, theta_crank_compression)

    theta_crank_power = np.linspace(2*np.pi, 3*np.pi, npoints)
    kinematics_data_power = piston_kinematics(B, S, r, theta_crank_power)

    ## Compute P, V, and T at all four points
    k = 1.4 # Specific heat ratio
    V1 = Vc + (np.pi*S*B**2)/4 # Volume at 1
    V2 = Vc # Volume at 2
    V3 = V2 # Volume at 3
    V4 = V1 # Volume at 4
    T2 = T1*CR**(k - 1) # Temperature at 2
    P2 = P1*CR**k
    P3 = P2*(T3/T2)
    T4 = T3/(CR**(k - 1)) # Temperature at 4
    P4 = (P1*T4)/T1 # Pressure at 4
    results['V1'] = V1
    results['V2'] = V2
    results['V3'] = V3
    results['V4'] = V4
    results['P1'] = P1
    results['P2'] = P2
    results['P3'] = P3
    results['P4'] = P4
    results['T2'] = T2
    results['T4'] = T4

    ## Find pressure for the compression stroke
    V_compression = kinematics_data_compression['Vd'] + Vc
    P_compression = P1*(V1**k)/(V_compression**k)

    ## Find pressure for the power stroke
    V_power = kinematics_data_power['Vd'] + Vc
    P_power = (P3*V3**k)/(V_power**k)

    theta_rod_compression = -kinematics_data_compression['theta_rod']
    theta_rod_power = kinematics_data_power['theta_rod']

    theta_crank = np.concat((theta_crank_compression, theta_crank_power))
    theta_rod = np.concat((theta_rod_compression, theta_rod_power))
    V = np.concat((V_compression, V_power))
    P = np.concat((P_compression, P_power))
    results['theta_crank'] = theta_crank
    results['theta_rod'] = theta_rod
    results['V'] = V
    results['P'] = P

    ## Compute load F in x and y
    A = np.pi/4*B**2 # Piston Head Area
    F = P*A # Force
    Fx = F*np.sin(theta_rod)
    Fy = -F*np.cos(theta_rod)
    results['F'] = F
    results['Fx'] = Fx
    results['Fy'] = Fy

    ## Evaluate each stress equation and
    ## Compute sigma 1
    x_stresses = []
    y_stresses = []
```

```
    tau_stresses = []
    principal_stresses = []

    for func in funcs:
        sigma_x, sigma_y, tau_xy = func(params, Fx, Fy)
        sigma_1 = (sigma_x + sigma_y)/2 + np.sqrt(((sigma_x + sigma_y)/2)**2 + tau_xy
            **2)

        x_stresses.append(sigma_x)
        y_stresses.append(sigma_y)
        tau_stresses.append(tau_xy)
        principal_stresses.append(sigma_1)


    stresses = {
        'sigma_x': x_stresses,
        'sigma_y': y_stresses,
        'tau_xy': tau_stresses,
        'principal': principal_stresses
    }
    results['stresses'] = stresses

    return results

def plot_results(results):
    theta_crank = results['theta_crank']

    ## Plot the PV diagram
    P = np.append(results['P'], results['P1'])
    V = np.append(results['V'], results['V1'])
    fig, ax = plt.subplots()

    ax.plot(V*10**9, P*10**-3)
    ax.set_xlabel("$Volume (mm^3)$")
    ax.set_ylabel("Pressure (kPa)")
    ax.set_title("PV Diagram")

    ## Plot rod angle
    theta_crank = results['theta_crank']
    theta_rod = results['theta_rod']
    fig, ax = plt.subplots()

    ax.plot(theta_crank, theta_rod)
    ax.set_xlabel(r"$\theta_{crank} (rad)$")
    ax.set_ylabel(r"$\theta_{rod}$ (rad)")
    ax.set_title("Rod Angle vs. Crank Angle")

    ## Plot Forces
    F = results['F']*10**-3
    Fx = results['Fx']*10**-3
    Fy = results['Fy']*10**-3
    fig, ax = plt.subplots()

    ax.plot(theta_crank, F, label = 'F')
    ax.plot(theta_crank, Fx, label = 'Fx')
    ax.plot(theta_crank, Fy, label = 'Fy')
    ax.set_xlabel(r"$\theta_{crank} (rad)$")
    ax.set_ylabel("Force (kN)")
    ax.legend()

    ## Plot x Stresses
    x_stresses = results['stresses']['sigma_x']
    fig, ax = plt.subplots()
    i = 0

    for sigma_x in x_stresses:
        i = i + 1
```

```
        ax.plot(theta_crank, sigma_x, label=r"$\sigma_{n,x}$".replace('n', f'{i}'))

    ax.set_xlabel(r"$\theta_{crank}␣(rad)$")
    ax.set_ylabel("Stress")
    ax.set_title(r"$\sigma_x$")
    ax.legend()

    ## Plot y Stresses
    y_stresses = results['stresses']['sigma_y']
    fig, ax = plt.subplots()
    i = 0

    for sigma_y in y_stresses:
        i = i + 1

        ax.plot(theta_crank, sigma_y, label=r"$\sigma_{n,y}$".replace('n', f'{i}'))

    ax.set_xlabel(r"$\theta_{crank}␣(rad)$")
    ax.set_ylabel("Stress")
    ax.set_title(r"$\sigma_y$")
    ax.legend()

    ## Plot Sheat Stresses
    tau_stresses = results['stresses']['tau_xy']
    fig, ax = plt.subplots()
    i = 0

    for tau_xy in tau_stresses:
        i = i + 1

        ax.plot(theta_crank, tau_xy, label=r"$\tau_{n,xy}$".replace('n', f'{i}'))

    ax.set_xlabel(r"$\theta_{crank}␣(rad)$")
    ax.set_ylabel("Stress")
    ax.set_title(r"$\tau_xy$")
    ax.legend()

    ## Plot Principal Stresses
    principal_stresses = results['stresses']['principal']
    fig, ax = plt.subplots()
    i = 0

    for sigma_x in principal_stresses:
        i = i + 1

        ax.plot(theta_crank, sigma_x, label=r"$\sigma_{n,1}$".replace('n', f'{i}'))

    ax.set_xlabel(r"$\theta_{crank}␣(rad)$")
    ax.set_ylabel("Stress")
    ax.set_title("Principal␣Stresses")
    ax.legend()

    plt.show()

def analyze_results(results):
    x_stresses = results['stresses']['sigma_x']
    y_stresses = results['stresses']['sigma_y']
    tau_stresses = results['stresses']['tau_xy']
    principal_stresses = results['stresses']['principal']
    i = 0

    np.set_printoptions(precision=4)

    print("Maximum␣and␣Average␣Stresses:")
    for sigma_1 in principal_stresses:
        i = i + 1
        avg_sigma_1 = np.mean(sigma_1)*10**-6
        max_sigma_1 = np.max(sigma_1)*10**-6
```

```
        print(f"At point {i}, mean stress = {avg_sigma_1:.3f} MPa, max stress = {
            max_sigma_1:.3f} Mpa")

    print("Stresses at the Start of Combustion:")


def save_results(results, filename):
    x_stresses = results['stresses']['sigma_x']
    y_stresses = results['stresses']['sigma_y']
    shear_stresses = results['stresses']['tau_xy']
    principle_stresses = results['stresses']['principal']

    data = {
        'thetacrank': results['theta_crank'],
        'Fx': results['Fx'],
        'Fy': results['Fy']
    }

    for i in range(len(principle_stresses)):
        data[f"sigmax{i + 1}"] = x_stresses[i]
        data[f"sigmay{i + 1}"] = y_stresses[i]
        data[f"tauxy{i + 1}"] = shear_stresses[i]
        data[f"sigma1{i + 1}"] = principle_stresses[i]

    df = pd.DataFrame(data)

    df.to_csv(f"results_{filename}.csv", index=False)
```

Listing 1: Simulation Code

```
import simulate as sim
import numpy as np

## Define piston/engine parameters
params = {
    # Geometry Parameters
    'B': 0.08, # m, bore
    'S': 0.096, # m, stroke
    'r': 0.2, # m, connecting rod length
    'CR': 10, # compression ratio
    'd_pin': 0.045, # m, pin diameter
    'd_out': 0.06, # m
    'w_beam': 0.05, # m, outer width of beam section
    't_beam': 0.03, # m, outer length of the beam
    'w_web': 0.03, # m, width of the web
    't_web': 0.015, # m
    'w_base': 0.1, # m, width of piston base
    'r_base_fillet': 0.01, # m, fillet where the beam meets base
    'r_web_fillet': 0.0045, # m, fillet inside the beam
     # Stress Concentrations
    'kt': {
        'axial': [
            1.8, # Point 1
            1.65, # Point 2
            2.15, # Point 3
            1.0, # Point 4, not used
            1.0, # Point 5
        ],
        'bending': [
            1.5, # Point 1
            1.0, # Point 2, not used
            1.2, # Point 3
        ]
    },
    # Otto Cycle Parameters
    'T1': 21, # Celcius
    'P1': 101.325*(10**3), # Pa
```

```
    'T3': 1871 # Celcius
}

## Stress functions
def stress_at_1(params, Fx, Fy):
    kt_axial = params['kt']['axial'][0]
    kt_bending = params['kt']['bending'][0]
    w_beam = params['w_beam']
    t_beam = params['t_beam']
    r_rod = params['r']
    w_base = params['w_base']

    A_cross = w_beam*t_beam
    M = Fx*(r_rod - 0.5*w_base)
    sigma_x = kt_bending*((6*M)/(t_beam*w_beam**2))
    sigma_y = kt_axial*(Fy/A_cross)
    tau_xy = 1.5*(Fx/A_cross)

    return sigma_x, sigma_y, tau_xy

def stress_at_2_revised(params, Fx, Fy):
    kt_axial = params['kt']['axial'][1]
    t_web = params['t_web']
    w_web = params['w_web']
    t_beam = params['w_beam']
    w_beam = params['w_beam']

    A_cross = t_web*w_beam
    A_web = w_web*t_web
    A_total = w_beam*t_beam - w_web*(t_beam - t_web)
    A_remaining = A_total - A_web
    concentrated_percent = A_web/A_total

    sigma_x = np.zeros_like(Fx)
    sigma_y = (kt_axial*concentrated_percent*Fy)/A_web
    tau_xy = 1.5*Fx/A_cross

    return sigma_x, sigma_y, tau_xy


def stress_at_3(params, Fx, Fy):
    w_beam = params['w_beam']
    t_beam = params['t_beam']
    w_web = params['w_web']
    t_web = params['t_web']
    r_rod = params['r']
    w_base = params['w_base']
    r_base_fillet = params['r_base_fillet']
    r_web_fillet = params['r_web_fillet']

    A_fillet = (1 - 0.25*np.pi)*r_web_fillet**2

    A_cross = w_beam*t_beam - w_web*(t_beam - t_web) + 4*A_fillet
    M = Fx*(r_rod - 0.5*w_base - r_base_fillet - 0.5*w_web)
    I = (w_web*t_web**3)/12 + ((t_beam**3)/12)*(w_beam - w_web)
    sigma_x = (M*w_beam)/(2*I)
    sigma_y = Fy/A_cross
    tau_xy = 1.5*(Fx/w_beam*t_web)

    return sigma_x, sigma_y, tau_xy

def stress_at_4(params, Fx, Fy):
    kt_axial = params['kt']['axial'][4]
    d_pin = params['d_pin']
    d_out = params['d_out']
    t_beam = params['t_beam']

    A_cross = ((d_out-d_pin)*t_beam)
```

```
    sigma_x = np.zeros_like(Fx)
    sigma_y = kt_axial*Fy/A_cross
    tau_xy = 2*Fx/A_cross

    return sigma_x, sigma_y, tau_xy

funcs = [stress_at_1, stress_at_2_revised, stress_at_3, stress_at_4]

##
npoints = 200

results = sim.simulate_connecting_rod(params, funcs, npoints)

sim.analyze_results(results)
sim.save_results(results, "final")
sim.plot_results(results)
```

Listing 2: Final Design Code