ULS MOMENT AXIAL FORCE INTERACTION DIAGRAM

This script calculates and plots a Moment-Thrust Interaction capacity diagram for a given concrete/shotcrete section.

Written by: Nick Mirsepassi & Amir Mastan Date: 05/06/2024 Revision: 0

IMPORT PYTHON MODULES - READ ONLY

In [2]:
```
# Import Python Modules
import numpy as np
import pandas as pd
import math
import plotly.express as px
import plotly.graph_objects as go
from scipy.stats import linregress
```

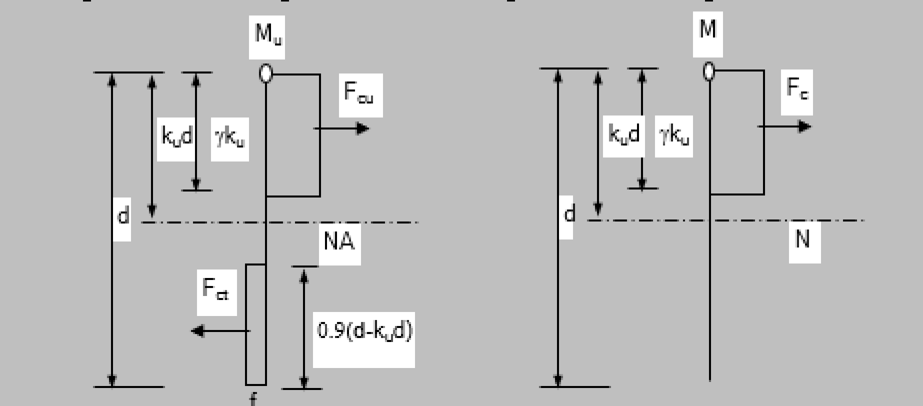
USER INPUT

In [3]:
```
# Inputs Section

depth_mm = 100
width_mm = 1000
fiber_tension_loss_mm = 25

# Material Inputs
fc_mpa = 40
feq_mpa = 3

# Capacity Reduction Factor
phi = 0.6
compression_cut_off = 1
```

CALCULATIONS AND PLOTS - READ ONLY

Rectangular Stress Block Diagram for SFRC & Rectangular Stress Block Diagram for Plain Concrete

- ***Basis for Moment-Thrust Curve***



(b) a uniform compressive stress of $\alpha_2 f'_c$ acts on an area bounded by—

(i) the edges of the cross-section; and

(ii) a line parallel to the neutral axis under the loading concerned, and located at a distance $\gamma k_u d$ from the extreme compressive fibre, where—

$\alpha_2 = 1.0 - 0.003 f'_c$ (within the limits of $0.67 \le \alpha_2 \le 0.85$) . . . 8.1.3(1)

$\gamma = 1.05 - 0.007 f'_c$ (within the limits $0.67 \le \gamma \le 0.85$) . . . 8.1.3(2)

HELPER FUNCTION

In [4]:

```
# Function to calculate parameters
def calculate_parameters(fc_mpa, feq_mpa, depth_mm):
    fcf_mpa = 0.6 * math.sqrt(fc_mpa)
    fe_mpa = 0.37 * feq_mpa
    emax = 0.05 * depth_mm
    return fcf_mpa, fe_mpa, emax

# Function to calculate alpha and gamma (Clause 8.1.3 AS5100)
def calculate_alpha_gamma(fc_mpa):
    gamma = min(max(1.05 - 0.007 * fc_mpa, 0.67), 0.85)
    alpha = min(max(1 - 0.003 * fc_mpa, 0.67), 0.85)
    return gamma, alpha

# Function to calculate pure axial forces
def calculate_pure_axial_forces(depth_mm, fc_mpa, alpha, fiber_tension_loss_mm, feq_mpa)
    pure_compression = depth_mm * fc_mpa * alpha
    pure_tension = -(depth_mm - 2 * fiber_tension_loss_mm) * (0.37 * feq_mpa)
    return pure_compression, pure_tension

# Function to calculate compression bloc
def calculate_compression_block(neutral_axis_depth_mm, gamma, alpha, fc_mpa):
    return neutral_axis_depth_mm * gamma * alpha * fc_mpa

def calculate_compression_lever_arm(neutral_axis_depth_mm, gamma):
    return 0.5 * gamma * neutral_axis_depth_mm

# Function to calculate tension block
def calculate_tension_block(neutral_axis_depth_mm, depth_mm, fiber_tension_loss_mm, fe_m
    return -(0.9 * (depth_mm - neutral_axis_depth_mm) - fiber_tension_loss_mm) * fe_mpa

# Function to calculate axial force
def calculate_tension_lever_arm(neutral_axis_depth_mm, depth_mm, fiber_tension_loss_mm):
    return depth_mm - 0.5 * (0.9 * (depth_mm - neutral_axis_depth_mm - fiber_tension_los

# Function to calculate bending moment and axial forces
def calculate_axial_force(compression_block_mpa, tension_block_mpa):
```

```
    return compression_block_mpa + tension_block_mpa

def calculate_bending_moment(axial_force_kn, compression_block_mpa, compression_lever_ar
                             tension_lever_arm_mm, depth_mm):
    return (0.5 * axial_force_kn * depth_mm -
            compression_block_mpa * compression_lever_arm_mm -
            tension_block_mpa * tension_lever_arm_mm) / 1000

# Function to reduced bending moment and axial forces by reduction factor phi
def calulate_reduced_bending_moment(bending_moment_kn_m, phi, width_mm):
    return bending_moment_kn_m * phi * (width_mm / 1000)

def calulate_reduced_axial_forces(axial_force_kn, phi, width_mm):
    return axial_force_kn * phi * (width_mm / 1000)

def check_reduced_axial_forces_values(reduced_axial_force_kn_SFRC, phi, pure_compression
    for i in range(len(reduced_axial_force_kn_SFRC)):
        if reduced_axial_force_kn_SFRC[i] > phi * pure_compression * compression_cut_off
            reduced_axial_force_kn_SFRC[i] = phi * pure_compression * compression_cut_of

    return reduced_axial_force_kn_SFRC

# Function to reduce pure axial forces
def reduce_pure_tension(pure_tension, phi, pure_compression, compression_cut_off, width_
    if phi * pure_tension <= phi * pure_compression * compression_cut_off:
        reduced_pure_tension = (phi * pure_tension) * (width_mm / 1000)
    else:
        reduced_pure_tension = phi * pure_compression * compression_cut_off * (width_mm

    return reduced_pure_tension

def reduce_pure_compression(phi, pure_compression, compression_cut_off, width_mm):
    if phi * pure_compression <= phi * pure_compression * compression_cut_off:
        reduced_pure_compression = (phi * pure_compression) * (width_mm / 1000)
    else:
        reduced_pure_compression = phi * pure_compression * compression_cut_off * (width_

    return reduced_pure_compression

# Function to add the reduced pure axial forces to reduced bending moment and axial capa
def full_set_axial (modified_reduced_axial_force_kn_SFRC, reduced_pure_tension, reduced_
    modified_reduced_axial_force_kn_SFRC = np.insert(modified_reduced_axial_force_kn_SFR
    modified_reduced_axial_force_kn_SFRC = np.append(modified_reduced_axial_force_kn_SFR
    return modified_reduced_axial_force_kn_SFRC

def full_set_moment (reduced_bending_moment_kn_m_SFRC):
    reduced_bending_moment_kn_m_SFRC = np.insert(reduced_bending_moment_kn_m_SFRC, 0, 0)
    reduced_bending_moment_kn_m_SFRC = np.append(reduced_bending_moment_kn_m_SFRC, 0)
    return reduced_bending_moment_kn_m_SFRC

# Function to calculate unreinforced axial forces and benidng moment
def unreinforced_axial (compression_block_mpa, pure_compression, phi, compression_cut_of
    plain_axial_force = np.append(compression_block_mpa, pure_compression)
    reduced_plain_axial_force = (np.where(phi * plain_axial_force <= phi * pure_compress
                                          compression_cut_off, phi * plain_axial_force *
                                          (width_mm / 1000), phi * pure_compression * co
                                          (width_mm / 1000)))
    return reduced_plain_axial_force

def unreinforced_moment(compression_block_mpa, depth_mm, compression_lever_arm_mm, phi,
    plain_bending_moment = (((compression_block_mpa * 0.5 * depth_mm) -
                             (compression_block_mpa * compression_lever_arm_mm)) / 1000)
    plain_bending_moment = np.append(plain_bending_moment, 0)
    for i in range(len(plain_bending_moment)):
        if plain_bending_moment[i] < 0:
            plain_bending_moment[i] = 0
```

```
        else:
            plain_bending_moment[i] = phi * plain_bending_moment[i] * (width_mm / 1000)
    return plain_bending_moment

# Function to calculate minimum eccentricity coordinates
def minimum_eccentricity_coordinates(phi, pure_compression, width_mm, emax,
                                     reduced_plain_axial_force, reduced_plain_bending_mo
    squash_load_kn = 0.95 * phi * pure_compression * (width_mm / 1000)
    pure_bending_moment_knm = squash_load_kn * (emax / 1000)
    y1 = np.array([0, squash_load_kn])
    x1 = np.array([0, pure_bending_moment_knm])
    slope1, intercept1 = linregress(x1, y1)[:2]
    y2 = reduced_plain_axial_force[-2:]
    x2 = reduced_plain_bending_moment[-2:]
    slope2, intercept2 = linregress(x2, y2)[:2]
    a = - (intercept2 - intercept1) / (slope2 - slope1)
    eccentricity_x_cor = np.array([0, a, 0])
    eccentricity_y_cor = np.array([0, a * slope1, a * slope1])
    return eccentricity_x_cor, eccentricity_y_cor


# Function to update SFRC values according to eccentricity coordinates
def update_SFRC_values(reduced_bending_moment_kn_m_SFRC, modified_reduced_axial_force_kn
    reduced_bending_moment_kn_m_SFRC[-1] = ecc_x [1]
    reduced_bending_moment_kn_m_SFRC = np.append(reduced_bending_moment_kn_m_SFRC,0)
    modified_reduced_axial_force_kn_SFRC[-1] = ecc_y [1]
    modified_reduced_axial_force_kn_SFRC = np.append(modified_reduced_axial_force_kn_SFR
    return reduced_bending_moment_kn_m_SFRC, modified_reduced_axial_force_kn_SFRC

# Function to read structural actions from CSV
def read_user_data(filename):
    user_data = pd.read_csv(filename)
    return user_data
```

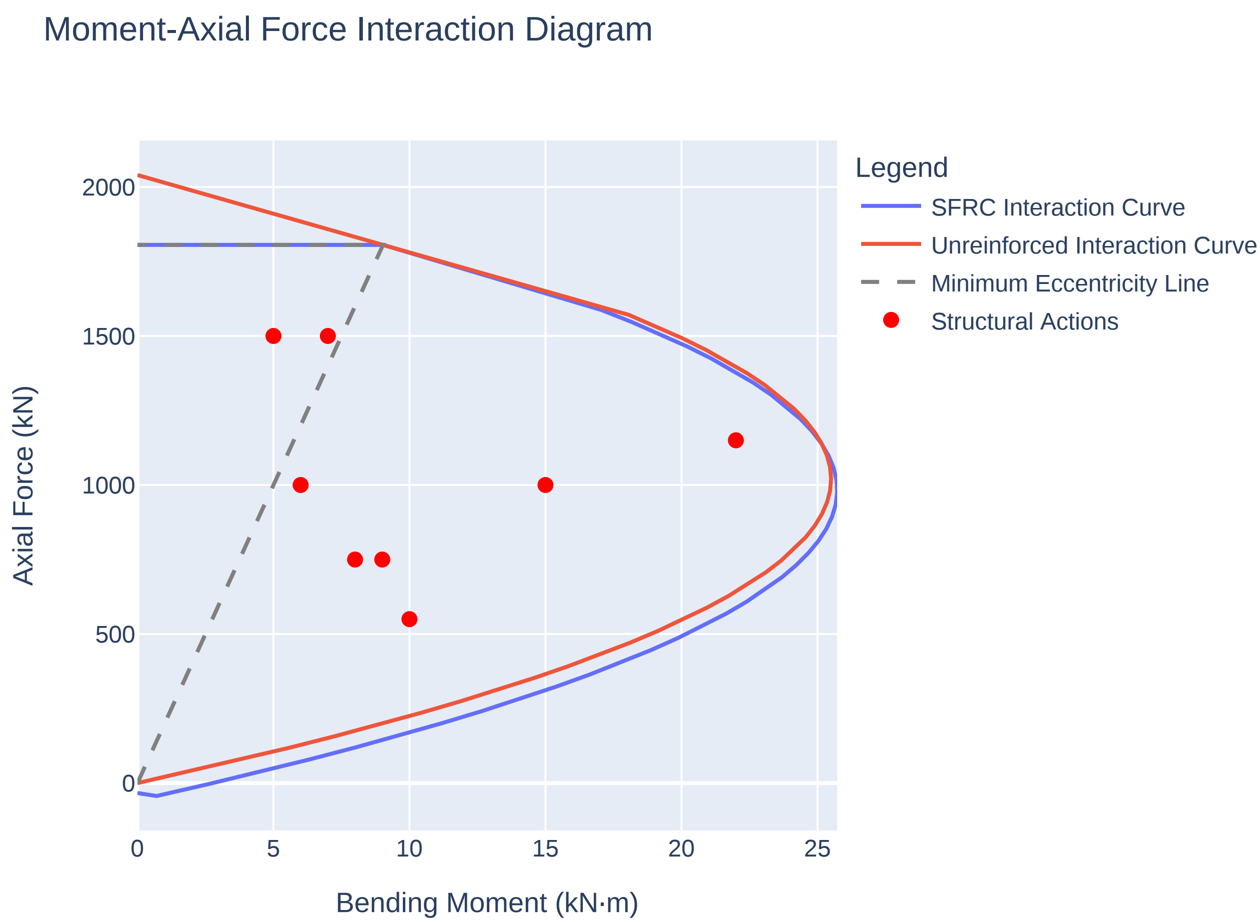
MAIN FUNCTION AND PLOTS

In [5]:

```
def main():
    fcf_mpa, fe_mpa, emax = calculate_parameters(fc_mpa, feq_mpa, depth_mm)
    gamma, alpha = calculate_alpha_gamma(fc_mpa)

    # Create Neutral Axis Depth array
    ku = np.linspace(0, 1, int(1 / 0.025) + 1)
    neutral_axis_depth_mm = ku * depth_mm

    # Calculate pure axial forces
    pure_compression, pure_tension = calculate_pure_axial_forces(depth_mm, fc_mpa, alpha

    # Calculate compression block and lever arms
    compression_block_mpa = calculate_compression_block(neutral_axis_depth_mm, gamma, al
    compression_lever_arm_mm = calculate_compression_lever_arm(neutral_axis_depth_mm, ga

    # Calculate tension block and lever arms
    tension_block_mpa = calculate_tension_block(neutral_axis_depth_mm, depth_mm, fiber_t
    tension_lever_arm_mm = calculate_tension_lever_arm(neutral_axis_depth_mm, depth_mm,

    # Calculate axial force and bending moment
    axial_force_kn = calculate_axial_force(compression_block_mpa, tension_block_mpa)
    bending_moment_kn_m = calculate_bending_moment(axial_force_kn, compression_block_mpa
                                                   compression_lever_arm_mm, tension_blo
                                                   tension_lever_arm_mm, depth_mm)

    # Calculate reduced SFRC bending moment
    reduced_bending_moment_kn_m_SFRC = calulate_reduced_bending_moment(bending_moment_kn
```

```
    reduced_axial_force_kn_SFRC = calulate_reduced_axial_forces(axial_force_kn, phi, wid

    # check the Axial forces value
    modified_reduced_axial_force_kn_SFRC = check_reduced_axial_forces_values(reduced_axi
                                                                             phi, pure_c

    # Reduce pure Axial Forces
    reduced_pure_tension = reduce_pure_tension(pure_tension, phi, pure_compression, comp
    reduced_pure_compression = reduce_pure_compression(phi, pure_compression, compressio

    # full set SFRC bending moment and axial forces
    reduced_bending_moment_kn_m_SFRC = full_set_moment (reduced_bending_moment_kn_m_SFRC
    modified_reduced_axial_force_kn_SFRC = full_set_axial (modified_reduced_axial_force_
                                                           reduced_pure_tension, reduced

    # Calculate Unreinfored Axial forces
    reduced_plain_axial_force = unreinforced_axial (compression_block_mpa, pure_compress
                                                    phi, compression_cut_off, width_mm)
    reduced_plain_bending_moment = unreinforced_moment (compression_block_mpa, depth_mm,
                                                        compression_lever_arm_mm, phi, w

    # Calculate the Minimum eccentricity for graph
    ecc_x, ecc_y = minimum_eccentricity_coordinates(phi, pure_compression,
                                                    width_mm, emax, reduced_plain_axial_

    # Update update_SFRC_values for plot
    reduced_bending_moment_kn_m_SFRC, modified_reduced_axial_force_kn_SFRC = (
        update_SFRC_values(reduced_bending_moment_kn_m_SFRC, modified_reduced_axial_forc

    # Store results in a pandas DataFrame
    interaction_data = pd.DataFrame({
        'Reduced Bending Moment SFRC (kN·m)': reduced_bending_moment_kn_m_SFRC,
        'Reduced Axial Forces SFRC (kN)': modified_reduced_axial_force_kn_SFRC
    })


    # Store results for unreinforced concrete in a pandas DataFrame
    unreinforced_data = pd.DataFrame({
        'Unreinforced Bending Moment (kN·m)': reduced_plain_bending_moment,
        'Unreinforced Axial Force (kN)': reduced_plain_axial_force
    })


    # Plot Interaction Diagram with Plotly
    fig = go.Figure()

    # Add SFRC interaction curve
    fig.add_trace(go.Scatter(
        x=interaction_data['Reduced Bending Moment SFRC (kN·m)'],
        y=interaction_data['Reduced Axial Forces SFRC (kN)'],
        mode='lines',
        name='SFRC Interaction Curve'
    ))

    # Add unreinforced interaction curve
    fig.add_trace(go.Scatter(
        x=unreinforced_data['Unreinforced Bending Moment (kN·m)'],
        y=unreinforced_data['Unreinforced Axial Force (kN)'],
        mode='lines',
        name='Unreinforced Interaction Curve'
    ))

    # Add minimum eccentricity line
    fig.add_trace(go.Scatter(
        x=ecc_x,
        y=ecc_y,
```

```python
        mode='lines',
        name='Minimum Eccentricity Line',
        line=dict(dash='dash', color='gray')
    ))


    # Read user data from CSV
    user_filename = 'structural_actions.csv'
    user_data = read_user_data(user_filename)

    # Add user data to the plot
    fig.add_trace(go.Scatter(
        x=user_data['Bending Moment'],
        y=user_data['Axial Force'],
        mode='markers',
        name='Structural Actions',
        marker=dict(color='red', size=8)
    ))


    # Update layout
    fig.update_layout(
        title='Moment-Axial Force Interaction Diagram',
        xaxis_title='Bending Moment (kN·m)',
        yaxis_title='Axial Force (kN)',
        legend_title='Legend',
        hovermode='closest'
    )


    # Show the plot
    fig.show()



    print("SFRS INTERACTION RAW DATA ...")
    print(interaction_data)

    print("\n\nPLAIN CONCRETE INTERACTION RAW DATA ...")
    print(unreinforced_data)

## =====================================================================================

if __name__ == "__main__":
    main()
```



SFRS INTERACTION RAW DATA ...

	Reduced Bending Moment SFRC (kN·m)	Reduced Axial Forces SFRC (kN)
0	0.000000	-33.300000
1	0.703462	-43.290000
2	2.651830	-2.521500
3	4.521231	38.247000
4	6.311666	79.015500
5	8.023134	119.784000
6	9.655636	160.552500
7	11.209172	201.321000
8	12.683741	242.089500
9	14.079343	282.858000
10	15.395980	323.626500
11	16.633650	364.395000
12	17.792354	405.163500
13	18.872091	445.932000
14	19.872862	486.700500
15	20.794666	527.469000
16	21.637505	568.237500
17	22.401376	609.006000
18	23.086282	649.774500
19	23.692221	690.543000
20	24.219194	731.311500
21	24.667200	772.080000
22	25.036240	812.848500
23	25.326313	853.617000
24	25.537421	894.385500
25	25.669561	935.154000
26	25.722736	975.922500
27	25.696944	1016.691000
28	25.592186	1057.459500
29	25.408461	1098.228000
30	25.145770	1138.996500
31	24.804112	1179.765000

32	24.383489	1220.533500
33	23.883898	1261.302000
34	23.305342	1302.070500
35	22.647819	1342.839000
36	21.911330	1383.607500
37	21.095874	1424.376000
38	20.201452	1465.144500
39	19.228063	1505.913000
40	18.175709	1546.681500
41	17.044387	1587.450000
42	9.027586	1805.517241
43	0.000000	1805.517241

PLAIN CONCRETE INTERACTION RAW DATA ...

	Unreinforced Bending Moment (kN·m)	Unreinforced Axial Force (kN)
0	0.000000	0.00
1	1.925703	39.27
2	3.775810	78.54
3	5.550324	117.81
4	7.249242	157.08
5	8.872566	196.35
6	10.420295	235.62
7	11.892429	274.89
8	13.288968	314.16
9	14.609913	353.43
10	15.855263	392.70
11	17.025018	431.97
12	18.119178	471.24
13	19.137744	510.51
14	20.080714	549.78
15	20.948091	589.05
16	21.739872	628.32
17	22.456059	667.59
18	23.096650	706.86
19	23.661648	746.13
20	24.151050	785.40
21	24.564858	824.67
22	24.903070	863.94
23	25.165689	903.21
24	25.352712	942.48
25	25.464141	981.75
26	25.499975	1021.02
27	25.460214	1060.29
28	25.344858	1099.56
29	25.153908	1138.83
30	24.887362	1178.10
31	24.545223	1217.37
32	24.127488	1256.64
33	23.634159	1295.91
34	23.065234	1335.18
35	22.420716	1374.45
36	21.700602	1413.72
37	20.904894	1452.99
38	20.033590	1492.26
39	19.086693	1531.53
40	18.064200	1570.80
41	0.000000	2040.00