# I. Problem:

Moment-Curvature relationship curve of a reinforced concrete beam of different cases with parameters as follows:

**General Cases**

| Cases | As | As' |
|---|---|---|
| Case 1 | $A_{sb}$ | 0 |
| Case 2 | $0.5A_{sb}$ | 0 |
| Case 3 | $A_{sb}$ | $0.5A_{sb}$ |

**Beam Properties**

| Property | Value | Unit |
|---|---|---|
| f'c | 21 | MPa |
| fy | 275 | MPa |
| fr = 0.7 $\sqrt{f'c}$ | 3.208 | MPa |
| Es | 200,000 | MPa |
| Ec = (4,700$\sqrt{f'c}$) | 21538.10 | MPa |
| $\beta_1$ | 0.85 | |
| η = Es / Ec | 9.28 | |
| b (beam width) | 300 | mm |
| h (beam height) | 450 | mm |
| d (effective depth) | 400 | mm |
| d' (compression steel location) | 50 | mm |

## II. Solutions / Methodology

As a general solution to the problem, analysis as doubly reinforced beam is applied to address all the cases (singly or doubly reinforced). The following are the steps used:

1. Compute for the balanced steel at tension.

   $Asb = 4,539.92mm^2$

2. Steel area is assigned to both tension and compression side as indicated in the general cases.
3. For the 3 stages of the behavior of the beam:

   ### Stage 1 : Cracking point of concrete in tension

   1. Neutral axis location (kd) from the compression fiber of concrete is calculated by transforming area of steel to area of concrete using the modular ratio $\eta$:

      **a. Uncrack section (Case 1 : Doubly Reinforced)**

      | Particulars | Calculated Values |
      |---|---|
      | $As(transformed) = (\eta - 1)As$ | 37,617.24 $mm^2$ |
      | $As'(transformed) = (\eta - 1)As'$ | 18,808.62 $mm^2$ |
      | By taking moment of areas of concrete and steel to topmost fiber: | |
      | $kd$ | 242.19 $mm$ |

   2. Calculation of $M\ cr$ and $\phi\ cr$

      $\epsilon o$ must first be calculated using:

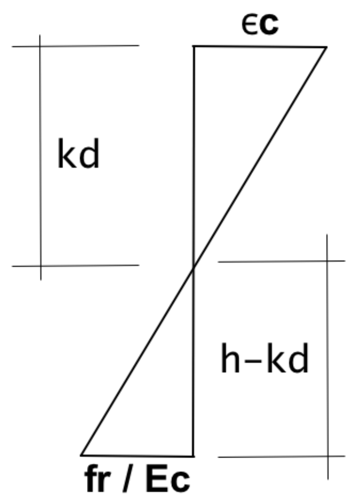
$$\epsilon o = \frac{2 \cdot 0.85 \cdot f'c}{Ec}$$

(

Calculating, we have $\epsilon o = 0.001657$

Now that we have $kd$ and $fr$, we solve for $\epsilon c$ by ratio and proportion based on the strain diagram below:

***Figure 1***

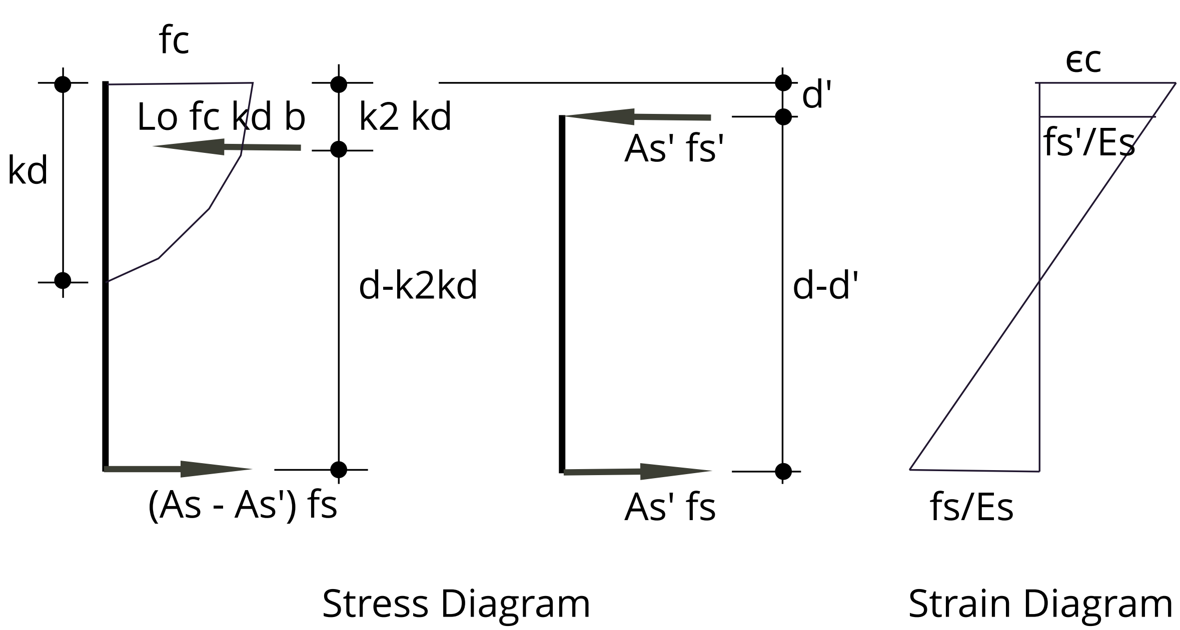



$$\frac{\epsilon c}{kd} = \frac{\frac{fr}{Ec}}{h - kd} \tag{2}$$

Calculating, we have $\epsilon c = 0.00017$

Calculating $fc = \epsilon c \cdot Ec$, we get $fc = 3.697$ MPa

***Figure 2***




From the strain diagram in Figure 2 above, we have,

$$\frac{\epsilon c}{kd} = \frac{\frac{fs'}{Es}}{kd - d'} \tag{3}$$

This gives us, $fs' = 27.2059$ MPa.

Now, we solve for required parameters in PCA stress block, $Lo(k1 \cdot k3)$ and $k2$.

$$\lambda o = \frac{\epsilon c}{\epsilon o} \tag{4}$$

From the calculated $\epsilon o$ and $\epsilon c$, we get $\lambda o = 0.1035$.

For calculating parameters $Lo(k1 \cdot k3)$ and $k2$,

For $0 < \epsilon c < \epsilon o$:

$$Lo = \frac{0.85}{3} \cdot \lambda o \cdot (3 - \lambda o) \tag{5}$$

$$k2 = \frac{1}{4}[\frac{4 - \lambda o}{3 - \lambda o}] \tag{6}$$

For $\epsilon o < \epsilon c < \epsilon cu$

$$Lo = 0.85 \cdot (\frac{3\lambda o - 1}{3\lambda o}) \tag{7}$$

$$k2 = \frac{6(\lambda o)^2 - 4\lambda o + 1}{4\lambda o(3\lambda o - 1)} \tag{8}$$

Since $\epsilon c < \epsilon o$, $k2 = 0.3363$

Following the formula above, we get, $Lo = 0.08498$

Then from the stress diagram above, we take moment at the tension steel,

$$Mc = Lo \cdot fc \cdot kd \cdot b \cdot (d - k2 \cdot kd) + As \cdot fs'(d - d') \tag{9}$$

We then get, $Mcr = 37.5kN \cdot m$

For the curvature $\phi c = \frac{\epsilon c}{kd}$, $\phi c = 7.12x10^{-7} rad/mm$

3. Calculate the curvature $\phi c$ right after cracking

   The neutral axis will shift after the crack, so taking moment of area for transformed steel in tension ($\eta As$) and compression ($(\eta - 1)As$) and concrete at the compressive area into the neutral axis:

$$b \cdot kd \cdot \frac{kd}{2} + (\eta - 1)As' \cdot (kd - d') = \eta As \cdot (d - kd) \tag{10}$$

| Particulars | Calculated Values |
|---|---|
| $kd$ | 196.76 $mm$ |
| $\phi c = \dfrac{\epsilon c}{kd}$ | $8.5303x10^{-7} rad/mm$ |

## Stage 2 : Concrete compression yield at ($fc = 0.5f'c$)

Now let's find the point wherein the concrete yields at a specified stress ($0.5f'c$).

By equilibrium, $C = T$

$$Lo \cdot fc \cdot kd \cdot b + As' \cdot fs' = As \cdot fs \tag{11}$$

By using $fc = 0.5f'c$, $\epsilon c = 0.000487$

Using equations (4), (5) and (6), we obtain, $kd = 194.03mm$

By deriving from the **Strain Diagram** above, we get:

$$fs = Es \cdot \epsilon c \cdot \frac{d - kd}{kd} \tag{12}$$

$$fs' = Es \cdot \epsilon c \cdot \frac{kd - d'}{kd} \tag{13}$$

After this, $fs$ and $fs'$ are compared to $fy$, if any of them is greater than $fy$, steel yields and, so use $fs = fy$ or $fs' = fy$ correspondingly is solving for Moments.

Moment can now be solved using equation (9):

| Particulars | Calculated Values |
|---|---|
| $Mc$ | $130.19kN - m$ |
| $\phi c$ | $2.099x10^{-6} rad/mm$ |

## Stage 3 : Inelastic Stage

At this stage, compression block is no longer triangular, concrete modulus of elasticity is also no longer constant.

Solving for the moment and curvature is divided into two (2) more stages:

1. $0 < \epsilon c < \epsilon o$
2. $\epsilon o < \epsilon c < 0.003$

wherein we iterate in the value of $\epsilon c$

Calculation inside these stages are almost the same except for the calculations of factors such as $k2$, $Lo$ and $fc$:

1. First, $fs$ and $fs'$ are assumed to yield, so to solve for $kd$ using equilibrium:

$$C = T \tag{14}$$

$$Lo \cdot fc \cdot kd \cdot b = (As - As') \cdot fy \tag{15}$$

$$kd = (As - As') \cdot \frac{fy}{Lo \cdot fc \cdot b} \tag{16}$$

2. Now, steel stresses $fs$ and $fs'$ are computed using the calculated $kd$ with equations (3) and (4) respectively then compares them to $fy$:

   a. If $fs > fy$:

   This means steel yields at tension. We then test:

   a.1. if $fs' > fy$: Since the assumption that steel in compression and tension yields, we accept the calculated value of $kd$ then proceeds with $fs = fy$ and $fs' = fy$.

   a.2. if $fs' < fy$:

   Instead of both $As$ and $As'$ multiplying to $fy$ in equation (7), we multiply $As'$ by $fs'$

   in equation (4), thus

$$Lo \cdot fc \cdot kd \cdot b = Asfy - As'fs' \tag{17}$$

$$Lo \cdot fc \cdot kd \cdot b = Asfy - As' \cdot Es \cdot \epsilon c \cdot \frac{kd - d'}{kd} \tag{18}$$

   $kd$ can now be recalculated using quadratic formula, then $fs'$ based on the new calculated $kd$

   b. if $fs < fy$:

   Now, assume the steel at compression yields, $fs' = fy$, we now then get re-write equation (9):

$$Lo \cdot fc \cdot kd \cdot b = As \cdot fs - As'fy \tag{19}$$

$$Lo \cdot fc \cdot kd \cdot b = As \cdot Es \cdot \epsilon c \cdot \frac{d - kd}{kd} - As'fy \tag{20}$$

   We now recalculate $kd$ using the quadratic equation above.

   After calculating $kd$, we might want to check $fs'$ again if compression steel yields to check if assumption in equation (13) is correct. If not, we just replace $fy$ in equation (12) with equation (4) then recalculate $kd$.

   $fs$ and $fs'$ can now be recalculated based on the new $kd$.

3. After getting the stresses in steel, we can now solve for moment by taking moment at the tension steel

$$Mc = Lo \cdot fc \cdot kd \cdot b \cdot (d - k2 \cdot kd) + As' \cdot fs' \cdot (d - d') \tag{21}$$

4. For the curvature, $\phi c$

$$\$\phi c = \epsilon c / kd \qquad (22$$

## III. Results / Charts

Following is the result of the run of the script for the problem in all three (3) cases:

**Case 1**

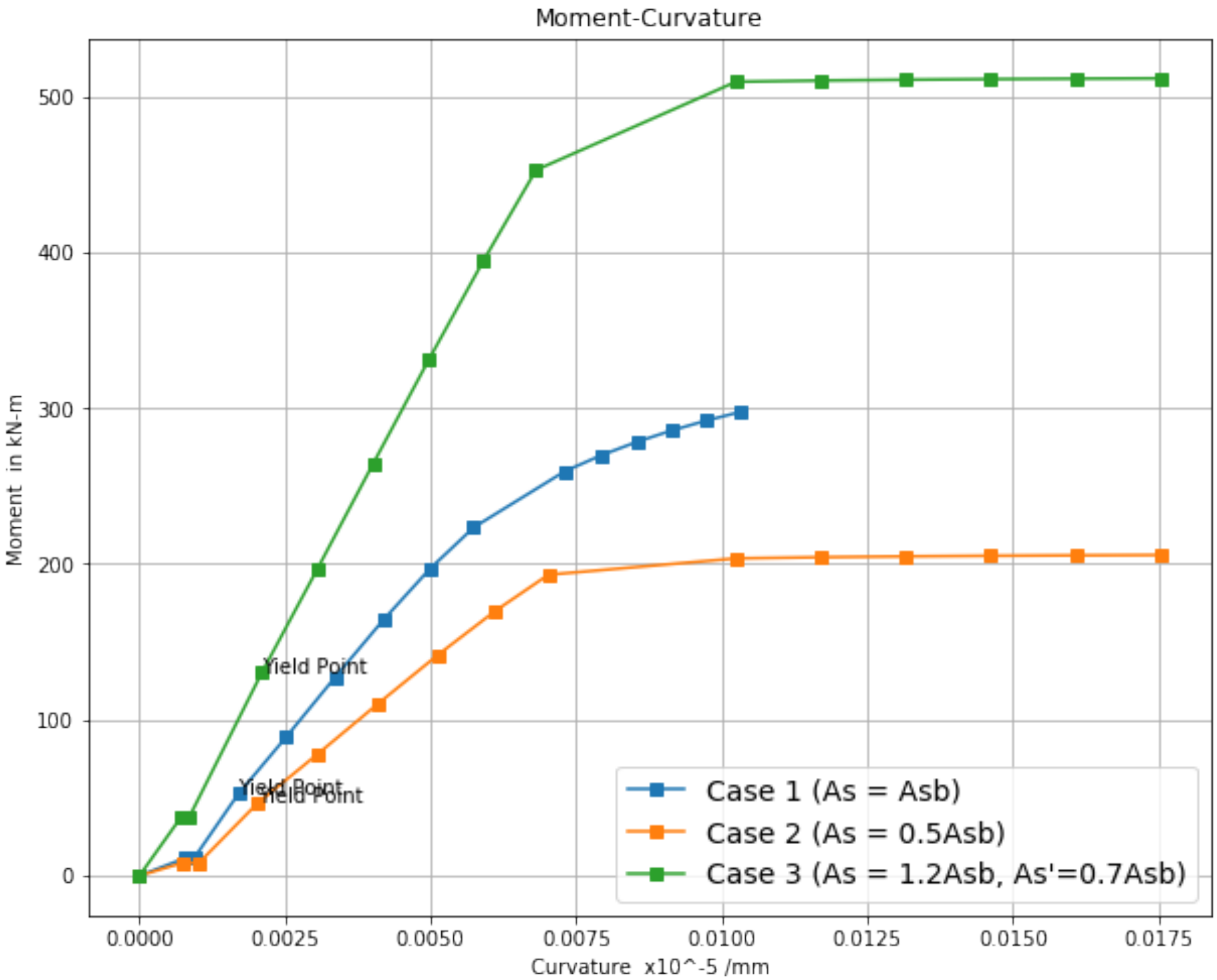
| **Moment ($kN \cdot m$ )** | $\phi c$ **(rad/mm)** | $\epsilon c$ | **Tension Steel Yield** | **Compression Steel Yield** |
|---|---|---|---|---|
| 11.44 | 7.9703e-07 | | | |
| 11.44 | 9.4038e-07 | | | |
| 52.48 | 1.6977e-06 | 0.00048 | | |
| ***For 0 < ϵc < ϵo*** | | | | |
| 89.09 | 2.52e-06 | 0.00069 | False | False |
| 127.38 | 3.36e-06 | 0.00089 | False | False |
| 164.18 | 4.19e-06 | 0.00109 | False | False |
| 196.92 | 4.99e-06 | 0.00129 | False | False |
| 223.4 | 5.74e-06 | 0.00149 | False | False |
| ***For ϵo < ϵc < ϵcu*** | | | | |
| 259.5 | 7.3e-06 | 0.00196 | False | True |
| 269.74 | 7.93e-06 | 0.00216 | False | True |
| 278.35 | 8.54e-06 | 0.00236 | False | True |
| 285.68 | 9.14e-06 | 0.00256 | False | True |
| 292.0 | 9.73e-06 | 0.00276 | False | True |
| 297.51 | 1.032e-05 | 0.00296 | False | True |

**Case 2**

| **Moment ($kN \cdot m$)** | $\phi c$ **(rad/mm)** | $\epsilon c$ | **Tension Steel Yield** | **Compression Steel Yield** |
|---|---|---|---|---|
| 8.11 | 7.315e-07 | | | |
| 8.11 | 1.018e-06 | | | |
| 46.36 | 2.022e-06 | 0.00048 | | |
| ***For 0 < ϵc < ϵo*** | | | | |
| 77.65 | 3.05e-06 | 0.00069 | False | False |
| 110.23 | 4.09e-06 | 0.00089 | False | False |
| 141.62 | 5.12e-06 | 0.00109 | False | False |
| 169.81 | 6.11e-06 | 0.00129 | False | False |
| 193.12 | 7.02e-06 | 0.00149 | False | False |
| ***For ϵo < ϵc < ϵcu*** | | | | |
| 203.58 | 1.024e-05 | 0.00196 | True | True |
| 204.35 | 1.17e-05 | 0.00216 | True | True |
| 204.89 | 1.316e-05 | 0.00236 | True | True |
| 205.27 | 1.462e-05 | 0.00256 | True | True |
| 205.55 | 1.608e-05 | 0.00276 | True | True |
| 205.76 | 1.754e-05 | 0.00296 | True | True |

**Case 3**

| **Moment ($kN \cdot m$)** | $\phi c$ **(rad/mm)** | $\epsilon c$ | **Tension Steel Yield** | **Compression Steel Yield** |
|---|---|---|---|---|
| 37.5 | 7.124e-07 | | | |
| 37.5 | 8.530e-07 | | | |
| 130.19 | 2.099e-06 | 0.00048 | | |
| ***For 0 < ϵc < ϵo*** | | | | |
| 196.57 | 3.05e-06 | 0.00069 | False | False |
| 264.6 | 4.02e-06 | 0.00089 | False | False |
| 331.44 | 4.97e-06 | 0.00109 | False | False |
| 394.8 | 5.91e-06 | 0.00129 | False | False |
| 452.77 | 6.81e-06 | 0.00149 | False | False |
| ***For ϵo < ϵc < ϵcu*** | | | | |
| 509.46 | 1.024e-05 | 0.00196 | True | True |
| 510.23 | 1.17e-05 | 0.00216 | True | True |
| 510.76 | 1.316e-05 | 0.00236 | True | True |
| 511.14 | 1.462e-05 | 0.00256 | True | True |
| 511.42 | 1.608e-05 | 0.00276 | True | True |
| 511.64 | 1.754e-05 | 0.00296 | True | True |




# IV. Comments

---

Following are comments and findings in this problem set.

- The first that I find here is in the chart above, the curve for **Case 1**. It can be seen that it has the smallest curvature. Compared to the curve of **Case 2** which has a smallest amount of tensile reinforcement, shows a gradual change in Moment/Load with a high degree of visibility in change in curvature. The same goes to **Case 3**. This indicates, in my opinion, that they shows ductile behavior. The beams shows a large change in curvature which can be relate to the beams deflection (the larger the angle of curvature, the larger the deflection) while the beam at case 1 shows a brittle behavior. The beam reached its allowable strain of 0.003 in concrete without much change in curvature relative to load.
- Following the 1st comment, if we look at the table in the Results section at Case 1, we can see that the tensile reinforcements did not yield until the beam failed. This could be the reason why we avoid to have a balanced design or even an over reinforced design for that matter.

# V. Appendix

### References


- Gillesania, DI T., *Simplified Reinforced Concrete Design*, Diego Innocencio Tapang Guillesania, 2013
- Ćurić, I., Radić, J., Franetović, M., *DETERMINATION OF THE BENDING MOMENT – CURVATURE RELATIONSHIP FOR REINFORCED CONCRETE HOLLOW SECTION BRIDGE COLUMNS*, n.d.
- American Concrete Institute, *Building Code Requirements for Structural Concrete (ACI 318-95) and Commentary (ACI 318R-95)*, 1995
- Nilson, A. H., Darwin, D., Dolan, C. W., *Design of Concrete Structures 14th ed.*, McGraw-Hill, 2010, Retrieved from http://www.engineeringbookspdf.com


### Source Code

The programming language used in this problem set is **Python3** with the help of **Jupyter Notebook** for presenting the data. The full source code used is shown below. This source code is also available at github (https://github.com/alexiusacademia/masteral-advanced-concrete-design/blob/master/Notebooks/Problem%20Set%201.ipynb)

```
# Imports
import math
import matplotlib.pyplot as plt

# Define parameters
b = 300                             # Beam width
h = 450                             # Beam height
clearance = 50                      # Clearance from tension steel to bottom
of concrete
d = h - clearance                   # d - Effective depth
d_prime = 50                        # d' - Distance from compression steel to
concrete compression fiber
fcprime = 21                        # f'c - Concrete compressive strength
fy = 275                            # fy - Steel tensile strength
fr = 0.7 * math.sqrt(fcprime)       # Modulus of fructure
Es = 200000                         # Modulus of elasticity of steel
Ec = 4700 * math.sqrt(fcprime)      # Modulus of elasticity of concrete
β1 = 0.85                           # Beta
η = Es / Ec                         # Modular ratio

# As balance
```

```
ρb = (0.85 * fcprime * β1 * 600) / (fy * (600 + fy)) # Balance concret-steel
ratio
Asb = ρb * b * d                    # As balance

# Cases
As = [Asb, 0.5*Asb, Asb]       # Tension reinforcements
AsPrime = [0.0, 0.0, 0.5*Asb]      # Compression reinforcements

# Data holders
M = ([], [], [])                    # Array of moments for the 3 cases
ϕ = ([], [], [])                    # Array of curvature for the 3 cases
I = ([], [], [])                    # Array of all computed moment of inertias
kd = ([], [], [])                   # Array of values of neutral axis to
compression fiber
fsm = ([], [], [])                  # Array of strains in concrete
yield_pts = []
```

```
# =========================================
# Utilities
# =========================================
def solveLo(case_no, 𝜆):
    if case_no ==1:
        return 0.85 / 3 * 𝜆 * (3 - 𝜆)
    else:
        return 0.85 * (3*𝜆 - 1) / (3 * 𝜆)
```

```
# Insert initial values for moment and curvature
for i in range(3):
    M[i].append(0.0)
    ϕ[i].append(0.0)

for i in range(3):
    # =========================================== #
    # Calculation before cracking                 #
    # =========================================== #
    # Calculate for kd of each case
    At = b * h                                         # Concrete alone
    At += (η-1) * As[i]                                # Concrete plus
transformed tension steel
    At += (η-1) * AsPrime[i]                           # Plus transformed
compression steel
    Ma = (b * h) * (h / 2)                             # Moment of area of
concrete to compression fiber
    Ma += (η-1) * As[i] * d                            # Moment of tension
reinf. to compression fiber
```

```
    Ma += (η-1) * AsPrime[i] * d_prime                  # Moment of compression reinf. to compression fiber
    kdCalculated = Ma / At
    kd[i].append(kdCalculated)                          # Insert to list of kd

    # Calculate for moment of inertia of each case
    Ic = (b * kdCalculated**3 / 12) + (b * kdCalculated * (kdCalculated / 2)**2)
    Ic += (b * (h - kdCalculated)**3 / 12) + (b * (h - kdCalculated) * ((h - kdCalculated) / 2)**2)
    Ic += (η-1) * As[i] * (d - kdCalculated)**2
    Ic += (η-1) * AsPrime[i] * (kdCalculated - d_prime)**2
    I[i].append(Ic)                                     # Insert to list of I

    # Calculate the cracking moment
    Mcr = fr* Ic / (h - kdCalculated)                   # Cracking moment
    M[i].append(Mcr)                                    # Insert to list of M

    # Calculate the curvature
    ϕc = fr / (Ec * (h - kdCalculated))                 # Curvature right before cracking
    ϕ[i].append(ϕc)                                     # Insert to list of ϕ

    # =========================================== #
    # Calculation after cracking                  #
    # =========================================== #
    # Finding the neutral axis using equilibrium of moment of areas
    # b(kd)(kd/2) + (n-1)As'(kd-d') = nAs(d-kd)
    # -- solve the quadratic equation
    qa = b
    qb = 2 * ((η-1) * AsPrime[i] + η * As[i])
    qc = -2 * ((η-1) * AsPrime[i] * d_prime + η * As[i] * d)
    qd = (qb**2) - (4 * qa * qc)                        # Discriminant
    kdCalculated = (-1 * qb + math.sqrt(qd)) / (2 * qa) # Neutral axis after cracking
    kd[i].append(kdCalculated)

    # Calculate moment of inertia
    Ic = (b * kdCalculated**3 / 12) + (b * kdCalculated * (kdCalculated / 2)**2)
    Ic += (η) * As[i] * (d - kdCalculated)**2
    Ic += (η-1) * AsPrime[i] * (kdCalculated - d_prime)**2
    I[i].append(Ic)

    # Calculate the curvature
```

```
    ϕc = M[i][1] / (Ec * Ic)                              # Curvature right
after cracking

    M[i].append(Mcr)
    ϕ[i].append(ϕc)

    # =========================================== #
    # Calculation at yield point                  #
    # =========================================== #
    fc = 0.5 * fcprime
    ϵc = fc / Ec

    qa = 0.5 * fc * b
    qb = (Es * ϵc) * (AsPrime[i] + As[i])
    qc = -(Es * ϵc) * (AsPrime[i] * d_prime + As[i] * d)
    qd = (qb**2) - (4 * qa * qc)           # Discriminant
    kdCalculated = (-1 * qb + math.sqrt(qd)) / (2 * qa)

    fs = (Es * ϵc) * (d - kdCalculated) / kdCalculated
    fsPrime = Es * ϵc / kdCalculated * (kdCalculated - d_prime)
    if fs > fy:
        fs = fy

    if fsPrime > fy:
        fsPrime = fys

    Mc = 0.5 * fc * b * kdCalculated * (d - kdCalculated / 3) +\
                AsPrime[i] * fsPrime * (d - d_prime)
    ϕc = ϵc / kdCalculated

    M[i].append(Mc)
    ϕ[i].append(ϕc)

    yield_pts.append((ϕc*1000, Mc / 1000**2))

    # =========================================== #
    # Calculation at inelastic behaviour          #
    # =========================================== #
    # Calculate for ϵo
    ϵo = 2 * 0.85 * fcprime / Ec        # This is overridden below

    # Iterator increment
    iterator_increment = 0.0002

    # For 0 < ϵc < ϵo
    ϵc = 0.5 * ϵo                       # To override above ϵo
```

```
    # For case 0 < ϵc < ϵo
    while (ϵc + iterator_increment) <= ϵo:
        ϵc = ϵc + iterator_increment
        λo = ϵc / ϵo
        k2 = 1 / 4 * (4 - λo) / (3 - λo)
        Lo = solveLo(1, λo)
        fc = 0.85 * fcprime * (2 * λo - λo**2)
        kdCalculated = (As[i] - AsPrime[i]) * fy / (Lo * fc * b)
        fs = (Es * ϵc) * (d - kdCalculated) / kdCalculated
        fsPrime = Es * ϵc / kdCalculated * (kdCalculated - d_prime)

        if fs >= fy:                                    # Tension steel yields
            # Solve for the stress in compression steel
            if fsPrime < fy:
                # Compression steel does not yields
                qa = Lo * fc * b
                qb = (Es * ϵc) * AsPrime[i] - As[i] * fy
                qc = -(Es * ϵc) * AsPrime[i] * d_prime
                qd = (qb**2) - (4 * qa * qc)           # Discriminant
                kdCalculated = (-1 * qb + math.sqrt(qd)) / (2 * qa)
                fs = (Es * ϵc) * (d - kdCalculated) / kdCalculated
                fsPrime = Es * ϵc / kdCalculated * (kdCalculated - d_prime)
            else:
                # fs and fs' > fy
                kdCalculated = (As[i] - AsPrime[i]) * fy / (Lo * fc * b)
                fs = fy
                fsPrime = fy
        else:
            qa = Lo * fc * b
            qb = AsPrime[i] * fy + As[i] * Es * ϵc
            qc = -As[i] * Es * ϵc * d
            qd = (qb**2) - (4 * qa * qc)           # Discriminant
            kdCalculated = (-1 * qb + math.sqrt(qd)) / (2 * qa)
            fs = (Es * ϵc) * (d - kdCalculated) / kdCalculated
            fsPrime = Es * ϵc / kdCalculated * (kdCalculated - d_prime)

            if fsPrime < fy:
                # Compression syeel did not yield
                # Compression steel does not yields
                qa = Lo * fc * b
                qb = (Es * ϵc) * (AsPrime[i] + As[i])
                qc = -(Es * ϵc) * (As[i] * d + AsPrime[i] * d_prime)
                qd = (qb**2) - (4 * qa * qc)           # Discriminant
                kdCalculated = (-1 * qb + math.sqrt(qd)) / (2 * qa)
                fs = (Es * ϵc) * (d - kdCalculated) / kdCalculated
```

```python
                fsPrime = Es * ϵc / kdCalculated * (kdCalculated - d_prime)

        Mc = Lo * fc * b * kdCalculated * (d - k2 * kdCalculated) +\
                AsPrime[i] * fsPrime * (d - d_prime)
        ϕc = ϵc / kdCalculated

        M[i].append(Mc)
        ϕ[i].append(ϕc)

    # For case ϵo < ϵc < ϵcu
    ϵc = ϵo + 0.0001
    while (ϵc + iterator_increment) <= 0.003:
        ϵc = ϵc + iterator_increment
        ζc = ϵo / ϵc
        𝜆o = 1 / ζc
        Lo = solveLo(2, 𝜆o)
        k2 = (6 * 𝜆o**2 - 4 * 𝜆o + 1) / (4 * 𝜆o * (3 * 𝜆o - 1))
        fc = 0.85 * fcprime
        kdCalculated = (As[i] - AsPrime[i]) * fy / (Lo * fc * b)
        fs = (Es * ϵc) * (d - kdCalculated) / kdCalculated
        fsPrime = Es * ϵc / kdCalculated * (kdCalculated - d_prime)

        if fs >= fy:                                    # Tension steel yields
            # Solve for the stress in compression steel
            if fsPrime < fy:
                # Compression steel does not yields
                qa = Lo * fc * b
                qb = (Es * ϵc) * AsPrime[i] - As[i] * fy
                qc = -(Es * ϵc) * AsPrime[i] * d_prime
                qd = (qb**2) - (4 * qa * qc)           # Discriminant
                kdCalculated = (-1 * qb + math.sqrt(qd)) / (2 * qa)
                fs = (Es * ϵc) * (d - kdCalculated) / kdCalculated
                fsPrime = Es * ϵc / kdCalculated * (kdCalculated - d_prime)
            else:
                # fs and fs' > fy
                kdCalculated = (As[i] - AsPrime[i]) * fy / (Lo * fc * b)
                fs = fy
                fsPrime = fy
        else:
            qa = Lo * fc * b
            qb = AsPrime[i] * fy + As[i] * Es * ϵc
            qc = -As[i] * Es * ϵc * d
            qd = (qb**2) - (4 * qa * qc)           # Discriminant
            kdCalculated = (-1 * qb + math.sqrt(qd)) / (2 * qa)
            fs = (Es * ϵc) * (d - kdCalculated) / kdCalculated
            fsPrime = Es * ϵc / kdCalculated * (kdCalculated - d_prime)
```

```
            if fsPrime < fy:
                # Compression syeel did not yield
                # Compression steel does not yields
                qa = Lo * fc * b
                qb = (Es * ϵc) * (AsPrime[i] + As[i])
                qc = -(Es * ϵc) * (As[i] * d + AsPrime[i] * d_prime)
                qd = (qb**2) - (4 * qa * qc)           # Discriminant
                kdCalculated = (-1 * qb + math.sqrt(qd)) / (2 * qa)
                fs = (Es * ϵc) * (d - kdCalculated) / kdCalculated
                fsPrime = Es * ϵc / kdCalculated * (kdCalculated - d_prime)

        Mc = Lo * fc * b * kdCalculated * (d - k2 * kdCalculated) +\
                AsPrime[i] * fsPrime * (d - d_prime)
        ϕc = ϵc / kdCalculated

        ϕ[i].append(ϕc)
        M[i].append(Mc)
```

```
# Convert the values of data to smaller figures before plotting
ϕ_converted = ([], [], [])
M_converted = ([], [], [])
for i in range(3):
    for curvature in ϕ[i]:
        ϕ_converted[i].append(curvature * 1000)
    for moment in M[i]:
        M_converted[i].append(moment / 1000**2)

# Plot the curves
plt.figure(figsize=(10,8))
plt.title("Moment-Curvature")
plt.xlabel('Curvature  x10^-5 /mm')
plt.ylabel('Moment  in kN-m')
plt.grid()

for yp in yield_pts:
    plt.text(yp[0], yp[1], 'Yield Point')

# Plot the converted values
case1, = plt.plot(ϕ_converted[0], M_converted[0], marker='s', label='Case 1
(As = Asb)')
case2, = plt.plot(ϕ_converted[1], M_converted[1], marker='s', label='Case 2
(As = 0.5Asb)')
case3, = plt.plot(ϕ_converted[2], M_converted[2], marker='s', label='Case 3
(As = 1.2Asb, As\'=0.7Asb)')
```

```
plt.legend(handles=[case1, case2, case3], loc='best', fontsize=14)
plt.show()
```