### 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 | Asb | 0 |
| Case 2 | 0.5Asb | 0 |
| Case 3 | Asb | 0.5Asb |

**Beam Properties**

|  |  |  |
| --- | --- | --- |
| Property | Value | Unit |
| f'c | 21 | MPa |
| fy | 275 | MPa |
| fr = 0.7 | 3.208 | MPa |
| Es | 200,000 | MPa |
| Ec = (4,700) | 21538.10 | MPa |
| β1 | 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.
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 :

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

|  |  |
| --- | --- |
| * + Particulars | * + Calculated Values |
|  | * + 37,617.24 |
|  | * + 18,808.62 |
| * + By taking moment of areas of concrete and steel to topmost fiber: |  |
|  | * + 242.19 |

* 1. Calculation of and

|  |  |
| --- | --- |
| * + Particulars | * + Calculated Values |
|  |  |
|  |  |
|  |  |

* 1. Calculate the curvature right after cracking
  + The neutral axis will shift after the crack, so taking moment of area for transformed steel in tension () and compression () and concrete at the compressive area into the neutral axis:

|  |  |
| --- | --- |
| * + Particulars | * + Calculated Values |
|  | * + 196.76 |
|  |  |
|  |  |

#### Stage 2 : Concrete compression yield at ()

* At this stage, compression block is still assumed linear and so can be represented by a triangular shape as shown.
* 
* By equilibrium,
* Solving kd using quadratic formula,
* By deriving from the **Strain Diagram** above, we get:

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

Moment can now be solved by taking moment to tension steel:

|  |  |
| --- | --- |
| Particulars | Calculated Values |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

#### 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:

* wherein we iterate in the value of

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

1. First, and are assumed to yield, so to solve for using equilibrium:



1. Now, steel stresses and are computed using the calculated with equations (3) and (4) respectively then compares them to :

* a. If :
* This means steel yields at tension. We then test:
* a.1. if : Since the assumption that steel in compression and tension yields, we accept the calculated value of then proceeds with and .
* a.2. if :
* Instead of both and multiplying to in equation (7), we multiply by
* in equation (4), thus

* can now be recalculated using quadratic formula, then based on the new calculated
* b. if :
* Now, assume the steel at compression yields, , we now then get re-write equation (9):

* We now recalculate using the quadratic equation above.
* After calculating , we might want to check again if compression steel yields to check if assumption in equation (13) is correct. If not, we just replace in equation (12) with equation (4) then recalculate .
* and can now be recalculated based on the new .

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



1. For the curvature,



$$$\phi c = \epsilon c / kd\qquad\text{(14)}$$

### III. Results / Charts

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

##### Case 1

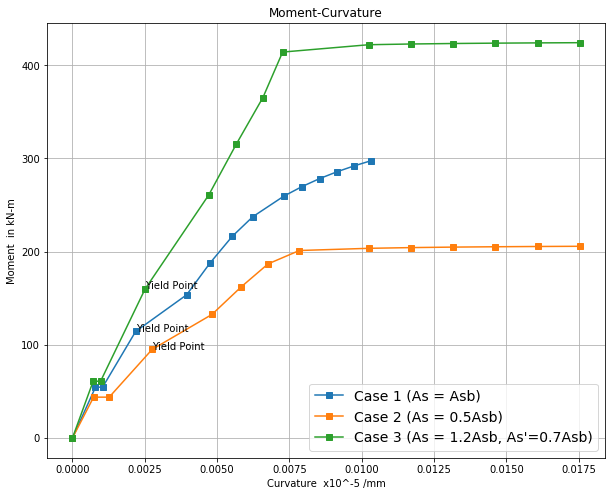
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Moment () | (rad/mm) |  | Tension Steel Yield | Compression Steel Yield |
| 54.57 | 7.9703e-07 |  |  |  |
| 54.57 | 1.0428e-06 |  |  |  |
| 114.39 | 2.1859e-06 | 0.00048 |  |  |
| ***For 0 < ⲉc < ⲉo*** |  |  |  |  |
| 153.69 | 3.95e-06 | 0.00103 | False | False |
| 187.86 | 4.76e-06 | 0.00123 | False | False |
| 216.42 | 5.52e-06 | 0.00143 | False | False |
| 237.47 | 6.23e-06 | 0.00163 | False | False |
| ***For ⲉo < ⲉc < ⲉcu*** |  |  |  |  |
| 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 () | (rad/mm) |  | Tension Steel Yield | Compression Steel Yield |
| 43.86 | 7.315e-07 |  |  |  |
| 43.86 | 1.270e-06 |  |  |  |
| 95.07 | 2.754e-06 | 0.00048 |  |  |
| ***For 0 < ⲉc < ⲉo*** |  |  |  |  |
| 132.65 | 4.82e-06 | 0.00103 | False | False |
| 161.97 | 5.83e-06 | 0.00123 | False | False |
| 186.87 | 6.76e-06 | 0.00143 | False | False |
| 201.24 | 7.85e-06 | 0.00163 | True | False |
| ***For ⲉo < ⲉc < ⲉcu*** |  |  |  |  |
| 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 |

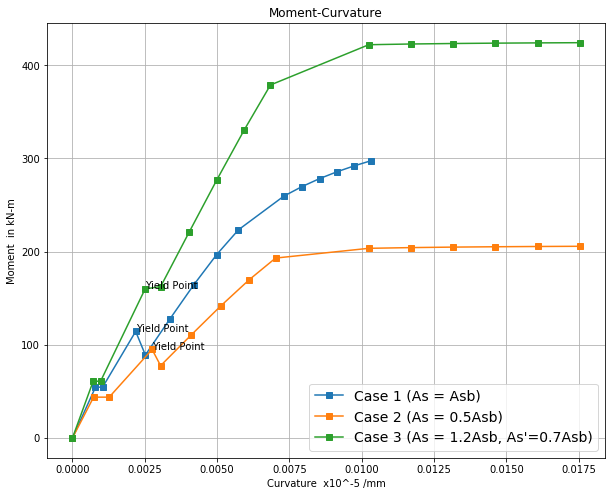
##### Case 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Moment () | (rad/mm) |  | Tension Steel Yield | Compression Steel Yield |
| 60.97 | 7.167e-07 |  |  |  |
| 60.97 | 9.733e-07 |  |  |  |
| 159.98 | 2.512e-06 | 0.00048 |  |  |
| ***For 0 < ⲉc < ⲉo*** |  |  |  |  |
| 260.98 | 4.72e-06 | 0.00103 | False | False |
| 315.58 | 5.67e-06 | 0.00123 | False | False |
| 365.39 | 6.58e-06 | 0.00143 | False | False |
| 414.18 | 7.27e-06 | 0.00163 | True | False |
| ***For ⲉo < ⲉc < ⲉcu*** |  |  |  |  |
| 422.07 | 1.024e-05 | 0.00196 | True | True |
| 422.84 | 1.17e-05 | 0.00216 | True | True |
| 423.37 | 1.316e-05 | 0.00236 | True | True |
| 423.75 | 1.462e-05 | 0.00256 | True | True |
| 424.03 | 1.608e-05 | 0.00276 | True | True |
| 424.25 | 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.
* After the elastic stage, I also noticed the change in slope in the range of . The slope goes steeper and steeper until approaches and then it abruptly goes almost flat. I'm not sure if this though if this is what's called the *strain hardening* before the crushing of concrete at failure.
* Lastly is the comparison of calculated moment at from that computed at yield point using triangular stress block (*end of elastic stage*) and that of computing it using PCA stress block (*considering inelastic stage*). In the figure at the result above, I started the iteration way far ahead of calculated strain (0.5f'c). The moment calculated using inelastic approach is less than of that computed using elastic approach at as shown below (for Case 1 and Case 2).
* 

### 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  
 ⲉc = fc / Ec  
   
 qa = 0.5 \* fc \* b  
 qb = (Es \* ⲉc) \* (AsPrime[i] + As[i])  
 qc = -(Es \* ⲉc) \* (AsPrime[i] \* d\_prime + As[i] \* d)  
 qd = (qb\*\*2) - (4 \* qa \* qc) # Discriminant  
 kdCalculated = (-1 \* qb + math.sqrt(qd)) / (2 \* qa)  
  
 fs = (Es \* ⲉc) \* (d - kdCalculated) / kdCalculated  
 fsPrime = Es \* ⲉc / 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 = ⲉc / kdCalculated  
   
 M[i].append(Mc)  
 ϕ[i].append(ϕc)  
   
 yield\_pts.append((ϕc\*1000, Mc / 1000\*\*2))  
   
 # =========================================== #  
 # Calculation at inelastic behaviour #  
 # =========================================== #  
 # Calculate for ⲉo  
 ⲉo = 2 \* 0.85 \* fcprime / Ec # This is overridden below  
   
 # Iterator increment  
 iterator\_increment = 0.0002  
   
 # For 0 < ⲉc < ⲉo  
 ⲉc = 0.5 \* ⲉo # To override above ⲉo  
  
 # For case 0 < ⲉc < ⲉo  
 while (ⲉc + iterator\_increment) <= ⲉo:  
 ⲉc = ⲉc + iterator\_increment  
 𝜆o = ⲉc / ⲉo  
 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 \* ⲉc) \* (d - kdCalculated) / kdCalculated  
 fsPrime = Es \* ⲉc / 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 \* ⲉc) \* AsPrime[i] - As[i] \* fy  
 qc = -(Es \* ⲉc) \* AsPrime[i] \* d\_prime  
 qd = (qb\*\*2) - (4 \* qa \* qc) # Discriminant  
 kdCalculated = (-1 \* qb + math.sqrt(qd)) / (2 \* qa)  
 fs = (Es \* ⲉc) \* (d - kdCalculated) / kdCalculated  
 fsPrime = Es \* ⲉc / 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 \* ⲉc  
 qc = -As[i] \* Es \* ⲉc \* d  
 qd = (qb\*\*2) - (4 \* qa \* qc) # Discriminant  
 kdCalculated = (-1 \* qb + math.sqrt(qd)) / (2 \* qa)  
 fs = (Es \* ⲉc) \* (d - kdCalculated) / kdCalculated  
 fsPrime = Es \* ⲉc / kdCalculated \* (kdCalculated - d\_prime)  
   
 if fsPrime < fy:  
 # Compression syeel did not yield  
 # Compression steel does not yields  
 qa = Lo \* fc \* b  
 qb = (Es \* ⲉc) \* (AsPrime[i] + As[i])  
 qc = -(Es \* ⲉc) \* (As[i] \* d + AsPrime[i] \* d\_prime)  
 qd = (qb\*\*2) - (4 \* qa \* qc) # Discriminant  
 kdCalculated = (-1 \* qb + math.sqrt(qd)) / (2 \* qa)  
 fs = (Es \* ⲉc) \* (d - kdCalculated) / kdCalculated  
 fsPrime = Es \* ⲉc / kdCalculated \* (kdCalculated - d\_prime)  
   
 Mc = Lo \* fc \* b \* kdCalculated \* (d - k2 \* kdCalculated) +\  
 AsPrime[i] \* fsPrime \* (d - d\_prime)  
 ϕc = ⲉc / kdCalculated  
   
 M[i].append(Mc)  
 ϕ[i].append(ϕc)  
   
 # For case ⲉo < ⲉc < ⲉcu  
 ⲉc = ⲉo + 0.0001  
 while (ⲉc + iterator\_increment) <= 0.003:  
 ⲉc = ⲉc + iterator\_increment  
 ζc = ⲉo / ⲉc  
 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 \* ⲉc) \* (d - kdCalculated) / kdCalculated  
 fsPrime = Es \* ⲉc / 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 \* ⲉc) \* AsPrime[i] - As[i] \* fy  
 qc = -(Es \* ⲉc) \* AsPrime[i] \* d\_prime  
 qd = (qb\*\*2) - (4 \* qa \* qc) # Discriminant  
 kdCalculated = (-1 \* qb + math.sqrt(qd)) / (2 \* qa)  
 fs = (Es \* ⲉc) \* (d - kdCalculated) / kdCalculated  
 fsPrime = Es \* ⲉc / 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 \* ⲉc  
 qc = -As[i] \* Es \* ⲉc \* d  
 qd = (qb\*\*2) - (4 \* qa \* qc) # Discriminant  
 kdCalculated = (-1 \* qb + math.sqrt(qd)) / (2 \* qa)  
 fs = (Es \* ⲉc) \* (d - kdCalculated) / kdCalculated  
 fsPrime = Es \* ⲉc / kdCalculated \* (kdCalculated - d\_prime)  
   
 if fsPrime < fy:  
 # Compression syeel did not yield  
 # Compression steel does not yields  
 qa = Lo \* fc \* b  
 qb = (Es \* ⲉc) \* (AsPrime[i] + As[i])  
 qc = -(Es \* ⲉc) \* (As[i] \* d + AsPrime[i] \* d\_prime)  
 qd = (qb\*\*2) - (4 \* qa \* qc) # Discriminant  
 kdCalculated = (-1 \* qb + math.sqrt(qd)) / (2 \* qa)  
 fs = (Es \* ⲉc) \* (d - kdCalculated) / kdCalculated  
 fsPrime = Es \* ⲉc / kdCalculated \* (kdCalculated - d\_prime)  
   
 Mc = Lo \* fc \* b \* kdCalculated \* (d - k2 \* kdCalculated) +\  
 AsPrime[i] \* fsPrime \* (d - d\_prime)  
 ϕc = ⲉc / 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()