Homework 3

Louis Lin

February 11, 2021

University of California, San Diego

Prof. José I. Restrepo

SE 211 Advance Structural Concrete

Contents

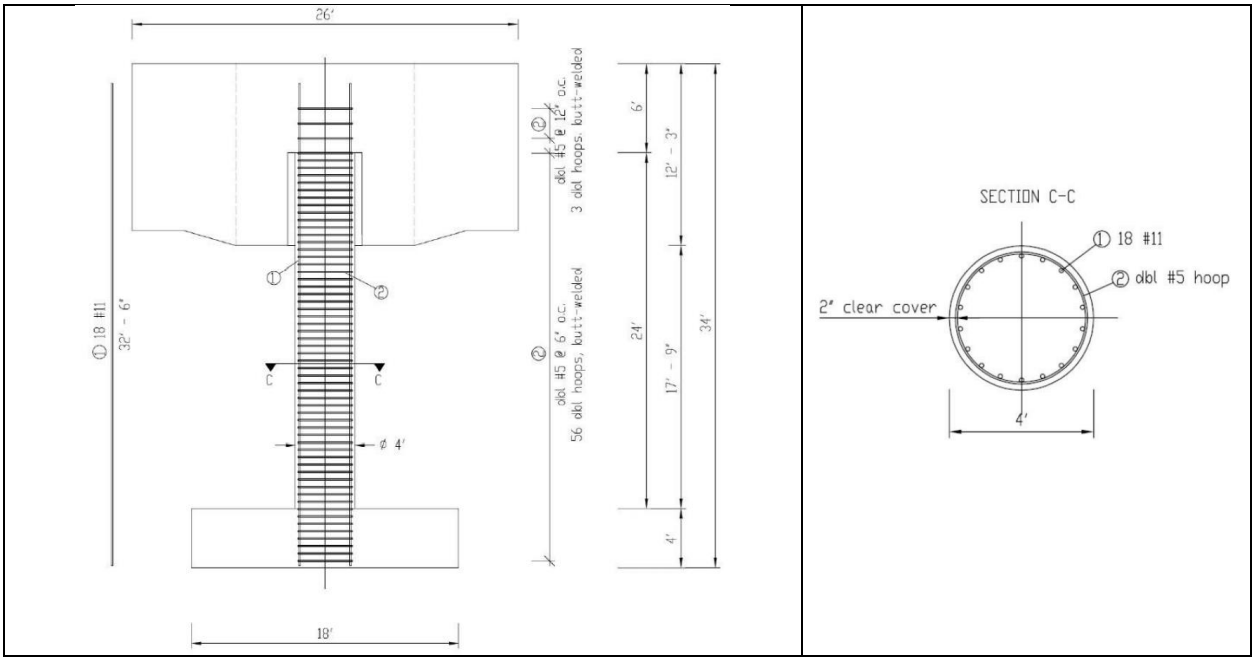
List of Figures

List of Tables

Part I – Moment Curvature Program

A moment curvature plot shows the moment developed by a section as it bends. For a prismatic body with a given cross section assuming that plane sections remain plane under bending, a linear strain profile can be assumed for the cross section. The strain profile can then be equated to a stress profile for the cross section. If the cross section is discretized, the force in each discretized layer can be found by multiplying the area of the layer with a corresponding mean stress value for the layer. This force produced a moment in the cross section given the layer's distance from the centroid. Thus, for a given curvature of bending of the cross section, the internal resisting moment can be found. If the moment is plotted for a range of curvature this is the moment curvature curve (Mφ curve).

For this report, a Mφ curve was developed for a reinforced concrete column with 4' diameter and 24' effective length from the base to the center of the mass block above. The reinforcement is shown in Figure 1.



(a) Column Reinforcement (b) Column Cross-Section

Figure 1 Reinforcement in Column

The longitudinal reinforcement was assumed to be lumped into 5 layers as given in Table 1. Lumping the steel rebar into these equivalent layers eliminates the need to model every layer of longitudinal reinforcement while still capturing the effects and behavior for the Mφ curve. This was not proved for this report.

Table 1 Reinforcement Layers

Reinforcement Fiber Layer [i]	Distance from base y_i [in]	Area of steel A_{si} [in^2]
1	3.35	4.68
2	8.81	6.24
3	24.00	6.24
4	39.19	6.24
5	44.65	4.68

The properties of the concrete are listed in Table 2 as provided by the instructor and were used for the constitutive model of the concrete. Properties of the longitudinal and hoop reinforcements are given in Table 3 and Table 4. Further calculations of the constitutive models are shown in appendix A.

Table 2 Concrete Properties

		Confined Concrete		Unconfined Concrete	
Compressive Strength	f_c'	7043.3	ksi	6000	ksi
Tensile strength	f_t	440	psi	440	psi
Elastic Modulus	E_c	3250	ksi	3250	ksi
Strain at Compressive Strength	ϵ'_c	-0.499%		-0.270%	
Ultimate Compressive Strain	ϵ'_{cu}	-2.000%		-0.600%	
Tensile Cracking strain	ϵ_t	0.0135%		0.0135%	

Table 3 Hoop Reinforcement Properties

Area	A_s	0.614	in^2
Steel Ratio	ρ_s	0.235%	
Hoop Diameter	D	43.375	in
Bar Diameter	d_b	0.625	in^2
Hoop Spacing	s	6.000	in
Yielding Stress	f_y	60.400	ksi

Table 4 Longitudinal Reinforcement Properties

Area	A_s	0.614	in^2
Steel Ratio	ρ_s	0.235%	
Elastic Modulus	E_s	29000	ksi
Yielding Stress	f_y	60.400	ksi
Ultimate Stress	f_{su}	101.700	ksi
Yielding Strain	ϵ_y	2.56%	
Onset of Strain Hardening	ϵ_{sh}	1.45%	
Ultimate Strain	ϵ_{su}	10.5%	

In the report, 100 fibers were used to model the cross section of the column. The discretization of confined concrete and unconfined concrete is shown in Figure 2.

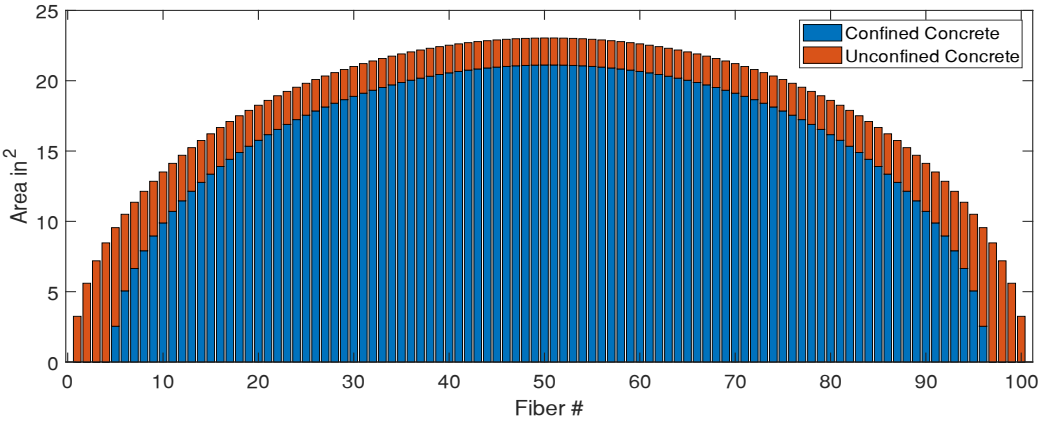


Figure 2 Discretized Area of Confined and Unconfined Concrete

Using the properties of the concrete and reinforcing steel as given above, the constitutive model for the materials can be generated as shown in Figure 3. This relates the strain of the material with a stress level. In this report, the Mφ graph is assumed to be monotonically loaded so a cyclic model was not used.

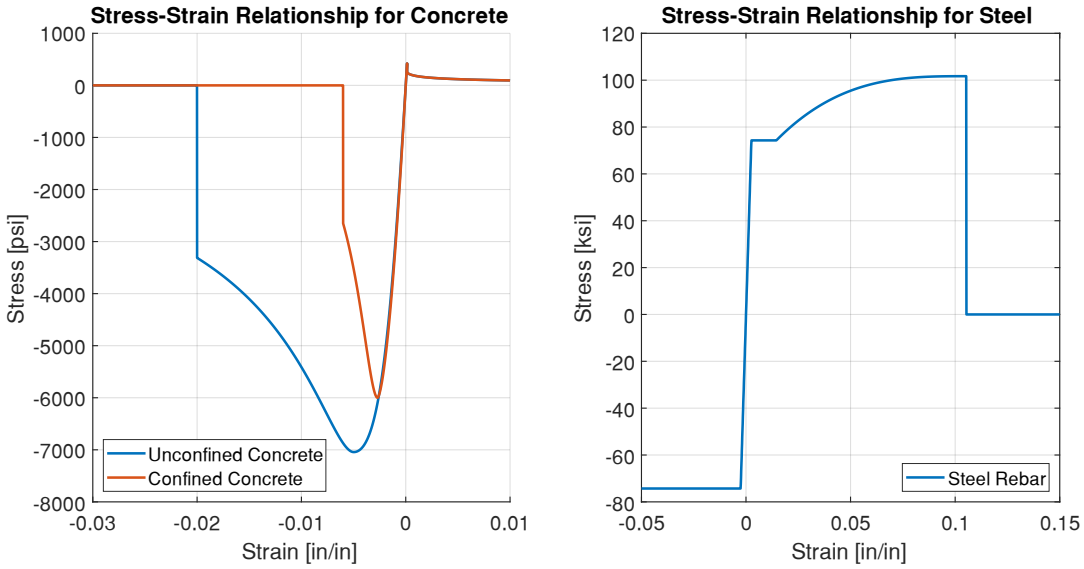


Figure 3 Constitutive Models for Confined & Unconfined Concrete and Steel Rebar

The strain history of the extreme fibers of the concrete and steel are shown in Figure 4 for the curvature range of a typical Mφ curve. For this report, the end condition of the Mφ curve happens when either the extreme compressive fiber of concrete reaches a strain of -0.2% or the extreme tensile steel fibers reaches a strain of 6%. For most cases, the concrete reached this end state first, while the strain in the steel is also near this end state. In this model column, -0.2% was reached by the extreme concrete fiber.

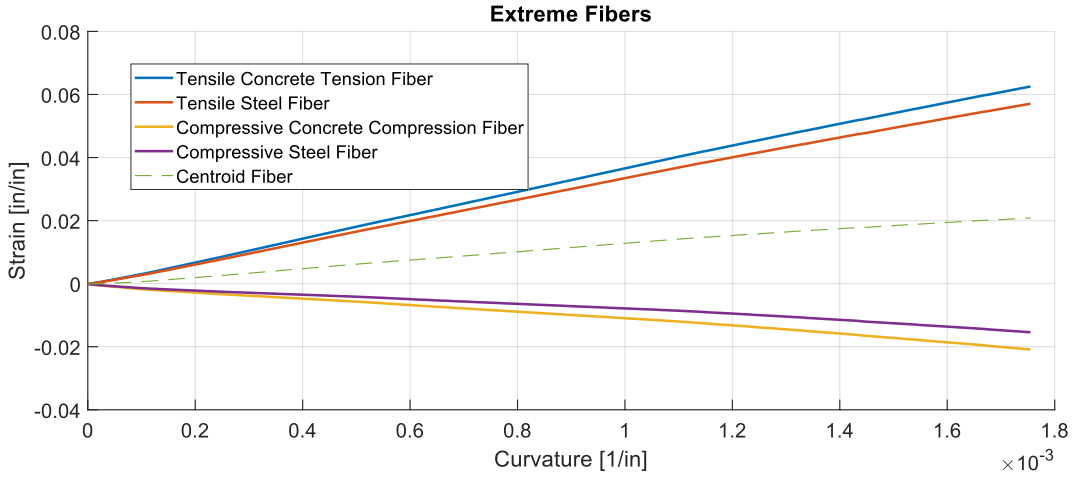


Figure 4 Strain in Extreme Fibers of Column per Curvature

One of the ways to verify if the constitutive model was behaving properly was to examine the strain history and stress history against the curvature. In Figure 5 the full strain history of the 5 fibers of steel is given. In Figure 6 a more zoomed-in strain history is shown to find the curvature when the fibers reached yield strain. In Figure 7, the yield curvatures show exact matches with the stress history of the fibers. These figures show that the steel rebar behaved as expected. Note the behavior of fiber 4 of steel.

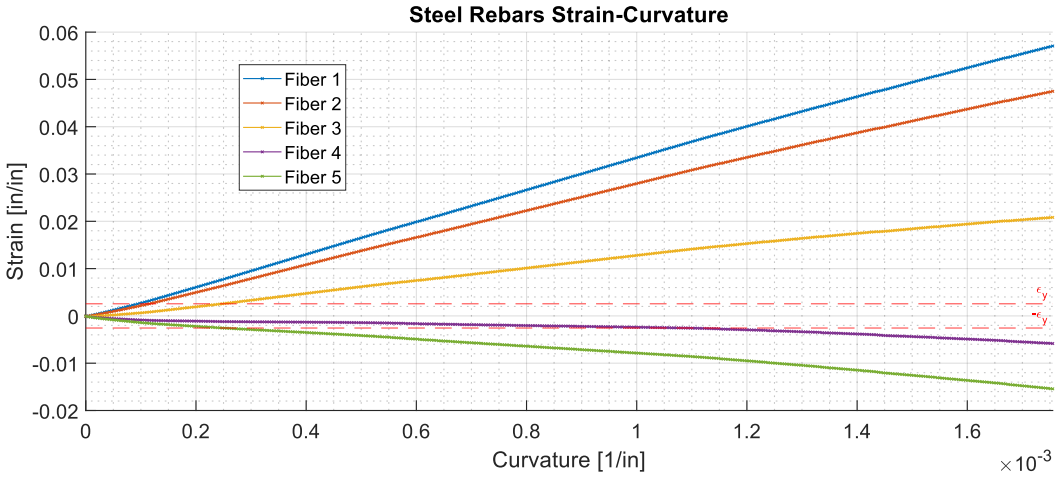


Figure 5 Steel Rebar Strain per Curvature

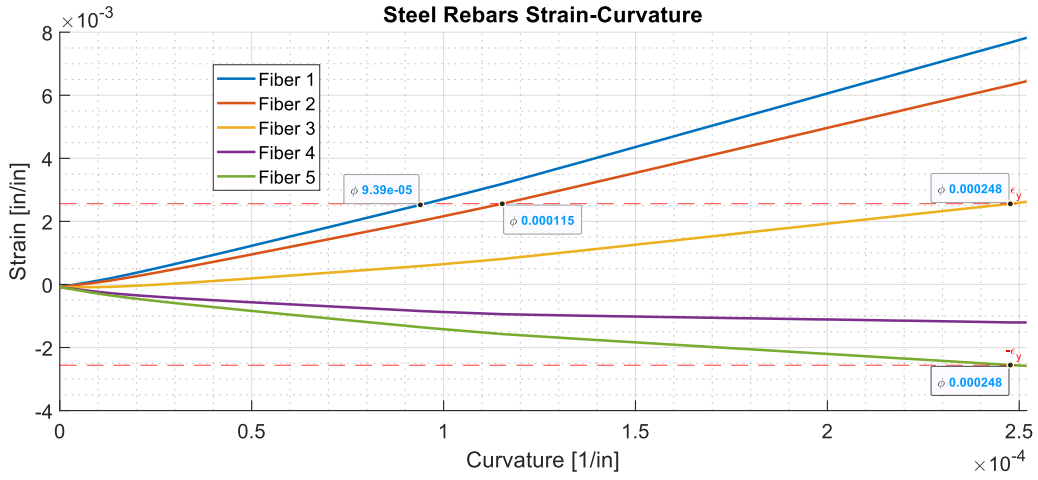


Figure 6 Steel Rebar Strain per Curvature - Finding Yeilding Strains

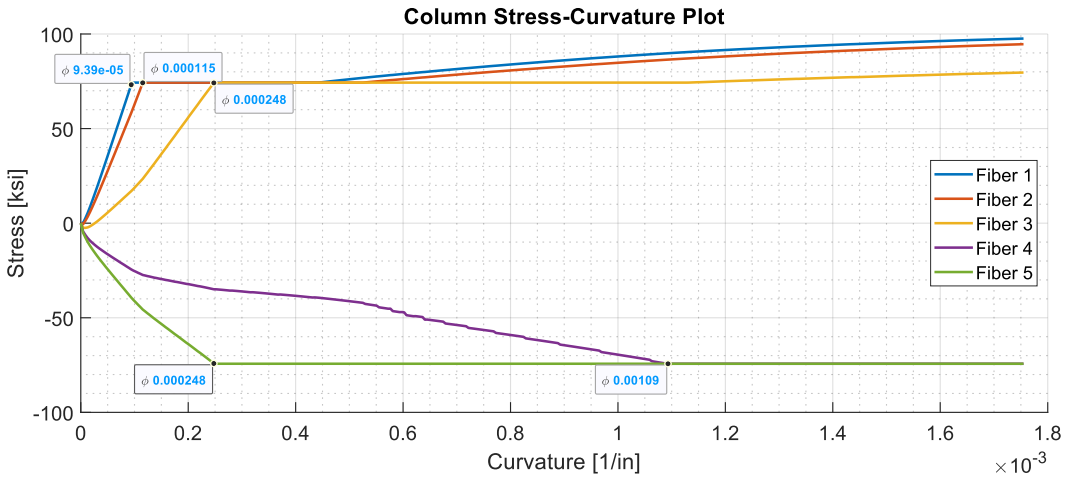


Figure 7 Matching Steel Strains with Steel Rebar Stresses

The concrete strain history is shown in Figure 8. The behavior of the extreme compressive fiber of concrete is used to determine critical points in the section analysis. The first few critical points are shown.

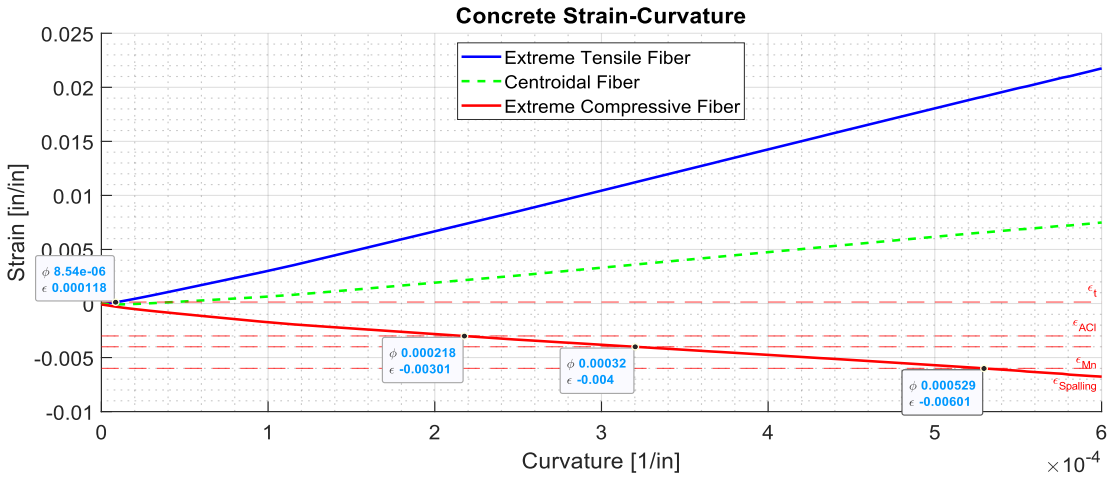


Figure 8 Strain in Concret per Curvature

Part II Section Analysis

The Mφ curve is shown in Figure 9. The critical points are the cracking point, (ϕ_{cr}, M_{cr}), the first yield point (ϕ_y, M_y), the ACI nominal moment $(\phi_{n,ACI}, M_{n,ACI})$, the nominal moment (ϕ_n, M_n), spalling point $(\phi_{spalling}, M_{spalling})$, the overstrength point$(\phi^o, M^o)$, and the ultimate point (ϕ_{ul}, M_{ul}). The cracking point is when the extreme tensile concrete layer reaches a tensile strain ϵ_t. The first yield point is where the extreme tensile longitudinal bar reaches the yield strain ϵ_y or the extreme compressive concrete layer reaches ϵ'_c. The two nominal points differs in that the ACI nominal moment point is located when the extreme compressive concrete layer reaches a strain of -0.3% as oppose nominal moment point of -0.4%. This point is also different from the flexure capacity if calculated following ACI as the code assumes different stress-strain model for concrete and steel and doesn't differentiate between confined and unconfined concrete. The spalling point occurs when the extreme unconfined concrete layer reaches a strain of -0.6%. The over strength point is the maximum point in the Mφ curve. The ultimate point happens when either the extreme tensile steel layer reaches 6% strain, or the extreme compressive concrete layer reaches -0.2%.

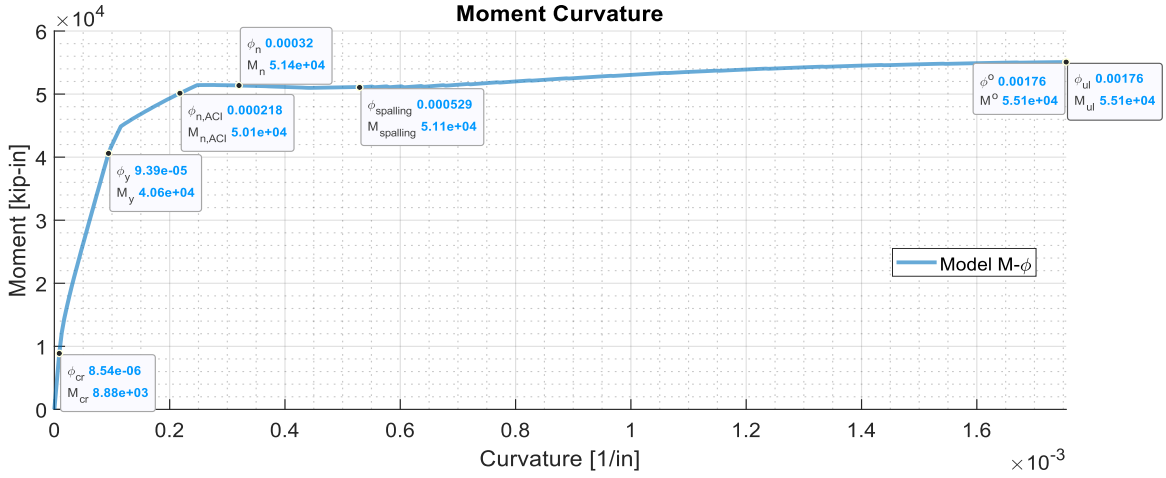


Figure 9 Moment Curvature with points

Table 5 Critical Points Along The Moment Curvature

Critical Points	Values
(ϕ_{cr}, M_{cr})	(8.54e-06, 8.88e+06)
(ϕ'_y, M_y)	(9.39e-05, 4.06e+07)
(ϕ_y, M_y)	(1.19e-04, 5.14e+07)
$(\phi_{n,ACI}, M_{n,ACI})$	(2.18e-04, 5.01e+07)
(ϕ_n, M_n)	(3.20e-04, 5.14e+07)
$(\phi_{spalling}, M_{spalling})$	(5.29e-04, 5.11e+07)
(ϕ^o, M^o)	(1.76e-03, 5.51e+07)
(ϕ_{ul}, M_{ul})	(1.76e-03, 5.51e+07)

The yield point (ϕ_y, M_y) is defined as the point that passes the (ϕ'_y, M_y) and intersects the horizontal line drawn from (ϕ_n, M_n). This is the approximated yield point shown in red in Figure 10. This point is used in the bilinear approximation of the Mφ curve and will be discussed later. The slope of this line is considered the effective flexural stiffness $E_c \cdot I_e$.

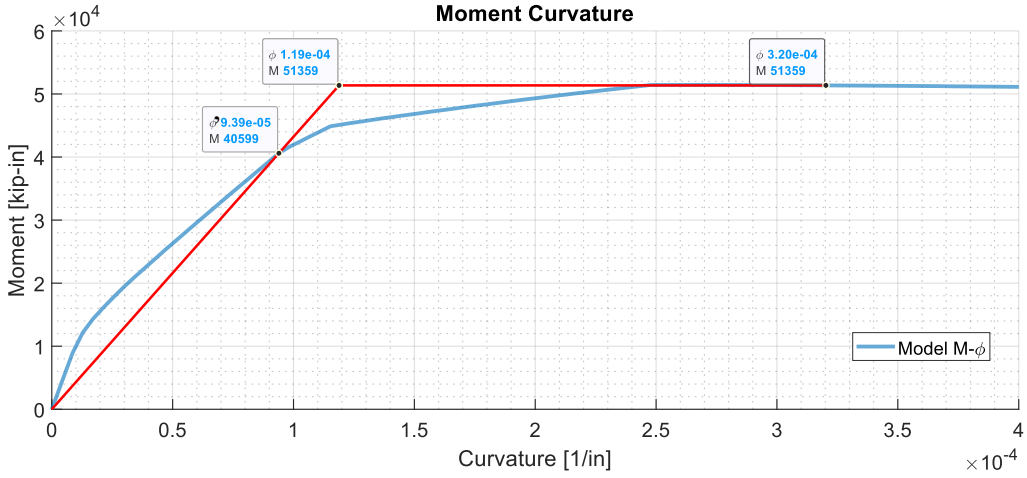


Figure 10 Finding the Yield Point

The flexural rigidity is given as the modulus of elasticity times the moment of inertia. Before the column cracks the concrete has a flexural stiffness equal to the uncracked section moment of inertia times the modulus of elasticity, but once cracking occurs, the moment of inertia changes. In Figure 11, using the transformed moment of inertia, I_t, the linear line estimated a cracking moment of 8,740 kip-in which is only 140 kip-in, or 1.5%, from the actual cracking moment of 8,880 kip-in. Using the gross section moment of inertia, I_g, the cracking moment was 7,230 kip-in which had an error of 22.7% from the model. The slight error of the transformed moment of inertia could come from the method by which the cracking was recorded.

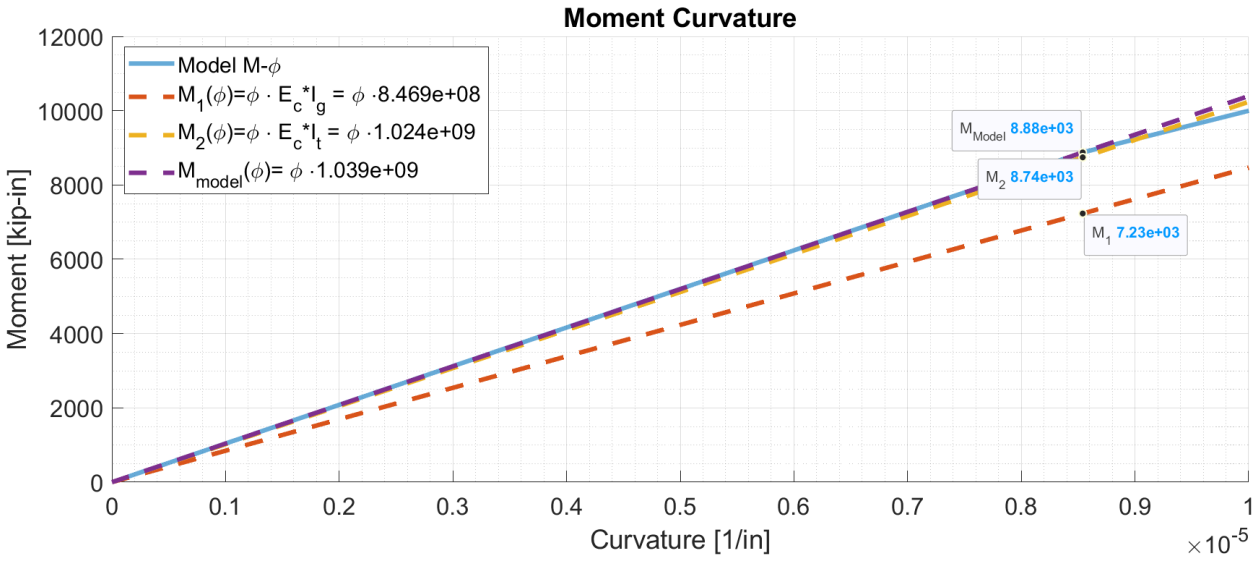


Figure 11 Calculating Flexural Stiffness of the Column

One way to simplify the Mφ curve is to use the critical points to form a bilinear or 4-point linear curve. The bilinear curve is defined as lines joining the points $[(0,0), (\phi_y, M_y), (\phi^o, M^o), (\phi_{ul}, M_{ul})]$. The 4-point linear curve is defined by the points $[(0,0), (\phi_{cr}, M_{cr}), (\phi_y, M_y), (\phi^o, M^o), (\phi_{ul}, M_{ul})]$. In this case, (ϕ^o, M^o) was equal to (ϕ_{ul}, M_{ul}).

The bilinear curve simplifies the tension hardening of the moment curvature curve in the pre-yield region. This is because the initial line of the bilinear curve has a slope of $E_c \cdot I_e$, the effective flexural stiffness, while the initial stiffness of the Mφ curve is actually $E_c \cdot I_t$. As the column cracks, the moment of inertia slowly lowers. Even though this initial behavior is not well captured, its simplicity of this approximation allows it to be used for initial designs. The 4-point linear curve accounts for the tension-stiffening phenomena by starting with the initial flexural stiffness before cracking before jumping up to the yield point.

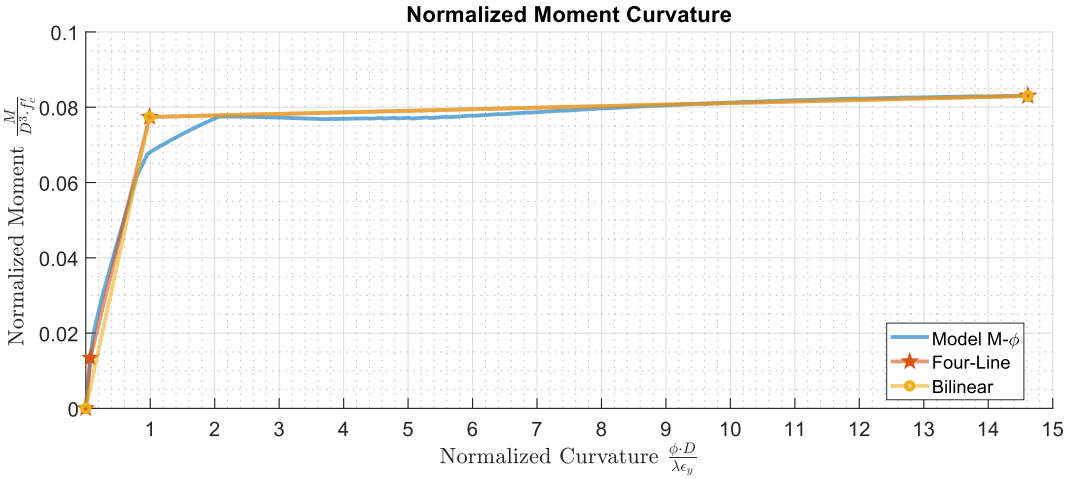


Figure 12 Normalized Moment Curvature with Bilinear & 4-Point Approximation

The effective flexural stiffness, $E_c \cdot I_e$, is calculated as the slope of the line through the origin and (ϕ_y, M_y). This was found to be $432 \cdot 10^6$ kip-in^2. The flexural stiffness of the beam calculated as $E_c \cdot I_g$ was $8.47 \cdot 10^6$ kip-in^2. The stiffness modifier is the ratio $\frac{E_c \cdot I_e}{E_c \cdot I_g}$ and was found to be 0.510. This is in line with *ACI 318-19 Code* 6.6.3.1.2 "For factored lateral load analysis. it shall be permitted to assume $\boldsymbol{I = 0.5 I_g}$ for all members or to calculate $\boldsymbol{I}$ by a more detailed analysis, considering the effective stiffness of all members under the loading conditions." This means that the effective flexural stiffness can be found as one half times the initial flexural rigidity as verified with model. In table 6.6.3.1.1(a), the effective moment of inertia of a column member can be taken as $I_e = 0.7 * I_g$ or as in 6.6.3.1.1(b), the effective moment of inertia is a range from $0.35 I_g \leq I \leq 0.875 I_g$, where I is a function of multiple variables.

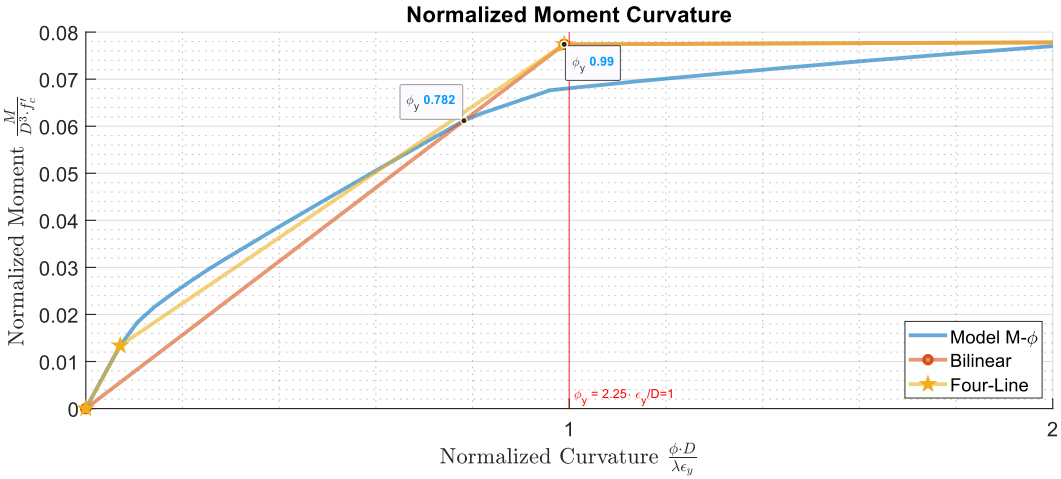


Figure 13 Comparing Priestley Approximation for Yield Curvature

Priestley offers an approximation of the yield point as $\phi_y = \frac{\lambda \epsilon_y}{D}$ where λ is a geometric parameter of the cross section. This parameter is 2.25 for circular cross section and is shown to be an excellent predictor for the yield point. As shown in the normalize Mφ curve above, the yield point found by this λ factor is 1% off from the actual calculated yield point.

The Mφ curve can be also normalized such that the moment and curvatures become dimensionless. This has been shown a few times now. The normalization transformation can be given as

$$M_{normalize} = \frac{M}{f_c' \cdot D^3} \qquad \phi_{normalized} = \frac{\phi D}{\epsilon_y}$$

This allows for plotting and comparing test results of different columns using similar measures of curvature and moment values. For the report, a column with half the size of the original column was tested and the results are shown in Figure 14. The scaled column had the same axial load ratio, same reinforcement ratio, and volumetric confinement ratio. With a normalized moment curvature curve, the plots are, by inspection, on top of each other. If the two curves were not scaled, as shown in Figure 15, the two graphs would show dramatically different behaviors. The normalized moment curvature allows the comparison of two different columns based on their responses rather than actual absolute magnitudes.

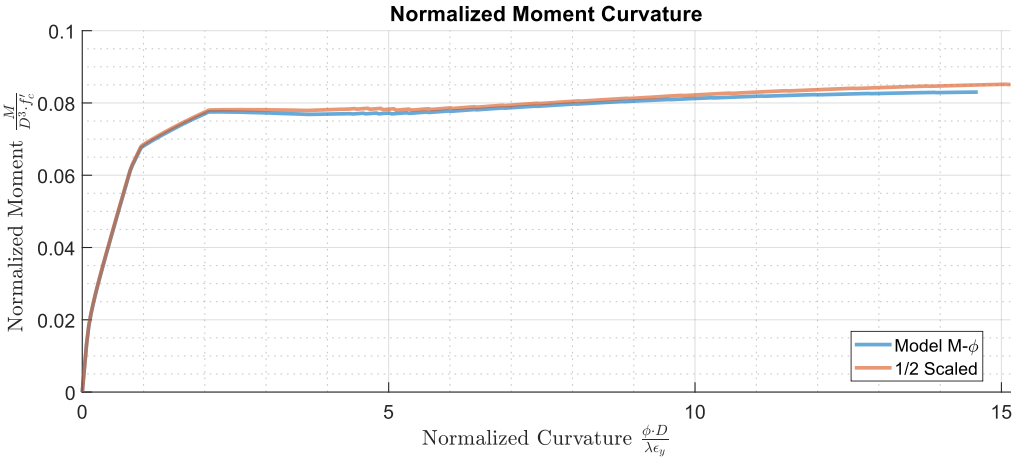


Figure 14 Comparison of Moment Curvatures with Normalized Moments and Curvatures

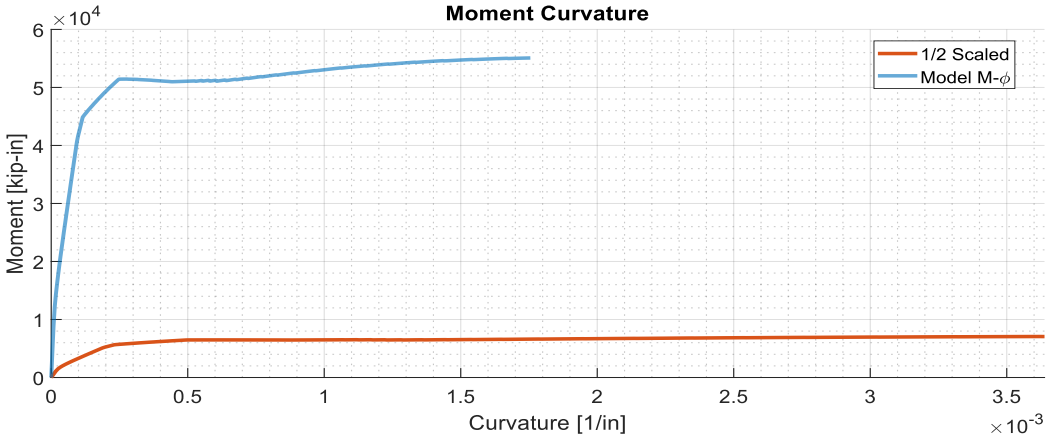


Figure 15 Comparison of Two Moment Curvatures Curves without Normalization

In Figure 16 the moment-curvature time history of the column are given for three ground motions. Using the monotonic Mφ curve as an envelope, the cyclilc moment-curvature behavior seems to be well captured. There are slight differenced in the initial stiffness predicted by the model which could be from errors in the sensors. The moment curvature developed above was flipped into the 3rd quadrant for the graph.

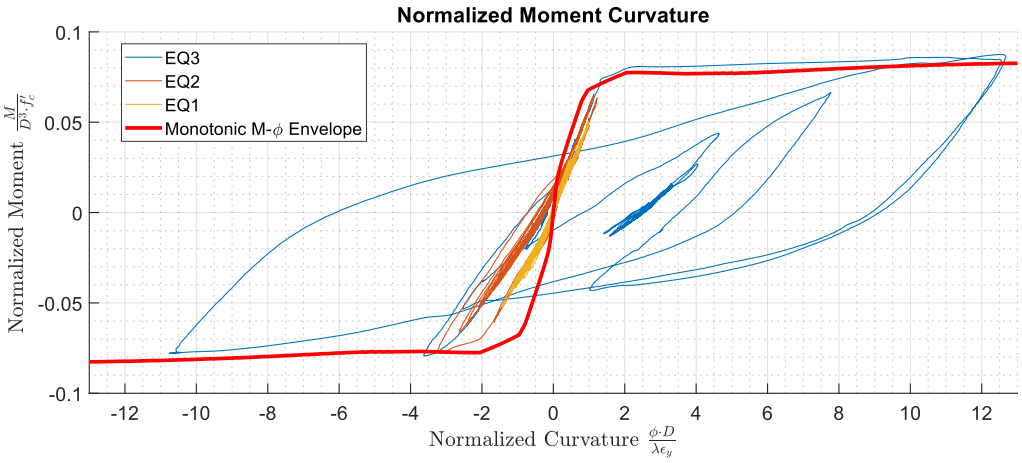


Figure 16 Monotonic Moment Curvature as an Envelope

The longitudinal reinforcement was increased by two and the resulting moment curvature curve is shown in Figure 17. The critical points are tabulated in Table 6. The normalized ϕ_y for both curves are less than 1% difference for each other and from the approximation given by Priestley's equation of $\phi_y = \frac{2.25\epsilon_y}{D}$ which means the lambda factor of 2.25 is a good match for the given models so far,

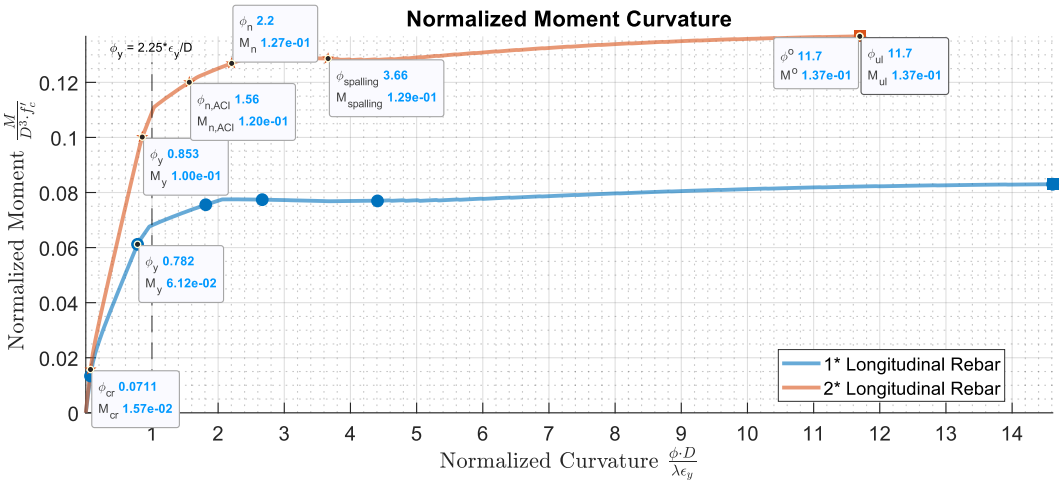


Figure 17 Normalized Moment Curvature Comparison of Twice the Longitudinal Rebar

Table 6 Comparing Critical Points Along The Normalized Moment Curvature

Critical Points	Normal Column	Modified Column
(ϕ_{cr}, M_{cr})	(0.07, 0.0134)	(0.07, 0.0134)
(ϕ'_y, M_y)	(0.78, 0.0612)	(0.78, 0.0615)
(ϕ_y, M_y)	(0.99, 0.0774)	(0.99, 0.0781)
$(\phi_{n,ACI}, M_{n,ACI})$	(1.81, 0.0756)	(1.78, 0.0757)
(ϕ_n, M_n)	(2.67, 0.0774)	(2.67, 0.0781)
$(\phi_{spalling}, M_{spalling})$	(4.41, 0.0769)	(4.48, 0.0783)
(ϕ^o, M^o)	(14.61, 0.0830)	(15.08, 0.0852)
(ϕ_{ul}, M_{ul})	(14.61, 0.0830)	(15.15, 0.0851)

For the original column, the nominal moment capacity was 0.0774 kip-in, the overstrength moment capacity was 0.083 kip-in, and the ratio between them was 0.93. For the modified column, the nominal moment capacity was 0.127 kip-in, the overstrength moment capacity was 0.137 kip-in, and the ratio between them was 0.93. Between the two columns, the nominal moment capacity of the modified column was 1.64 times greater. Between the two columns, the overstrength moment capacity of the modified column was 1.65 times greater. The section curvature ductility of original column is 14.61 while the section curvature ductility of modified column is 11.70. The flexural stiffness ratio of original column is 0.51 while the flexural stiffness ratio of second column is 0.77.

With increased longitudinal reinforcement ratio, the ratio between the nominal moment capacity and the overstrength moment capacity stayed the same; it stayed at 0.93. The increase in longitudinal reinforcement ratio increased the overstrength moment capacity but it was not proportional. By adding more longitudinal rebar, the column is becoming more 'elastic', such that the response goes from being more ductile and lower overstrength moment capacity to a higher overstrength moment capacity and more

brittle behavior. This is shown with the decreased ultimate curvature ductility between the two columns. There is a tradeoff between being ductile and having a high moment capacity. The stiffness modifier increased with the increase longitudinal rebar ratio since adding more bars would make the column stiffer.

In the portion of the report the column's applied load was modified in order to show the effects of increased axial load ratio on the moment curvature curve. The axial load was increased by 6, 9 and 12 times.

The increase axial load increased the curvature at which the column cracks. One explanation is that the axial force provides compressive strain which means the extreme fibers take more curvature to reach the tensile cracking strain which delays cracking. It is also shown that the yield curvature predicted by Priestley's equation has a larger error for higher axial load columns. As the axial load ratio increased, the yield point defined by the $\lambda = 2.25$ slowed drifted away from the ideal line as shown in Figure 18. This means that the axial load does play a role in calculating the yield curvature. When comparing the nominal moment capacity of the columns, the trend increases until a certain point before reversing. This occurs because the column has reached the balanced point in a virtual axial-moment diagram and went backwards in the upper region past this point. The ultimate moment of each curve in the normalized space is relatively similar in between 0.0830-0.0989 as shown in Figure 19 but which very different curvatures.

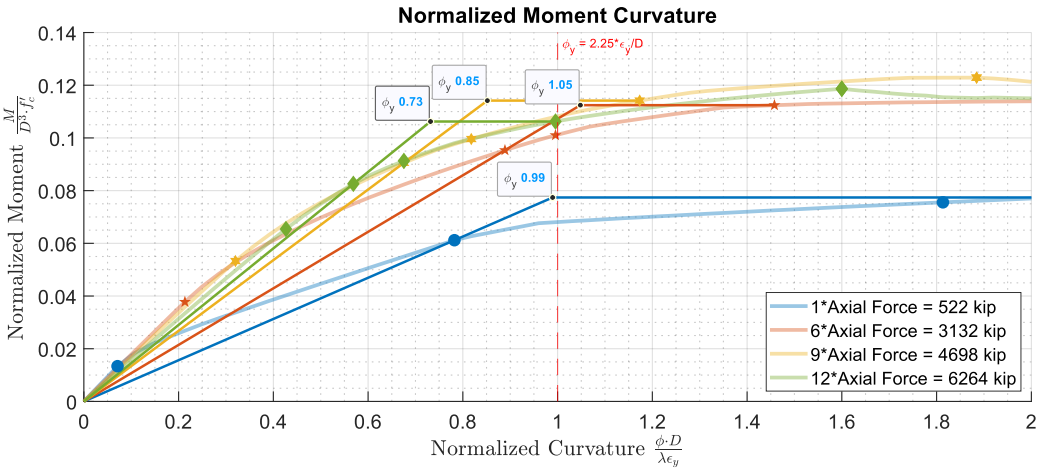


Figure 18 Normalized Moment Curvature Comparison of Yeild Point with Different Axial Force Ratios

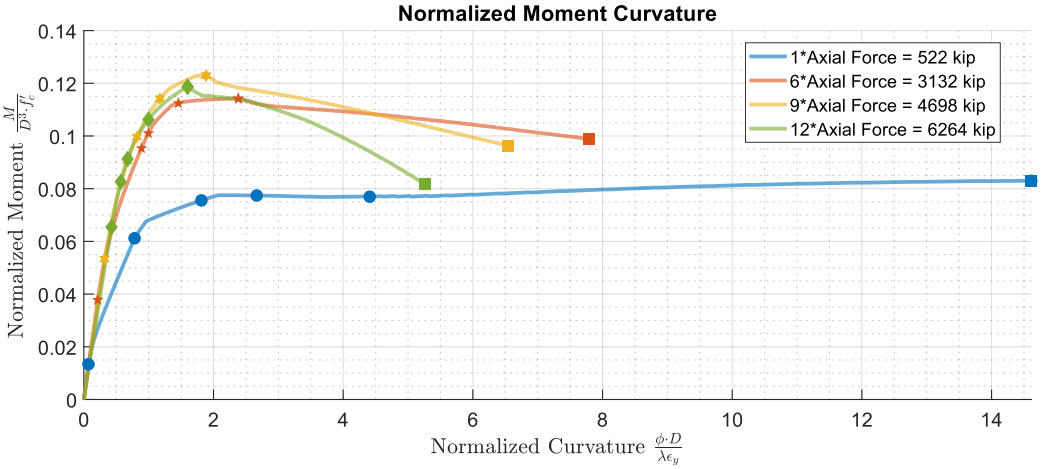


Figure 19 Normalized Moment Curvature for Different Axial Force Ratio

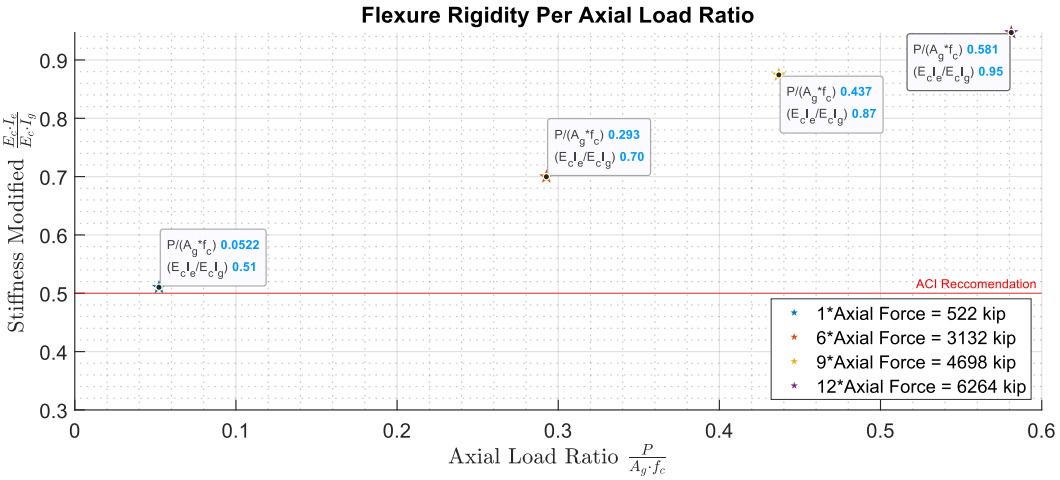


Figure 20 Comparison of Flexural Rigidity with Different Axial Force Ratio

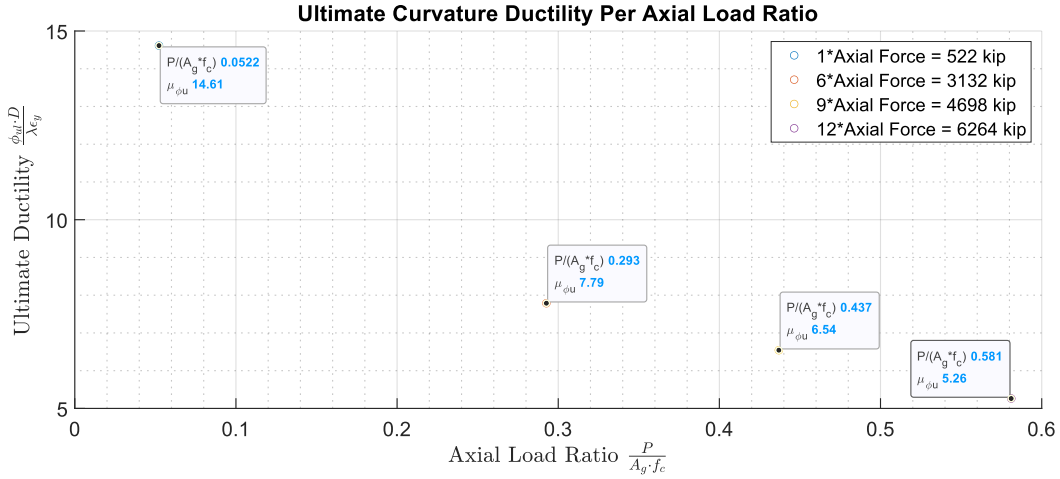


Figure 21 Comparison Ultimate Ductility with Different Axial Force Ratio

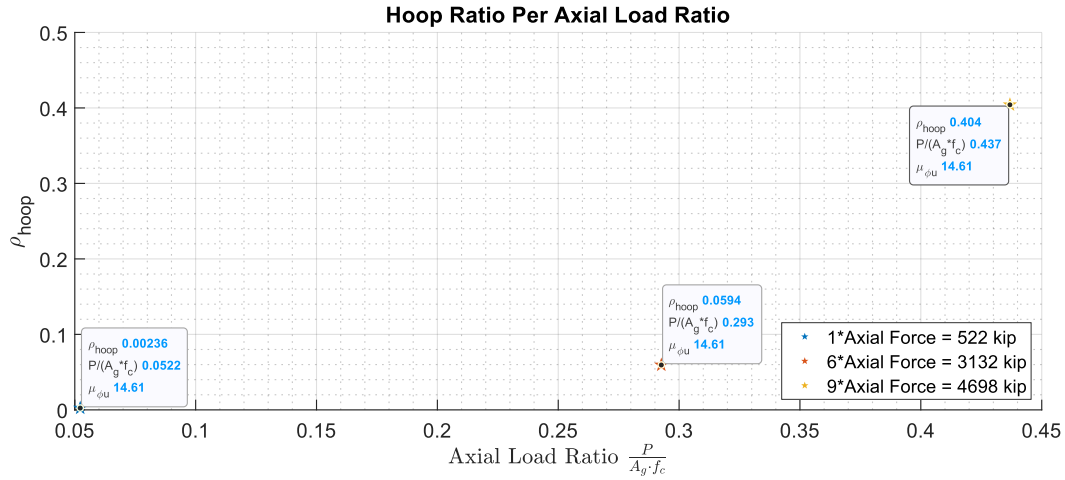


Figure 22 Comparison of Required Hoop Ratio with Different Axial Force Ratio

In Figure 20, the stiffness modifier is plotted against the axial load ratio. This shows that the stiffness modifier increases as the column's axial load increases. In *ACI 318-19 Code*, the recommended value is for the stiffness modifier was 0.5 which is also shown. This value is conservative when higher axial loads are applied to the column, which is reasonable since higher axial load translates to higher stiffness. The ultimate ducitility against the axial load ratio is also plotted which shows a decreasing trend in Figure 21. As the axial load increases the ductility decreases since the stiffness increase. As described before, the increase in stiffness will shift the behavior to a more elastic response, decreasing the maximum curvature and increasing the maximum moment capacity. In Figure 22, the hoop ratio was plotted against the axial load ratio and show an increasing trend if the ultimate ducitility of each column was kept the same. The required hoop ratios increased dramatically in order to obtain the same ductility of the normal column, from 2.36% to 5.94% to 40.4%. For the given column and following. *ACI 318-19* Table 18.7.5.4, the transverse reinforcement in a column with $\leq 0.3A_g f_c'$ axial load ratio and normal-strength concrete should 1.20%. As the axial load increases, the reinforcement ratio is a function of the load as shown in equation (f). Although the model column is not part of a moment frame, the code reflects the trend that increased axial loads needs greater transverse reinforcement to achieve the same ultimate ductility.

Table 18.7.5.4—Transverse reinforcement for columns of special moment frames

Transverse reinforcement	Conditions	Applicable expressions	
A_{sh}/sb_c for rectilinear hoop	$P_u \leq 0.3A_g f_c'$ and $f_c' \leq 10{,}000$ psi	Greater of (a) and (b)	$0.3\left(\frac{A_g}{A_{ch}}-1\right)\frac{f_c'}{f_{yt}}$ (a) $0.09\frac{f_c'}{f_{yt}}$ (b)
	$P_u > 0.3A_g f_c'$ or $f_c' > 10{,}000$ psi	Greatest of (a), (b), and (c)	$0.2k_f k_n \frac{P_u}{f_{yt}A_{ch}}$ (c)
ρ_s for spiral or circular hoop	$P_u \leq 0.3A_g f_c'$ and $f_c' \leq 10{,}000$ psi	Greater of (d) and (e)	$0.45\left(\frac{A_g}{A_{ch}}-1\right)\frac{f_c'}{f_{yt}}$ (d) $0.12\frac{f_c'}{f_{yt}}$ (e)
	$P_u > 0.3A_g f_c'$ or $f_c' > 10{,}000$ psi	Greatest of (d), (e), and (f)	$0.35k_f \frac{P_u}{f_{yt}A_{ch}}$ (f)

Figure 23 ACI 318-19 Table 18.7.5.4 Transverse Reinforcement for Columns of Special Moment Frames

Part III- Element Analysis

In Figure 24, the moment at two instances is plotted on the left and the corresponding curvature is plotted on the right for the column. The moment was assumed to be linear along the height of the column and broken into 8 segments above a plastic hinge length. As shown, when the base of the column reaches the nominal moment capacity, the curvature is of the column is already nonlinear. The development of a plastic hinge at the base of the column allows the column to rotate more and more even for a tiny increase in moment. So, while the moment diagram has a linear profile, the curvature increases dramatically between the two moment levels.

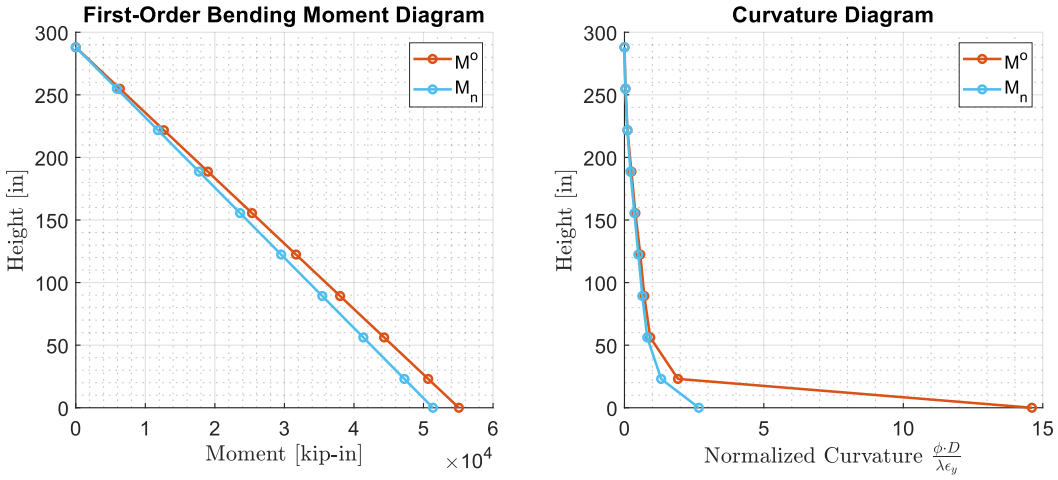


Figure 24 Moment Curvature Along Column

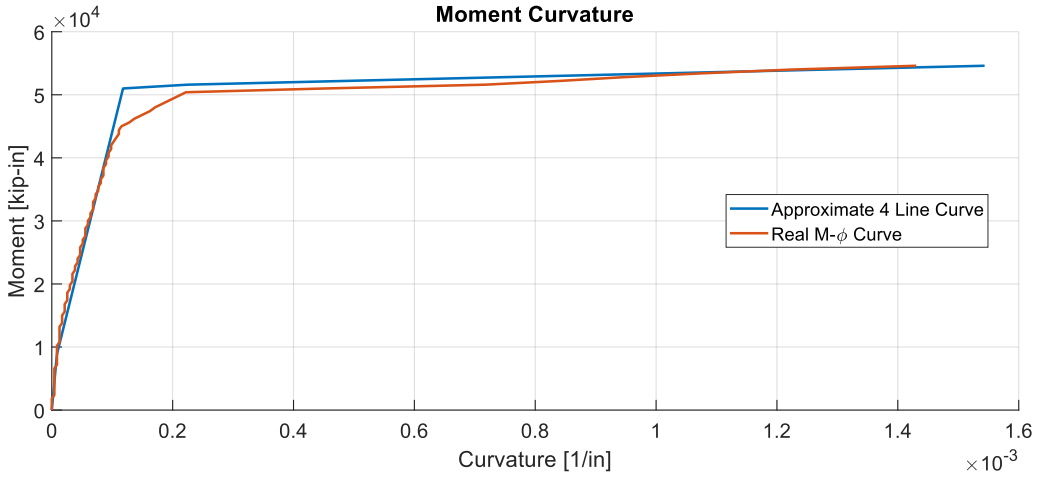


Figure 25 Moment Curvature Approximations for Force Displacement Curves

The force dispalcement curve is plotted by using the second moment area theorem. By discretizing the layers above the plastic hinge and integrating the curvature diagram, the lateral dispalcement can be calculated for the tip of the column. The curvature diagram will be generated by two functions: the actual moment curvature curve and the 4-point linear curve. As shown in Figure 25, for a given moment, a curvature is found; for the actual moment curvature curve function, it seeks out to minimze the error

between the moment given and the moment from the real curve so there are wrinkles in this function rather than a smooth line.

The two force displacement curves for each moment-curavture models are given in Figure 26 amd Figure 27. The *P*Δ effect is descirbed as the moment due to the axial load applied to the deformed geometry of the column. At the ulimate moment step, the equivalent lateral force due to the moment at the 'top' of the column, is ~190 kips while considering *P*Δ, the equivalent lateral load is just ~151 kips. This effect is substaintial as it would cause a 26% difference in the axial load for with and without the load. The 4-point bilinear line was also shown to be a good approximation of the moment curvature curve as the critical points of the curve matched in values.

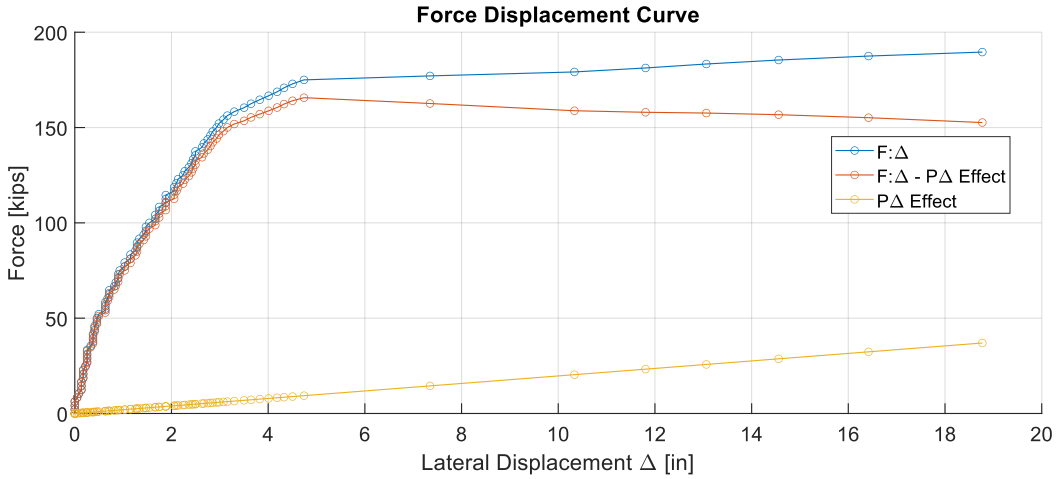


Figure 26 Force Displacement Curve using Real Moment Curvature Curve

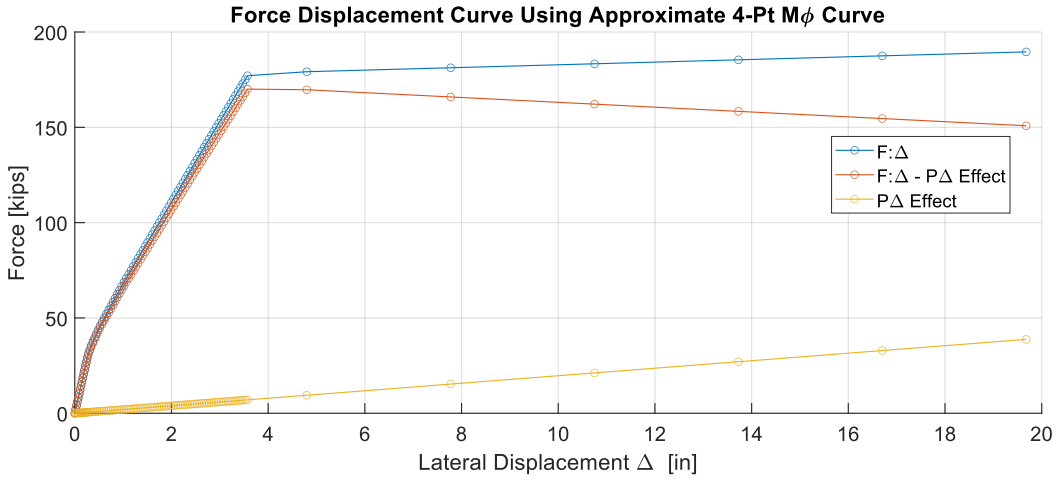


Figure 27 Force Displacement Curve using Approximate Moment Curvature Relationship

In Figure 28, the monotonic force-dispalcement curve was used as the envelope for the measured dynamic lateral force displacement response of the column. As shown the curve matches the initial stiffness of the measurements well as well as enclosing the drift ratio and base shear ratio in the first and third quadrants. When the measurements are above the envelope, this was due to the moment caused by the dispalcement of the mass of the block on top of the column. This dynamic behavior of the system was not considered in the monotonic loading.

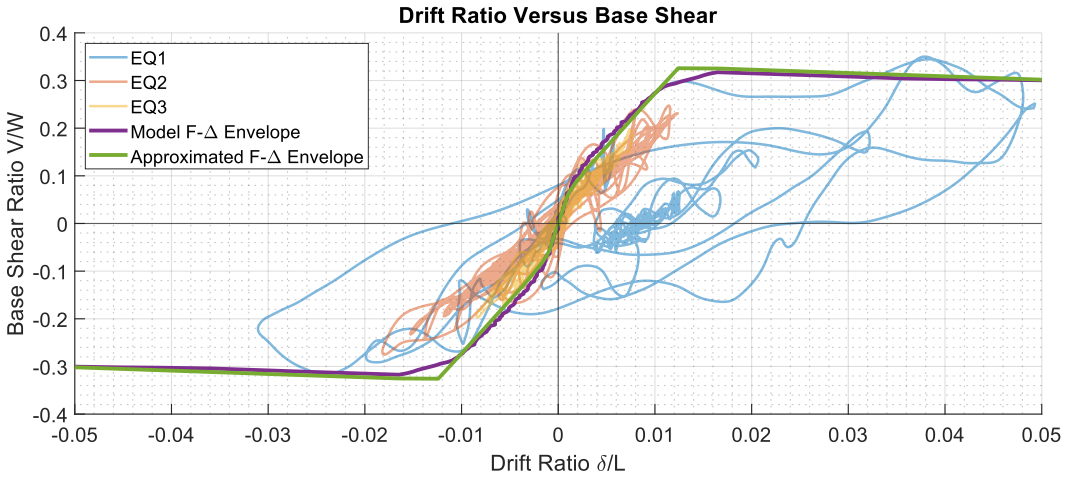


Figure 28 Force Displacement Curve Envelope of Dynamic Testing

The theoretical yield displacemet Δ_y is 2.78 inches, the maxmium displacment demand Δ_y is 14.2 inches, the yield drift ratio is 0.864%, and the maximum curvature demand is 0.1076%. The yield point is using the real moment curvature curve as oppose to the approximate curve. The displacement ducitilty 5.11. If the 4-point line curve was used, the theoretical yield displacement would be $\Delta_y = 3.57$in and the displacment ductility would be 3.97.

References

American Concrete Institute. (2019). *ACI 318-19.*

Priestley, M. (2003). Myths and Fallacies in Earthquake Engineering, Revisited. In *The Ninth Mallet Milne Lecture*. Via Ferrata. Retrieved 11 February 2021, from.

Louis Lin

Sample calculation for confined concrete

***Strength of concrete** $f'_c := 6\ ksi$

Calculate effective confining stress

Clear Spacing between hoops $s' := 6\ in - 2 \cdot 0.625\ in = 4.75\ in$

Hoop diameter $D_s := 44\ in - 0.625\ in = 43.375\ in$

Longitudinal bar yield strength $f_y := 60.4\ ksi$

Area of Long. bars $A_b := 2 \cdot 0.31\ in^2$

Spacing of hoops $s := 6\ in$

Passive confining stress $f'_i := 2 \cdot \frac{f_y \cdot A_b}{s \cdot D_s} = 287.78482\ psi$

Confinement efficiency coefficient $K_e := \left(1 - \frac{s'}{2 \cdot D_s}\right)^2 = 0.89349$

Effective confining stress $f'_{ie} := K_e \cdot f'_i = 257.13229\ psi$

***Ratio**
added it for clarity $\frac{f'_{ie}}{f'_c} = 0.04286$

Calculate confined concrete compressive strength and strain

Confinement coefficient $K_c := 4.1$

Confined concrete compressive strength $f'_{cc} := f'_c + K_c \cdot f'_{ie} = 7.05424\ ksi$

***Ratio**
added it for clarity $\frac{f'_{cc}}{f'_c} = 1.17571$

Strain at unconfined compressive strength $\varepsilon'_c := -0.0027$

Strain at f'_{cc} $\varepsilon'_{cc} := \varepsilon'_c \cdot \left(1 + 20 \cdot \frac{f'_{ie}}{f'_c}\right) = -0.00501$

Calculate normalization term

Longitudinal bar yield strength $f_y := 74.3\ ksi$

***Column length** $L := 24\ ft$

***Column spread of plasticity** $l_{p1} := 0.08 \cdot L = 23.04\ in$

Gage length ratio for unconfine conc. $\lambda_c := \frac{16\ in}{l_{p1}} = 0.69444$

Calculate terms for Modified Mander's model

***Confined concrete secant modulus**
formatted output $E_{secc} := \frac{-f'_{cc}}{\varepsilon'_{cc}} = 1407\ ksi$

***Concrete elastic modulus**
you had Ec in psi! $E_c := 3250\ ksi$

***Ratio** added it for clarity - $\frac{E_{secc}}{E_c} = 0.43288$

Power term $r_{cc} := \frac{1}{1 - \frac{E_{secc}}{E_c}} = 1.76329$

Louis Lin

*** lambda_c = 1 from zero to the peak, ie no gage length dependence in this part of the curve. See Eq. 11**

Modified Mander's Model

$E_c := 3250\ ksi$

$f'_t := 300\ psi \cdot \ln\left(1 + \frac{f'_c}{1800\ psi}\right) = 439.90112\ psi$

$\varepsilon'_t := \frac{f'_t}{E_c} = 0.00014$

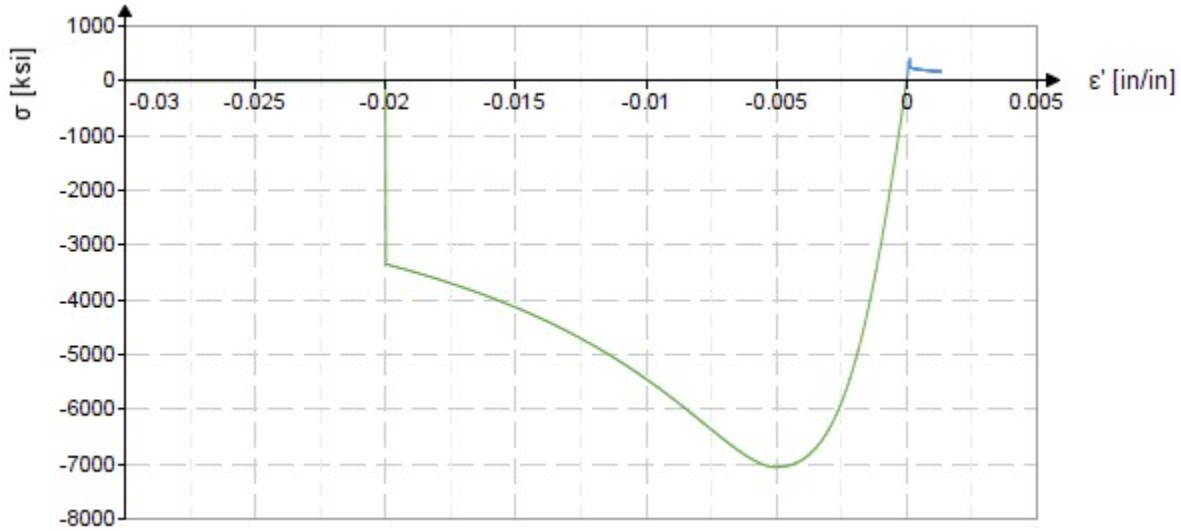
$$f_{confined}(\varepsilon) := \begin{cases} E_c \cdot \varepsilon & \text{if } 0 \le \varepsilon \le \varepsilon'_t \\ \frac{0.7 \cdot f'_t}{1 + \sqrt{500\ \varepsilon}} & \text{else if } \varepsilon > \varepsilon'_t \\ \lambda_c \leftarrow 1 \\ -\frac{\left(1 + \frac{1}{\lambda_c} \cdot \left(\frac{\varepsilon}{\varepsilon'_{cc}} - 1\right)\right)}{\left(r_{cc} - 1 + \left(1 + \left(\frac{1}{\lambda_c}\right) \cdot \left(\frac{\varepsilon}{\varepsilon'_{cc}} - 1\right)\right)^{r_{cc}}\right)} \cdot r_{cc} \cdot f'_{cc} & \text{else if } \varepsilon'_{cc} \le \varepsilon \le 0 \\ -\frac{\left(1 + \frac{1}{\lambda_c} \cdot \left(\frac{\varepsilon}{\varepsilon'_{cc}} - 1\right)\right)}{\left(r_{cc} - 1 + \left(1 + \left(\frac{1}{\lambda_c}\right) \cdot \left(\frac{\varepsilon}{\varepsilon'_{cc}} - 1\right)\right)^{r_{cc}}\right)} \cdot r_{cc} \cdot f'_{cc} & \text{else if } \varepsilon \le \varepsilon'_{cc} \wedge \varepsilon \ge -0.02 \\ 0 & \text{else} \end{cases}$$

$\varepsilon_1 := 0, 0.00001 .. \varepsilon'_t \cdot 10$

$\varepsilon_2 := 0, -0.00001 .. -0.03$

$\mathbf{Y}_1 := f_{confined}(\varepsilon_1)$ $\quad \mathbf{Y}_2 := f_{confined}(\varepsilon_2)$

$\mathbf{X}_1 := \varepsilon_1$ $\quad \mathbf{X}_2 := \varepsilon_2$



Louis Lin

Sample calculation for Unconfined concrete

Strength of concrete $f'_c := 6\ ksi$

Power ratio $\eta_E := 1 + \frac{3600}{f'_c \div psi} = 1.6$

Strain at compressive strength $\varepsilon'_c := -0.0027$

Ultimate strain $\varepsilon'_{cu} := -0.006$

Power ratio $r_c := \frac{1}{1 - 1 \div \eta_E} = 2.66667$

More material properties

Longitudinal bar yield strength $f_y := 74.3\ ksi$

Column length $L := 24\ ft$

Column spread of plasticity $l_{pl} := 0.08 \cdot L = 23.04\ in$

Gage length ratio for unconfine conc. $\lambda_c := \frac{16\ in}{l_{pl}} = 0.69444$

Young's modulus of Concrete $E_c := 3250\ ksi$

Cracking Stress of Concrete $f'_t := 300\ psi \cdot \ln\left(1 + \frac{f'_c}{1800\ psi}\right) = 439.90112\ psi$

Cracking Strain of Concrete $\varepsilon'_t := \frac{f'_t}{E_c} = 0.00014$

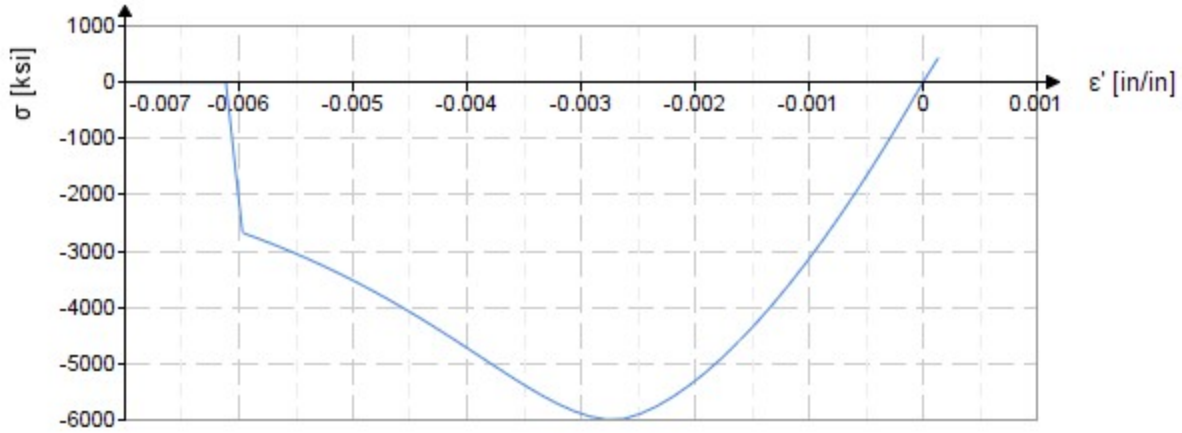
Modified Mander's Model

$$f_{unconfined}(\varepsilon) := \begin{cases} \text{if } 0 \le \varepsilon \le \varepsilon'_t \\ \quad \varepsilon \cdot E_c \\ \text{else if } \varepsilon > \varepsilon'_t \\ \quad \dfrac{0.6 \cdot f'_t}{1 + \sqrt{500\ \varepsilon}} \\ \text{else if } \varepsilon'_c \le \varepsilon \le 0 \\ \quad -\left(1 - \left(1 - \dfrac{\varepsilon}{\varepsilon'_c}\right)^{\eta_E}\right) \cdot f'_c \\ \text{else if } \varepsilon'_{cu} \le \varepsilon \le \varepsilon'_c \\ \quad -\dfrac{\left(1 + \frac{1}{\lambda_c} \cdot \left(\frac{\varepsilon}{\varepsilon'_c} - 1\right)\right)}{\left(r_c - 1 + \left(1 + \left(\frac{1}{\lambda_c}\right) \cdot \left(\frac{\varepsilon}{\varepsilon'_c} - 1\right)\right)^{r_c}\right)} \cdot r_c \cdot f'_c \\ \text{else} \\ \quad 0 \end{cases}$$

$\varepsilon := \varepsilon'_t, -0.00001 .. -0.007$

$$\mathbf{Y} := f_{unconfined}(\varepsilon)$$

$$\mathbf{X} := \varepsilon$$



Louis Lin

Sample calculation for confined concrete

Yield stress of steel

$f_y := 74.3\ \textit{ksi}$

$f_{su} := 101.5\ \textit{ksi}$

$E_s := 29000\ \textit{ksi}$

$\varepsilon_{sh} := 0.0027$

$\varepsilon_{su} := 0.154$

$\varepsilon_y := \frac{f_y}{E_s} = 0.00256$

$P := 3$

Modified Mander's Model

$$f_{steel}(\varepsilon) := \begin{cases} E_s \cdot \varepsilon & \text{if } 0 \le \varepsilon \le \varepsilon_y \\ f_y & \text{else if } \varepsilon_y \le \varepsilon \le \varepsilon_{sh} \\ f_{su} - (f_{su} - f_y) \cdot \left(\frac{\langle \varepsilon_{su} - \varepsilon \rangle}{\langle \varepsilon_{su} - \varepsilon_{sh} \rangle} \right)^P & \text{else if } \varepsilon_{sh} \le \varepsilon \le \varepsilon_{su} \\ E_s \cdot \varepsilon & \text{else if } -\varepsilon_y \le \varepsilon \le 0 \\ -f_y & \text{else if } \varepsilon \le -\varepsilon_y \\ 0 & \text{else} \end{cases}$$

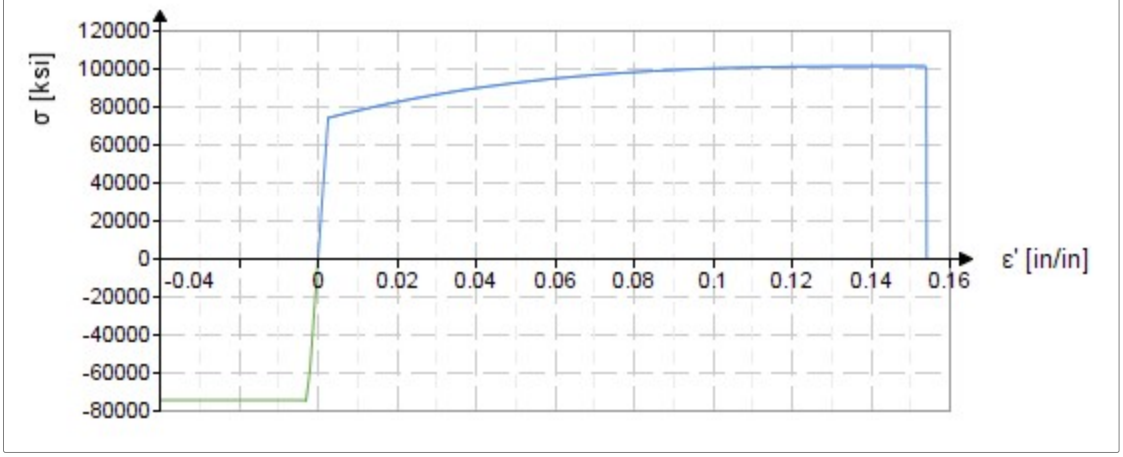
$\varepsilon_1 := 0, 0.0001 .. \varepsilon_{su} + 0.001$

$\varepsilon_2 := 0, -0.001 .. -0.04$

Inputs

$\mathbf{Y}_1 := f_{steel}(\varepsilon_1)$ $\mathbf{Y}_2 := f_{steel}(\varepsilon_2)$

$\mathbf{X}_1 := \varepsilon_1$ $\mathbf{X}_2 := \varepsilon_2$



Appendix B. MATLAB Code

```
%%
addpath('functions');
clear; clc;

%% Regular column
Column1 = GiveMeAColumn("Model M-\phi");
%% Scaled Regular Column down by 0.5
Column2 = GiveMeAColumn("1/2 Scaled",... % name
                        141809,... % Applied Load
                        24, ... % Diameter  [in]
                        [1.1700    1.5600    1.5600    1.5600    1.1700],...% Long steel rebar area [in^2]
                        [1.6750    4.4050   12.0000   19.5950   22.3250],... % Long steel rebar y
                        .705,... % Diameter of Longitudinal Bars [in]
                        0.3125,... % Hoop Diameter [in]
                        3,...% Hoop spacing [in]
                        12,...% Height of Column [ft]
                        1); % Cover [in]

%% Two times the longitudinal rebar
Column3 = GiveMeAColumn('2* Longitudinal Rebar', ... %name
                        522000,... % Applied Weight
                        48,... % Diameter [in]
                        2*[4.68, 6.24, 6.24, 6.24, 4.68]); % Long Steel rebar area [in^2]
%% Increase axial load
Column4 = GiveMeAColumn('6*Axial Force = 3132 kip', 6*522000); % 6 times axial force everything else default
Column5 = GiveMeAColumn('9*Axial Force = 4698 kip', 9*522000); % 9 times axial force everything else default
Column6 = GiveMeAColumn('12*Axial Force = 6264 kip', 12*522000); % 12 times axial force everything else default

%% Increase axial load- keep same ductility, alter the hoop reinforcement ratio
% Change the spacing in order to vary the transverse reinforcement
hoop.spacing_2 =  .238;
hoop.spacing_3 =  .035;
hoop.spacing_4 =  .01;
Column7 = GiveMeAColumn('6*Axial Force = 3132 kip', 6*522000, 48, [4.68, 6.24, 6.24, 6.24, 4.68], [3.35, 8.81, 24., 39.19, 44.65],1.410, 0.625, ↙
hoop.spacing_2 );
Column8 = GiveMeAColumn('9*Axial Force = 4698 kip', 9*522000, 48, [4.68, 6.24, 6.24, 6.24, 4.68], [3.35, 8.81, 24., 39.19, 44.65],1.410, 0.625, ↙
hoop.spacing_3  );
Column9 = GiveMeAColumn('12*Axial Force = 6264 kip', 12*522000, 48, [4.68, 6.24, 6.24, 6.24, 4.68], [3.35, 8.81, 24., 39.19, 44.65],1.410, ↙
0.625,hoop.spacing_4 );


%% Columm Function
function Column = GiveMeAColumn(name, Axial_Load, Diameter,Steel_Area, Steel_Y, long_db, hoop_db, hoop_spacing, height, cover)
% Returns a Coloumn structure couple of main data structures:
%   Confined concrete; used for the constitutive functions
%   Unconfined concrete; used for the constitutive functions
%   Long; (for the longitudinal rebars)
%   Hoop; (for the hoop rebar)
%   Section; (with all section properties.
%   id; array of indices for the critical points of the curve
%   A bunch  of calcualted properties like Eclg and Ecle

% The moment curvature function is called in the body

  arguments
      % Allows me to set the defautl column values
      % All of them are given
```

```
55      name ="Column"
56      Axial_Load = 522000
57      Diameter = 48
58      Steel_Area = [4.68, 6.24, 6.24, 6.24, 4.68]
59      Steel_Y = [3.35, 8.81, 24., 39.19, 44.65]
60      long_db = 1.410
61      hoop_db = 0.625
62      hoop_spacing = 6
63      height = 24
64      cover = 2
65  end
66  % Cross Section Geometry
67  Column.name = name; % Name the column for plotting purposes
68  Column.Section.diameter = Diameter; % in ;
69  Column.Section.area = Column.Section.diameter^2*pi()/4; % in^2 ;
70  Column.Section.cover = cover; % in ;
71  Column.Section.diameter_confined = Diameter-2*Column.Section.cover ; % in
72  Column.Section.no_of_fibers = 100; % #
73  Column.Section.fiber_width = Diameter/Column.Section.no_of_fibers; % in
74  Column.Section.centroid_y = Column.Section.diameter/2; % in
75  Column.Section.lamba = 2.25; % Priestley's lambda value to find yield curv.
76
77  Column.height = height*12; % in; given in feet
78  Column.density = 150*12^-3; % pcf
79  Column.volume = Column.Section.area*Column.height;  % in^3
80  Column.self_weight = Column.density * Column.volume; %lbf
81  Column.applied_force = Axial_Load; % lbf
82  Column.total_axial_load = -(Column.self_weight + Column.applied_force); % lbf
83
84  % Hoop reinforcement properties
85  Hoop.fy = 60400; % psi; Hoops
86  Hoop.db = hoop_db; %in; Diameter of hoops
87  Hoop.spacing = hoop_spacing; % in; spacing
88  Hoop.clear_spacing = Hoop.spacing-2*Hoop.db; % in; clear vertical spacing between hoops
89  Hoop.diameter = Column.Section.diameter - 2*cover - Hoop.db; % Hoop diameter
90  Hoop.area = 2*hoop_db^2*pi()/4; % in^2; area of hoop
91  Hoop.rho = Hoop.area/Hoop.diameter/Hoop.spacing; % Steel ratio of the transverse rebar
92  Column.Hoop = Hoop; % Saves the hoop
93
94  % Longitudinal reinforcement properties
95  Long.fy = 74300; % psi; Longitudinal
96  Long.db = long_db; % Longitudinal rebar diameter
97  Long.Es = 29000000; % psi
98  Long.P = 3; % Mander's model
99  Long.esh = 0.0145; % Strain hardening strain
100  Long.esu = 0.1054; % Maximum longitudinal strain of steel
101  Long.fsu = 101700; % psi
102  Long.ey = Long.fy/Long.Es; % Yeilding strain
103  Column.Long = Long;  % Saves the longitudinal rebar
104
105  % Concrete data structure
106  Concrete.fc = 6000; % psi; Compressive strength of concrete
107  Concrete.Ec = 3250000; % ksi; Young's Modulus
108  Concrete.ec = -0.0027; % Maximum compressive strain
109  Concrete.lp1 = 0.08*Column.height; % Length of the plastic hinge region
110  Concrete.lp2 = 0.15*Long.db*Long.fy/1000; % Strain penetration length
111  Concrete.lambda_c = 16/ Concrete.lp1; % Normalization factor
```

```
112     Concrete.ft = 300*log(1+Concrete.fc/1800); % Cracking stress
113     Concrete.et = Concrete.ft/Concrete.Ec; % Cracking strain
114
115     [Unconfined_C, Confined_C] = deal(Concrete); % Both concrete have base values
116     % Unconfined Concrete modifications
117     Unconfined_C.ecu = -0.006; % Maximum compressive strain
118     Column.Unconfined_Concrete = Unconfined_C;
119     % Confined Concrete modifications
120     Confined_C.ecu = -0.02; % Maximum compressive strain of confined concrete
121     Confined_C.Ke = (1-Hoop.clear_spacing/2/Hoop.diameter)^2; % Confinement efficiency coefficient
122     Confined_C.fl = 2*Hoop.fy*Hoop.area/Hoop.spacing/Hoop.diameter;% Passive confining stress
123     Confined_C.fie = Confined_C.Ke * Confined_C.fl; % Effective confining stress
124     Confined_C.Kc = 4.1;
125     Confined_C.fcc = Confined_C.fc + Confined_C.Kc*Confined_C.fie;
126     Confined_C.ecc = Confined_C.ec*(1+20*Confined_C.fie/Confined_C.fc);
127     Confined_C.E_secc = -Confined_C.fcc/Confined_C.ecc;
128     Confined_C.r_cc = 1/(1-Confined_C.E_secc/Confined_C.Ec);
129     Column.Confined_Concrete = Confined_C;
130     % Discretize the section
131     % Reinforcement
132     Column.Section.steel_y = Steel_Y; % height for each later
133     Column.Section.steel_fiber_area = Steel_Area; % steel area per layer
134     Column.Section.transformed_area = Column.Section.area +(Long.Es/Concrete.Ec - 1)*sum(Column.Section.steel_fiber_area); %↙
transformed area
135     [Column.Section.conc_fiber_y, Column.Section.unconfined_fiber_area, Column.Section.confined_fiber_area] = areas_circle(Column.↙
Section); % discretize the system
136     Column = MomentCurvature(Column); % RUNS THE MOMENT CURVATURE!!!!!!!!
137
138     Column.norm_M = 1/(Column.Section.diameter^3*Column.Confined_Concrete.fc); % Moment normillization factor
139     Column.norm_P = Column.Section.diameter/Column.Section.lamba/Column.Long.ey; % Curvature normillization factor
140     Column.Curvature_ductility = Column.Curvature(end)*Column.Section.diameter/Column.Section.lamba/Column.Long.ey; % Curvature↙
ductility
141     Column.Normalized_P = Column.Curvature * Column.norm_P; % normalized curvature
142     Column.Normalized_M = Column.Moment * Column.norm_M;% normalized moment
143
144
145     [~, id.et] = min(abs(Column.Confined_Concrete.et- Column.Concrete_strain(:,2))); % index for cracking
146     [~, id.fyc] = min(abs(Column.Confined_Concrete.ec - Column.Concrete_strain(:,Column.Section.no_of_fibers))); % index for yeilding in↙
concrete
147     [~, id.fys] = min(abs(Column.Long.ey - Column.Steel_strain(:,1))); % index for yeilding in rebar
148     id.fy = min(id.fyc,id.fys); % takes the minimum
149     [~, id.ACI] = min(abs(-0.003- Column.Concrete_strain(:,Column.Section.no_of_fibers))); % index for -.3% strain
150     [~, id.Mn] = min(abs(-0.004- Column.Concrete_strain(:,Column.Section.no_of_fibers))); % index for -.4% strain
151     [~, id.Spalling] = min(abs(-0.006- Column.Concrete_strain(:,Column.Section.no_of_fibers))); % index for -.6% strain
152     [~, id.Max] = max(Column.Moment); % index for overstrength moment
153     id.End = length(Column.Moment); % index for ultimate strain
154
155     Column.id = id; % Saves all of those ids
156
157     % determine the end marker
158     if abs(Column.Steel_strain(end,1) - 0.06) <= 0.001
159         Column.end_marker = "d";
160     elseif abs(Column.Concrete_strain(end,end) + 0.02) <=0.001
161         Column.end_marker = "s";
162     end
163
164     Column.Mcr = Column.Normalized_M(id.et);
```

```
    Column.My = Column.Normalized_M(id.fy);
    Column.MACI = Column.Normalized_M(id.ACI);
    Column.Mn = Column.Normalized_M(id.Mn);
    Column.Ms = Column.Normalized_M(id.Spalling);
    Column.Mo = Column.Normalized_M(id.Max);
    Column.Mul = Column.Normalized_M(id.End);

    Column.Pcr = Column.Normalized_P(id.et);
    Column.Py = Column.Normalized_P(id.fy);
    Column.PACI = Column.Normalized_P(id.ACI);
    Column.Pn = Column.Normalized_P(id.Mn);
    Column.Ps = Column.Normalized_P(id.Spalling);
    Column.Po = Column.Normalized_P(id.Max);
    Column.Pul = Column.Normalized_P(id.End);

    Column.Section.Ig = Column.Section.diameter^4 * pi()/64;
    Column.Section.n = Column.Long.Es / Column.Confined_Concrete.Ec;
    Column.Section.It = Column.Section.Ig + (Column.Section.n-1)*sum(Column.Section.steel_fiber_area.*(Column.Section.centroid_y - ↙
Column.Section.steel_y).^2);
    Column.Ec_It = Column.Confined_Concrete.Ec * Column.Section.It;
    Column.Ec_Ig = Column.Confined_Concrete.Ec * Column.Section.Ig;
    Column.Ec_Ie = Column.Moment(id.fy)/Column.Curvature(id.fy);
    Column.Ec_Ie_n = Column.My/Column.Py;

    Column.Axial_load_ratio = -Column.total_axial_load/(Column.Section.area*Column.Confined_Concrete.fc);
end
```

```
function [Column] = MomentCurvature(Column)
%% Begin Solutions
n = 25; % As chosen by the class
d_curv = 2*Column.Long.fy/Column.Long.Es/Column.Section.diameter/n; % Curvature step
tol = 0.001; % Tolerance to move onto the next step
max_i = 1000; % Max iteration for each curvature step

i = 1; % Step counter
Column.Curvature(i) = 0; % Initialize
strain = Column.total_axial_load/Column.Confined_Concrete.Ec/Column.Section.transformed_area ; % P/E/At; Initial Strain at Centroid
Column.Concrete_strain(i,:) = strain*ones(Column.Section.no_of_fibers,1); % Initialize
Column.Steel_strain(i,:) = strain*ones(numel(Column.Section.steel_y),1); % Initialize

i = 2; % start at step 2 since we're defining initial conditions
% Shorthands
A_cc = Column.Section.confined_fiber_area;
A_uc = Column.Section.unconfined_fiber_area;
A_s = Column.Section.steel_fiber_area;
Y_bar = Column.Section.centroid_y;
Y_conc = Column.Section.conc_fiber_y;
Y_steel = Column.Section.steel_y;

while max(Column.Steel_strain(i-1,:)) <= 0.06 && min(Column.Concrete_strain(i-1,2:end-1)) >= Column.Confined_Concrete.ecu
    delta_strain = 1*10^-6; % Initial change in strain
    Column.Curvature(i) = Column.Curvature(i-1) + d_curv; % increase the curvature
    j = 0;
    while j <= max_i
        % Concrete
        Column.Concrete_strain(i,:) = Column.Curvature(i) .* (Y_bar - Y_conc) + strain; % Calcualte the strain in each fiber layer
        Column.Concrete_Confined_stress(i,:) = constitutive_confined_concrete(Column.Concrete_strain(i,:), Column.Confined_Concrete); % Calcualte the stress in each fiber layer
        Column.Concrete_Unconfined_stress(i,:) = constitutive_unconfined_concrete(Column.Concrete_strain(i,:), Column.Unconfined_Concrete); % Calcualte the stress in each fiber layer
        Column.Concrete_Confined_force(i,:) = Column.Concrete_Confined_stress(i,:) .* A_cc';  % Calculate the force in each fiber layer
        Column.Concrete_Unconfined_force(i,:) = Column.Concrete_Unconfined_stress(i,:) .* A_uc';  % Calculate the force in each fiber layer
        Column.Concrete_force(i,:) = Column.Concrete_Confined_force(i,:) + Column.Concrete_Unconfined_force(i,:); % % Calculate the force in each fiber layer
        % Steel
        Column.Steel_strain(i,:) = Column.Curvature(i) * (Y_bar - Y_steel) + strain; % Calcualte the strain in each fiber layer
        Column.Steel_stress(i,:) = constitutive_steel(Column.Steel_strain(i,:), Column.Long);% Calcualte the stress in each fiber layer
        Column.Confined_stress_atSteel(i,:) = constitutive_confined_concrete(min(Column.Steel_strain(i,:),0), Column.Confined_Concrete); % Calcualte the stress in each fiber layer
        Column.Steel_force(i,:) =  (Column.Steel_stress(i,:)-Column.Confined_stress_atSteel(i,:)) .* A_s; % Calculate the force in each fiber layer
        Column.TotalForce(i) = sum(Column.Steel_force(i,:)) + sum(Column.Concrete_force(i,:)); % Calculate force in each fiber layer minus the concrete

        error = (Column.TotalForce(i) - Column.total_axial_load);   % Error is not normalize :/
        if abs(error) <= tol
            Column.centroid_strain(i) = strain; % Save the strain and break
            break
        else
            [delta_strain, strain] = brute_force(error, tol,delta_strain,strain); % Change the strain at the centroid
            j = j + 1; % Iteration counter
        end

    end
    Column.Moment(i) = sum(Column.Concrete_force(i,:) .*(Y_bar- Y_conc')) + sum(Column.Steel_force(i,:) .* (Y_bar-Y_steel)); % Compute the
```

```
moment
53     i = i + 1; % Increase curvature counter
54 end
55 end
56
```

```
%% (PRINTED) Fig 0.1 Confined and unconfined concrete from area_circle
close all; figure(); clc;
bar([Column1.Section.confined_fiber_area,Column1.Section.unconfined_fiber_area],'stacked');
legend(["Confined Concrete","Unconfined Concrete"]);
xlabel('Fiber #'); ylabel('Area in^2');
set(gcf, 'Position',  [0, 800, 1300,500])
print_figure("01 Discretized Concrete Area");

%% (PRINTED) Fig 0.2 Stress-Strain Relationship
for Column = Column1
close all;
figure();
subplot(1,2,1);
hold on; grid on;
ylabel("Stress [psi]"); xlabel("Strain [in/in]");
strain_range = 0.01:-0.00001:-0.03;
[confined_stress, unconfined_stress] = deal(zeros(1,numel(strain_range)));
for i = 1:length(strain_range)
    unconfined_stress(i) = constitutive_unconfined_concrete(strain_range(i), Column.Unconfined_Concrete);
    confined_stress(i) = constitutive_confined_concrete(strain_range(i), Column.Confined_Concrete);
end
plot(strain_range,confined_stress,'LineWidth',2);
plot(strain_range,unconfined_stress,'LineWidth',2);
legend(["Unconfined Concrete","Confined Concrete"],"Location","southwest");
title("Stress-Strain Relationship for Concrete");

subplot(1,2,2);
strain_range = -0.05:0.0001:0.15;
steel_stress = zeros(1,numel(strain_range));
for i = 1:length(strain_range)
    steel_stress(i) = constitutive_steel(strain_range(i), Column.Long);
end
plot(strain_range,steel_stress/1000,'LineWidth',2,'DisplayName','Steel Rebar');
grid on; legend("location","Southeast");
ylabel("Stress [ksi]"); xlabel("Strain [in/in]");
title("Stress-Strain Relationship for Steel");
set(gcf, 'Position',  [0, 800, 1300,600])
end
print_figure("02 Constitutive Models");

%% (PRINTED) Fig 0.3 Extreme Fiber Plots Strain Plots
for Column = Column1
close all;
figure; hold on; title('Extreme Fibers');
plot(Column.Curvature,Column.Concrete_strain(:,1),'-',"DisplayName","Tensile Concrete Tension Fiber",'LineWidth',2)
plot(Column.Curvature,Column.Steel_strain(:,1),'-',"DisplayName","Tensile Steel Fiber","MarkerSize",3, 'LineWidth',2);
plot(Column.Curvature,Column.Concrete_strain(:,end),'-',"DisplayName","Compressive Concrete Compression Fiber",'LineWidth',2);
plot(Column.Curvature,Column.Steel_strain(:,end),'-',"DisplayName","Compressive Steel Fiber", 'LineWidth',2);
plot(Column.Curvature,Column.centroid_strain,'--',"DisplayName","Centroid Fiber","MarkerSize",3);
set(gcf, 'Position',  [0, 800, 1300,500])
grid on; legend('Location','Best');
ylabel("Strain [in/in]"); xlabel("Curvature [1/in]");
end
print_figure("03 Strains of Extreme Fibers");

%% (PRINTED) Fig 0.4 Steel Strain Plots (FULL)
close all;
```

```
figure; hold on; grid minor
ms = 3;
plot(Column.Curvature,Column.Steel_strain(:,1),'x-',"DisplayName","Fiber 1","MarkerSize",ms);
plot(Column.Curvature,Column.Steel_strain(:,2),'x-',"DisplayName","Fiber 2","MarkerSize",ms);
plot(Column.Curvature,Column.Steel_strain(:,3),'x-',"DisplayName","Fiber 3","MarkerSize",ms);
plot(Column.Curvature,Column.Steel_strain(:,4),'x-',"DisplayName","Fiber 4","MarkerSize",ms);
plot(Column.Curvature,Column.Steel_strain(:,5),'x-',"DisplayName","Fiber 5","MarkerSize",ms);
% plot(Column.Curvature,Column.centroid_strain,'--',"DisplayName","Centroid Fiber","MarkerSize",ms);

yline(Column.Long.ey,'r--',"\epsilon_y","HandleVisibility","off",'LabelVerticalAlignment','top');
yline(-Column.Long.ey,'r--',"-\epsilon_y","HandleVisibility","off",'LabelVerticalAlignment','top');
title("Steel Rebars Strain-Curvature");
set(gcf, 'Position',  [0, 800, 1300,500])
grid on; legend("Location","Best");
ylabel("Strain [in/in]"); xlabel("Curvature [1/in]");
xlim([0,inf]);
print_figure("04 Steel Fibers Strain Curvature FULL");

%% (PRINTED) Fig 0.5 Strain Steel Plots (INITIAL) Finding where the Steel Yeilds
close all; clc;
figure; hold on; grid minor;
ms = 3;
set(gca,'DefaultLineLineWidth',2)

P1= plot(Column.Curvature,Column.Steel_strain(:,1),'-',"DisplayName","Fiber 1");
P2 = plot(Column.Curvature,Column.Steel_strain(:,2),'-',"DisplayName","Fiber 2");
P3 = plot(Column.Curvature,Column.Steel_strain(:,3),'-',"DisplayName","Fiber 3");
P4 = plot(Column.Curvature,Column.Steel_strain(:,4),'-',"DisplayName","Fiber 4");
P5 = plot(Column.Curvature,Column.Steel_strain(:,5),'-',"DisplayName","Fiber 5");

[~, id1] = min(abs(Column.Long.ey- Column.Steel_strain(:,1)));
[~, id2] = min(abs(Column.Long.ey- Column.Steel_strain(:,2)));
[~, id3] = min(abs(Column.Long.ey- Column.Steel_strain(:,3)));
[~, id4] = min(abs(-Column.Long.ey - Column.Steel_strain(:,4)));
[~, id5] = min(abs(-Column.Long.ey- Column.Steel_strain(:,5)));

add_dataTip(P1, id1, ["\phi","\epsilon"],'northwest');
add_dataTip(P2, id2, ["\phi","\epsilon"],'southeast');
add_dataTip(P3, id3, ["\phi","\epsilon"],'northwest');
add_dataTip(P4, id4, ["\phi","\epsilon"],'southwest');
add_dataTip(P5, id5, ["\phi","\epsilon"],'southwest');

P1.DataTipTemplate.DataTipRows(2) = [];
P2.DataTipTemplate.DataTipRows(2) = [];
P3.DataTipTemplate.DataTipRows(2) = [];
P4.DataTipTemplate.DataTipRows(2) = [];
P5.DataTipTemplate.DataTipRows(2) = [];

yline(Column.Long.ey,'r--',"\epsilon_y","HandleVisibility","off",'LabelVerticalAlignment','top');
yline(-Column.Long.ey,'r--',"-\epsilon_y","HandleVisibility","off",'LabelVerticalAlignment','top');
title("Steel Rebars Strain-Curvature");
grid on; legend("Location","Best");
ylabel("Strain [in/in]"); xlabel("Curvature [1/in]");
% xlim([0,2.5*10^-4]);
set(gcf, 'Position',  [0, 800, 1300,500])
% print_figure("05 Steel Fibers Strain Curvature ZOOM");

```

```
115 %% (PRINTED) Fig 0.6 Strain Concrete Plot (INITIAL)
116 for Column = Column1
117 close all; clc;
118 figure; hold on; grid minor;
119 ms = 3; % Markersize
120 id = find_points(Column);
121 curv_range = 1:150;
122 P1 = plot(Column.Curvature(curv_range),Column.Concrete_strain(curv_range,1),'b-',"DisplayName","Extreme Tensile Fiber","LineWidth",2);
123 plot(Column.Curvature(curv_range),Column.centroid_strain(curv_range),'g--',"DisplayName","Centroidal Fiber","LineWidth",2);
124 P2 = plot(Column.Curvature(curv_range),Column.Concrete_strain(curv_range,Column.Section.no_of_fibers),'r-',"DisplayName","Extreme ↙
Compressive Fiber","LineWidth",2);
125
126 add_dataTip(P1, id.et, ["\phi","\epsilon"],'northwest');
127 add_dataTip(P2, id.ACI, ["\phi","\epsilon"],'southwest');
128 add_dataTip(P2, id.Mn, ["\phi","\epsilon"],'southwest');
129 add_dataTip(P2, id.Spalling, ["\phi","\epsilon"],'southwest');
130
131 add_line('y', Column.Confined_Concrete.et, '\epsilon_{t}', 'r--', 'top')
132 add_line('y', -0.003, '\epsilon_{ACI}', 'r--', 'top')
133 add_line('y', -0.004, '\epsilon_{Mn}', 'r--', 'bot')
134 add_line('y', -0.006, '\epsilon_{Spalling}', 'r--', 'bot')
135 title("Concrete Strain-Curvature");
136 grid on; legend("Location","Best"); ylabel("Strain [in/in]"); xlabel("Curvature [1/in]"); xlim([0,6*10^-4]); set(gcf, 'Position',  [0, 800, 1300,500])
137 end
138 print_figure("06 Concrete Fibers Analysis");
139
140 %% (PRINTED) Fig 0.7  Steel Rebar Stress Plots
141 close all;
142 plots = plot_steel_stress(Column1, "Column Stress-Curvature Plot");
143
144 add_dataTip(plots(1), id1, ["\phi","\sigma_y"],'northwest');
145 add_dataTip(plots(2), id2, ["\phi","\sigma_y"],'northeast');
146 add_dataTip(plots(3), id3, ["\phi","\sigma_y"],'southeast');
147 add_dataTip(plots(4), id4, ["\phi","\sigma_y"],'southwest');
148 add_dataTip(plots(5), id5, ["\phi","\sigma_y"],'southwest');
149
150 grid minor;
151 print_figure("07 Steel Fibers Stress");
152
153 %% (PRINTED) Figure 1 Part i Moment Curvature
154 close all; clc;
155 plot_moment_curvature(Column1);
156 % print_figure("1 i Vanilla Moment Curvature");
157
158 %% (PRINTED) Figure 1 Part ii Moment Curvature w/ Points
159 close all; clc;
160 plot_moment_curvature(Column1);
161 add_all_dataTips(Column1);
162 print_figure("08 Moment Curvature with Points");
163
164 %% (PRINTED) Figure 1 Part iii Yield points
165 close all;
166 plot_moment_curvature(Column1);
167 P = plot_approx_curve(Column1,0, 1); P.Color = 'r';
168 xlim([0,4*10^-4]);
169 print_figure("09 True Yield Points");
170
```

```
171 %% (PRINTED) Figure 1 Part iv Effective Flexural Stiffness
172 close all; clc;
173
174 P = plot_moment_curvature(Column1,0); legend("Location",'northwest');
175 add_dataTip(P, Column1.id.et, ["\phi_{cr}","M_{Model}"],'northwest');
176 P.DataTipTemplate.DataTipRows(1) = [];
177 set(gca,'DefaultLineLineWidth',3)
178 x = [0,Column1.Pcr, Column1.Py]/Column1.norm_P;
179
180 % P = plot(x, Column1.Ec_Ie/1000*x,'--',"DisplayName","M_1(\phi)=\phi*E_c*I_{e}");
181 % add_dataTip(P, 2, ["\phi_{cr}","M_{1}"],'southwest');
182 % P.DataTipTemplate.DataTipRows(1) = [];
183
184 P = plot(x, Column1.Ec_Ig/1000*x,'--',"DisplayName","M_1(\phi)=\phi \cdot E_c*I_g = \phi \cdot"+ sprintf("%.3e",Column1.Ec_Ig/1000));
185 add_dataTip(P, 2 , ["\phi_{cr}","M_{1}"],'southeast');
186 P.DataTipTemplate.DataTipRows(1) = [];
187
188 P = plot(x, Column1.Ec_It/1000*x,'--',"DisplayName","M_2(\phi)=\phi \cdot E_c*I_t = \phi \cdot"+ sprintf("%.3e",Column1.Ec_It/1000));
189 add_dataTip(P, 2 , ["\phi_{cr}","M_{2}"],'southwest');
190 P.DataTipTemplate.DataTipRows(1) = [];
191
192 % P = plot(x, Column1.Ec_Ie/1000*2*x,'--',"DisplayName","M_3(\phi)=2*\phi*E_c*I_{e}");
193 % add_dataTip(P, 2 , ["\phi_{cr}","M_{3}"],'southeast');
194 % P.DataTipTemplate.DataTipRows(1) = [];
195
196 slope = Column1.Moment(Column1.id.et)/Column1.Curvature(Column1.id.et)/1000;
197 P = plot(x, slope*x,'--',"DisplayName","M_{model}(\phi)= \phi \cdot"+ sprintf("%.3e",slope));
198 xlim([0,1*10^-5]);
199 % print_figure("10 Measuring Effective Flexural Stiffness");
200
201 %% (PRINTED) Figure 1 Part v Bilinear and Four-Linear
202 close all; clc;
203 for Column = Column1
204     plot_moment_curvature([Column],1);
205     phi = Column.Curvature;
206     M = Column.Moment/1000;
207     E_y = Column.My/Column1.Py;
208     phi_y = Column.Mn/E_y;
209
210     bilinear_x = [0, phi_y, Column.Po, Column.Pul]; % Bilinear Points
211     bilinear_y = [0, Column.Mn, Column.Mo, Column.Mul]; % Bilinear Points
212     four_x = [0,  Column.Pcr,  phi_y,  Column.Po,  Column.Pul]; % Quad Points
213     four_y = [0, Column.Mcr, Column.Mn, Column.Mo, Column.Mul]; % Quad Points
214     set(gca,'DefaultLineLineWidth',3)
215
216     P = plot(four_x, four_y,'-p',"DisplayName","Four-Line");P.Color(4) = 0.6;
217     P = plot(bilinear_x, bilinear_y,'-o',"DisplayName","Bilinear"); P.Color(4) = 0.6;
218     xlim([0,15]); xticks(1:15);
219 end
220 % print_figure("11 Bilinear Quadlinear Curve");
221
222 %%
223 close all;clc;
224
225 E_y = Column.My/Column1.Py;
226 phi_y = Column.Mn/E_y;
227
```

```
228 P = plot_moment_curvature(Column1,1); legend("Location",'northwest');
229 add_dataTip(P, Column1.id.fy, ["\phi_{y}","M_y"],'northwest');
230 P.DataTipTemplate.DataTipRows(2) = [];
231 xlim([0,2]); xticks(1:15); legend("Location","Southeast");
232 add_line('x', 1, "\phi_y = 2.25\cdot \epsilon_y/D=1", 'r', 'bottom');
233 set(gca,'DefaultLineLineWidth',3);
234 four_x = [0,  Column.Pcr,  phi_y,  Column.Po,  Column.Pul]; % Quad Points
235 four_y = [0, Column.Mcr, Column.Mn, Column.Mo, Column.Mul]; % Quad Points
236 bilinear_x = [0, phi_y, Column.Po, Column.Pul]; % Bilinear Points
237 bilinear_y = [0, Column.Mn, Column.Mo, Column.Mul]; % Bilinear Points
238
239 P = plot(bilinear_x, bilinear_y,'-o',"DisplayName","Bilinear"); P.Color(4) = 0.6;
240 P = plot(four_x, four_y,'-p',"DisplayName","Four-Line");P.Color(4) = 0.6;
241 add_dataTip(P,3, ["\phi_y","M_y"],'southeast');
242 P.DataTipTemplate.DataTipRows(2) = [];
243 print_figure("11 Comparing Yield Points Bilinear Quadlinear Curve");
244 %%
245 close all; hold on; grid minor;
246 P = plot_moment_curvature(Column2,0); P.Color = '#D95319';
247 plot_moment_curvature(Column1,0)
248 print_figure("12 Comparing Two Not Normalized Curves");
249
250 %% (PRINTED) Figure 2 Normalized Moment Curvature
251 close all; clc;
252 plots = plot_moment_curvature([Column1,Column2],1);
253
254 % Ultimate curvature ductility
255 fprintf("The section curvature ductility of Column 1 is %.2f \n", Column1.Curvature_ductility);
256 fprintf("The section curvature ductility of Column 2 is %.2f \n", Column2.Curvature_ductility);
257 % print_figure("12 Normalized Moment Curvature");
258
259 %% (PRINTED) Figure 3 M-phi envelope EQ1 EQ2 EQ2
260 close all;
261
262 figure; hold on; grid on; legend('location','best'); grid minor;
263 [nM1, nM2, nM3, nP1, nP2 ,nP3] = readvars('..\files\Eq3.txt');
264
265 plot(nP3,nM3,"DisplayName","EQ3");
266 plot(nP2,nM2,"DisplayName","EQ2");
267 plot(nP1,nM1,"DisplayName","EQ1");
268 plot([-flip(Column1.Normalized_P),Column1.Normalized_P],[-flip(Column1.Normalized_M), Column1.Normalized_M],'r',"DisplayName"," ↙
Monotonic M-\phi Envelope","LineWidth",3);
269
270 title("Normalized Moment Curvature");
271 ylabel("Normalized Moment $\frac{M}{D^3\cdot f'_c}$","Interpreter","latex");
272 xlabel(" Normalized Curvature $\frac{\phi \cdot D}{\lambda\epsilon_y}$","Interpreter","latex");
273 set(gcf, 'Position', [0, 800, 1300,500]); xlim([-13 13]);xticks(-14:2:14);
274 print_figure("13 Monotonic Envelope");
275
276 %% (PRINTED) Figure 4 Double Longitudinal Rebar
277 clc; close all;
278 P = plot_moment_curvature([Column1,Column3],1);
279 P(1).DisplayName = '1* Longitudinal Rebar';
280 add_dataTip(P(1), Column1.id.fy, ["\phi_y","M_y"],'southeast');
281 add_all_dataTips(Column3,1)
282 P = add_line('x', 1, '\phi_y = 2.25*\epsilon_y/D', 'k--', 'top'); P.LabelHorizontalAlignment ='center';
283 add_markers([Column1,Column3],1); % Adds the markers
```

```
284 ylim([0, inf]);  xticks(1:15);
285 print_figure("14 Double Longitudinal Rebar");
286
287 %% (PRINTED) Figure 4 Report
288 clc;
289 fprintf("\nFor Column 1:\nMn = %.3g kip-in \nMo = %.3g kip-in \nThe ratio is %.2f\n",  Column1.Mn,Column1.Mo,Column1.Mn/Column1.
Mo);
290 fprintf("\nFor Column 1:\nMn = %.3g kip-in \nMo = %.3g kip-in \nThe ratio is %.2f\n",  Column3.Mn,Column3.Mo,Column3.Mn/Column3.
Mo);
291
292 fprintf('\nBetween the two columns, Mn2/Mn1 = %.2f', Mn2/Mn1);
293 fprintf('\nBetween the two columns, Mo2/Mo1 = %.2f', Mo2/Mo1);
294
295 % Ultimate Curvature Ratios
296 fprintf("\n\nThe section curvature ductility of Column 1 is %.2f \n", Column1.Curvature_ductility);
297 fprintf("The section curvature ductility of Column 2 is %.2f \n", Column3.Curvature_ductility);
298 % Ratio of flexural stiffness
299 fprintf("\nThe flexural stiffness ratio of Column 1 is %.2f \n", Column1.Ec_Ie / Column1.Ec_Ig);
300 fprintf("The flexural stiffness ratio of Column 2 is %.2f \n", Column3.Ec_Ie / Column3.Ec_Ig);
301
302 %% (PRINTED) Figure 5 Increased Axial Load Moment Curvature
303 close all; clc;
304 list_of_columns = [Column1, Column4, Column5, Column6];
305 P = plot_moment_curvature(list_of_columns, 1); % Plots the moment curvature
306 plot_approx_curve(list_of_columns);
307 add_markers(list_of_columns,1); % Adds the markers
308 add_line('x', 1, '\phi_y = 2.25*\epsilon_y/D', 'r--', 'top');
309 for p = P; p.Color(4) = 0.4; end
310 P(1).DisplayName = '1*Axial Force = 522 kip';
311 xlim([0,2]); legend("Location","southeast")
312 print_figure("15 Increased Axial Load Yield");
313
314 %% (PRINTED) Figure 5 (Full)
315 close all; clc;
316 list_of_columns = [Column1, Column4, Column5, Column6];
317 P = plot_moment_curvature(list_of_columns, 1); % Plots the moment curvature
318 add_markers(list_of_columns,1); % Adds the markers
319 % add_line('x', 1, '\phi_y = 2.25*\epsilon_y/D', 'r--', 'top');
320 P(1).DisplayName = '1*Axial Force = 522 kip';
321 print_figure("16 Increased Axial Load Full Picture");
322
323 %% (PRINTED) Figure 6 Increased Axial Load Stiffness
324 close all; clc;
325 plot_figure6([Column1, Column4, Column5, Column6]);
326 add_line('y', 0.5, 'ACI Reccomendation', 'r', 'top');
327 ylim([0.3 inf]);
328 print_figure("17 Increased Axial Load Stiffness Modifier");
329
330 %% (PRINTED) Figure 7
331 close all; clc;
332 plot_figure7([Column1, Column4, Column5, Column6]);
333 print_figure("18 Increased Axial Load Ultimate Ductility");
334
335 %% (PRINTED) Figure 8
336 close all; clc;
337 plot_figure8([Column1, Column7, Column8]);
338 print_figure("19 Increased Axial Load Hoop Ratio");
```

```
339
340 %% (PRINTED) Figure 9
341 close all; clc;
342 for Column = Column1
343     lp1 = Column.Confined_Concrete.lp1; % [in]; height of plastic hinge 1
344     height = Column.height; % [in]; overall height
345     no_levels = 8; %
346
347     height_levels = (height-lp1)*(1:no_levels)/no_levels + lp1;
348     height_levels = [0, lp1, height_levels]';
349     ratio = height_levels/height;
350
351     subplot(1,2,1); grid minor; hold on; title('First-Order Bending Moment Diagram'); ylabel('Height [in]','Interpreter','latex'); xlabel('Moment ↙
[kip-in]','Interpreter','latex'); grid(); legend();
352     Mn_list = Column.Mn*(1-ratio);
353     Mo_list = Column.Mo*(1-ratio);
354     plot(Mo_list/Column.norm_M/1000,height_levels,'-o','Color','#D95319','LineWidth',2,'DisplayName','M^o');
355     plot(Mn_list/Column.norm_M/1000,height_levels,'-o','Color','#4DBEEE','LineWidth',2,'DisplayName','M_n');
356
357     subplot(1,2,2);grid minor; hold on; title('Curvature Diagram'); ylabel('Height [in]','Interpreter','latex'); xlabel('Normalized Curvature $\frac ↙
{\phi \cdot D}{\lambda \epsilon_y}$','Interpreter','latex'); grid(); legend();
358     phi_n = find_curvature_of(Column, Mn_list, 1);
359     phi_o = find_curvature_of(Column, Mo_list, 1);
360     plot(phi_o,height_levels,'-o','LineWidth',2,'Color','#D95319','DisplayName','M^o');
361     plot(phi_n,height_levels,'-o','LineWidth',2,'Color','#4DBEEE','DisplayName','M_n');
362     set(gcf, 'Position',  [0, 800, 1300,500]);
363 end
364 print_figure("20 Moment-Curvature Along Element");
365
366 %% (PRINTED) Figure 10
367 % close all; clc;
368 test_moment_curvature(Column1)
369 % print_figure("21a Moment Curvature Approximations");
370 [Force_real, Delta_real] = plot_force_displacement_real(Column1);
371 % print_figure("21b Force Displacements");
372 [Force_approx, Delta_approx] = plot_force_displacement_approx(Column1);
373 % print_figure("21c Force Displacements Approximate");
374 %% (PRINTED) Figure 11
375 close all;
376 figure; hold on; grid on; legend('location','northwest'); grid minor;
377 xline(0,"HandleVisibility",'off');
378 yline(0,"HandleVisibility",'off');
379 [V_W1, V_W2, V_W3, d_L1, d_L2, d_L3 ] = readvars('..\files\Dynamic_EQ.txt');
380 P = plot(d_L3,V_W3,"DisplayName","EQ1","LineWidth",2); P.Color(4) = 0.5;
381 P = plot(d_L2,V_W2,"DisplayName","EQ2","LineWidth",2); P.Color(4) = 0.5;
382 P = plot(d_L1,V_W1,"DisplayName","EQ3","LineWidth",2); P.Color(4) = 0.5;
383
384 V_W = Force_real / Column1.applied_force;
385 d_L = Delta_real / Column1.height;
386 P = plot([-flip(d_L),d_L],[-flip(V_W),V_W],"DisplayName","Model F-\Delta Envelope","LineWidth",3);
387
388 V_W = Force_approx / Column1.applied_force;
389 d_L = Delta_approx / Column1.height;
390 P = plot([-flip(d_L),d_L],[-flip(V_W),V_W],"DisplayName","Approximated F-\Delta Envelope","LineWidth",3);
391
392 xlabel("Drift Ratio \delta/L"); ylabel("Base Shear Ratio V/W"); title("Drift Ratio Versus Base Shear");
393 xlim([-.05,0.05]);
```

```
394 set(gcf, 'Position',  [0, 800, 1300,500])
395 % print_figure("22 Dynamic Envelope");
396
397 % close all;
398 %%
399 clc;
400 print_id(Column1); disp(newline)
401 print_id(Column4); disp(newline)
402 print_id(Column5); disp(newline)
403 print_id(Column6); disp(newline)
404 find_curvature_of(Column1, 53400001,0)
405 %% Function
406 % Add individual Data tips
407 function add_dataTip(plot, n, name,location)
408     plot.DataTipTemplate.DataTipRows(1).Label = name(1);
409     plot.DataTipTemplate.DataTipRows(1).Format = '%.3g';
410 %     plot.DataTipTemplate.DataTipRows(2) = []
411     plot.DataTipTemplate.DataTipRows(2).Label = name(2);
412     plot.DataTipTemplate.DataTipRows(2).Format = '%.3g';
413     datatip(plot, 'DataIndex', n,'Location',location);
414 end
415
416 function add_all_dataTips(list_of_columns, normalized)
417     arguments
418         list_of_columns
419         normalized = false
420     end
421
422     for Column = list_of_columns
423         if normalized == false
424             phi = Column.Curvature;
425             M = Column.Moment/1000;
426         else
427             M = Column.Moment/(Column.Section.diameter^3*Column.Confined_Concrete.fc);
428             phi = Column.Curvature*Column.Section.diameter/Column.Section.lamba/Column.Long.ey;
429         end
430         id = find_points(Column);
431         P = plot(phi,M); add_dataTip(P, id.et, ["\phi_{cr}","M_{cr}"],'southeast'); P.HandleVisibility= 'off'; P.Color(4) = 0;
432         P = plot(phi,M); add_dataTip(P, id.fy, ["\phi_y","M_y"],'southeast'); P.HandleVisibility= 'off'; P.Color(4) = 0;
433         P = plot(phi,M); add_dataTip(P, id.ACI, ["\phi_{n,ACI}","M_{n,ACI}"],'southeast'); P.HandleVisibility= 'off'; P.Color(4) = 0;
434         P = plot(phi,M); add_dataTip(P, id.Mn, ["\phi_n","M_n"],'northeast'); P.HandleVisibility= 'off'; P.Color(4) = 0;
435         P = plot(phi,M); add_dataTip(P, id.Spalling, ["\phi_{spalling}","M_{spalling}"],'southeast'); P.HandleVisibility= 'off'; P.Color(4) = 0;
436         P = plot(phi,M); add_dataTip(P, id.Max, ["\phi^o","M^o"],'southwest'); P.HandleVisibility= 'off'; P.Color(4) = 0;
437         P = plot(phi,M); add_dataTip(P, id.End, ["\phi_{ul}","M_{ul}"],'southeast'); P.HandleVisibility= 'off'; P.Color(4) = 0;
438     end
439 end
440
441 function id = find_points(Column)
442     [~, id.et] = min(abs(Column.Confined_Concrete.et- Column.Concrete_strain(:,2)));
443
444     [~, id.fyc] = min(abs(Column.Confined_Concrete.ec- Column.Concrete_strain(:,Column.Section.no_of_fibers)));
445     [~, id.fys] = min(abs(Column.Long.ey- Column.Steel_strain(:,1)));
446     id.fy = min(id.fyc,id.fys);
447
448     [~, id.ACI] = min(abs(-0.003- Column.Concrete_strain(:,Column.Section.no_of_fibers)));
449     [~, id.Mn] = min(abs(-0.004- Column.Concrete_strain(:,Column.Section.no_of_fibers)));
450     [~, id.Spalling] = min(abs(-0.006- Column.Concrete_strain(:,Column.Section.no_of_fibers)));
```

```
451     [~, id.Max] = max(Column.Moment);
452     id.End = length(Column.Moment);
453
454     if (Column.Steel_strain(end,1) - 0.06) <= 0.0001
455         id.end_marker = "d";
456     elseif (Column.Concrete_strain(end,25) + 0.02) <=0.0001
457         id.end_marker = "s";
458     end
459
460 end
461
462 function print_id(Column)
463 %     clc;
464     [~, id.et] = min(abs(Column.Confined_Concrete.et- Column.Concrete_strain(:,2)));
465
466     [~, id.fyc] = min(abs(Column.Confined_Concrete.ec- Column.Concrete_strain(:,Column.Section.no_of_fibers)));
467     [~, id.fys] = min(abs(Column.Long.ey- Column.Steel_strain(:,1)));
468     id.fy = min(id.fyc,id.fys);
469
470     [~, id.ACI] = min(abs(-0.003- Column.Concrete_strain(:,Column.Section.no_of_fibers)));
471     [~, id.Mn] = min(abs(-0.004- Column.Concrete_strain(:,Column.Section.no_of_fibers)));
472     [~, id.Spalling] = min(abs(-0.006- Column.Concrete_strain(:,Column.Section.no_of_fibers)));
473     [~, id.Max] = max(Column.Moment);
474     id.End = length(Column.Moment);
475
476     pts = ["Cr", "Y", "ACI", "Mn", "Sp", "Mo", "Mu"];
477     ids = [id.et, id.fy, id.ACI, id.Mn, id.Spalling, id.Max, id.End];
478     [ids, sort_order] = sort(ids);
479     pts = pts(sort_order);
480
481 %     fprintf("%3s \t %3.0f \t %7.2e \t %.2e \n", [pts; ids; Column.Curvature(ids); Column.Moment(ids)]);
482 %     fprintf('\n');
483     fprintf("%3s \t %3.0f \t %7.2f \t %7.4f \n", [pts; ids; Column.Normalized_P(ids); Column.Normalized_M(ids)]);
484 %     fprintf("(%.2f, %.4f)\n", [Column.Normalized_P(ids); Column.Normalized_M(ids)]);
485 end
486
487 function plots = plot_steel_stress(Column, title_name)
488
489     figure; hold on; title(title_name)
490     set(gca,'DefaultLineLineWidth',2)
491     Column.Steel_stress = Column.Steel_stress/1000;
492     plots= [];
493     P = plot(Column.Curvature,Column.Steel_stress(:,1),"DisplayName","Fiber 1"); plots = [ plots,P];
494     P = plot(Column.Curvature,Column.Steel_stress(:,2),"DisplayName","Fiber 2"); plots = [ plots,P];
495     P = plot(Column.Curvature,Column.Steel_stress(:,3),"DisplayName","Fiber 3"); plots = [ plots,P];
496     P = plot(Column.Curvature,Column.Steel_stress(:,4),"DisplayName","Fiber 4"); plots = [ plots,P];
497     P = plot(Column.Curvature,Column.Steel_stress(:,5),"DisplayName","Fiber 5"); plots = [ plots,P];
498
499     set(gcf, 'Position',  [0, 800, 1300,500])
500     grid on; legend("Location","east");
501     ylabel("Stress [ksi]"); xlabel("Curvature [1/in]");
502 end
503
504 function P = add_line(x_or_y, value, label, color, position)
505     arguments
506         x_or_y
507         value
```

```
508         label
509         color = 'k'
510         position = 'bottom'
511     end
512     if x_or_y == 'x'
513         P = xline(value, color, label,'HandleVisibility','off','LabelOrientation','horizontal','LabelVerticalAlignment',position);
514     else
515         P = yline(value, color, label,'HandleVisibility','off','LabelOrientation','horizontal','LabelVerticalAlignment',position);
516     end
517 end
518
519 function add_markers(list_of_columns, normalized)
520     arguments
521         list_of_columns
522         normalized = false
523     end
524     markers = ['o','p','h','d','s']; i =1;
525     col = [ [0, 114, 189]; [217, 83, 25];[237, 177, 32]; [119, 172, 48]; [162, 20, 47]]/255;
526     for Column = list_of_columns
527         pt_x = [Column.Pcr, Column.Py, Column.PACI, Column.Pn, Column.Ps, Column.Po];
528         pt_y = [Column.Mcr, Column.My, Column.MACI, Column.Mn, Column.Ms, Column.Mo];
529         if normalized == false
530             scatter(pt_x/Column.norm_P, pt_y/Column.norm_M,100,col(i), markers(i,:),'filled','HandleVisibility', 'off');
531             scatter(Column.Curvature(end)/Column.norm_P,Column.Moment(end)/Column.norm_M,140,col(i,:),Column.end_marker,'filled', ↙
'HandleVisibility', 'off');
532         else
533             scatter(pt_x, pt_y, 100,col(i,:), markers(i) ,'filled', 'HandleVisibility', 'off');
534             scatter(Column.Pul,Column.Mul,140,col(i,:),Column.end_marker,'filled', 'HandleVisibility', 'off');
535         end
536         i = i + 1;
537     end
538 end
539
540 function plots = plot_figure6(list_of_columns)
541     figure; grid on; grid minor; hold on; plots= [];
542     for Column = list_of_columns
543         P = scatter(Column.Axial_load_ratio, Column.Ec_Ie/Column.Ec_Ig,140,'p','filled','DisplayName' ,Column.name);
544         P.DataTipTemplate.DataTipRows(1) = dataTipTextRow('P/(A_g*f_c)',Column.Axial_load_ratio,'%.3g');
545         P.DataTipTemplate.DataTipRows(2) = dataTipTextRow('(E_cI_e/E_cI_g)',Column.Ec_Ie/Column.Ec_Ig,'%.2f');
546         datatip(P,'DataIndex',1);
547         plots = [plots, P];
548     end
549     plots(1).DisplayName = '1*Axial Force = 522 kip';
550     legend('Location','southeast');
551     xlabel('Axial Load Ratio $\frac{P}{A_g \cdot f_c}$','Interpreter','latex');
552     ylabel('Stiffness Modified $\frac{E_c \cdot I_e}{E_c \cdot I_g}$','Interpreter','latex');
553     title('Flexure Rigidity Per Axial Load Ratio')
554     set(gcf, 'Position',  [0, 800, 1300,500])
555 end
556
557 function plots = plot_figure7(list_of_columns)
558     figure; grid on; grid minor; hold on; plots= [];
559     for Column = list_of_columns
560         P = scatter(Column.Axial_load_ratio,Column.Curvature_ductility,'DisplayName' ,Column.name);
561         plots = [plots, P];
562         P.DataTipTemplate.DataTipRows(1) = dataTipTextRow('P/(A_g*f_c)',Column.Axial_load_ratio,'%.3g');
563         P.DataTipTemplate.DataTipRows(2) = dataTipTextRow('\mu_{\phiu}',Column.Curvature_ductility,'%.2f');
```

```
564         datatip(P,'DataIndex',1);
565     end
566     plots(1).DisplayName = '1*Axial Force = 522 kip';
567     title('Ultimate Curvature Ductility Per Axial Load Ratio')
568     legend('Location','northeast');
569 
570     xlabel('Axial Load Ratio $\frac{P}{A_g \cdot f_c}$','Interpreter','latex');
571     ylabel('Ultimate Ductility $\frac{\phi_{ul} \cdot D}{\lambda\epsilon_y}$','Interpreter','latex');
572     set(gcf, 'Position',  [0, 800, 1300,500])
573 end
574 
575 function plots = plot_figure8(list_of_columns)
576     figure(8); grid on; grid minor; hold on; plots= [];
577     for Column = list_of_columns
578         P = scatter(Column.Axial_load_ratio,Column.Hoop.rho,140,'p','filled','DisplayName' ,Column.name);
579         P.DataTipTemplate.DataTipRows(1) = dataTipTextRow('\rho_{hoop}',Column.Hoop.rho,'%.3g');
580         P.DataTipTemplate.DataTipRows(2) = dataTipTextRow('P/(A_g*f_c)',Column.Axial_load_ratio,'%.3g');
581         P.DataTipTemplate.DataTipRows(3) = dataTipTextRow('\mu_{\phiu}',Column.Curvature_ductility,'%.2f');
582         datatip(P,'DataIndex',1);
583         plots = [plots, P];
584     end
585     plots(1).DisplayName = '1*Axial Force = 522 kip';
586     title('Hoop Ratio Per Axial Load Ratio')
587     legend('Location','southeast');
588     ylabel('\rho_{hoop}');
589     xlabel('Axial Load Ratio $\frac{P}{A_g \cdot f_c}$','Interpreter','latex');
590     set(gcf, 'Position',  [0, 800, 1300,500]);
591 end
592 
593 function plots = plot_approx_curve(list_of_columns , normalized, option)
594     arguments
595         list_of_columns
596         normalized = true
597         option = false
598     end
599     plots = [];
600     col = [ "#0072BD",  "#D95319",  "#EDB120", "#77AC30",   "#A2142F"];
601     i = 1;
602     for Column = list_of_columns
603         if normalized == false
604             p_factor = 1/ Column.norm_P;
605             m_factor = 1/Column.norm_M/1000;
606         else
607             p_factor = 1;
608             m_factor = 1;
609         end
610         x = [0, Column.Py Column.Mn/Column.Ec_Ie_n, Column.Pn]*p_factor;
611         y = [0, Column.My, Column.Mn, Column.Mn]*m_factor;
612         P = plot(x,y,'Color',col(i),"HandleVisibility",'off','LineWidth',2);
613         datatip(P,'DataIndex',3,'Location','northwest');
614 
615         i = i + 1;
616 
617         if option == true
618             P.DataTipTemplate.DataTipRows(1).Label = '\phi';
619             P.DataTipTemplate.DataTipRows(1).Format = '%.2e';
620             P.DataTipTemplate.DataTipRows(2).Label = 'M';
```

```
621             P.DataTipTemplate.DataTipRows(2).Format = '%.0f';
622             datatip(P,'DataIndex',2,'Location','northwest');
623             datatip(P,'DataIndex',3,'Location','northwest');
624             datatip(P,'DataIndex',4,'Location','northwest');
625         else
626             P.DataTipTemplate.DataTipRows(1).Label = '\phi_y';
627             P.DataTipTemplate.DataTipRows(1).Format = '%.2f';
628             P.DataTipTemplate.DataTipRows(2) = [];
629         end
630         plots = [plots,P];
631     end
632 end
633
634 function curvature = four_line(Column, list_of_moments, normalized)
635     arguments
636         Column
637         list_of_moments
638         normalized = false
639     end
640     if normalized == false
641         p_factor = 1/ Column.norm_P;
642         m_factor = 1/Column.norm_M;
643     else
644         p_factor = 1;
645         m_factor = 1;
646     end
647     list_of_moments = reshape(list_of_moments,[numel(list_of_moments),1]);
648
649     Pcr = Column.Pcr*p_factor;
650     Pn = Column.Mn/(Column.My/Column.Py)*p_factor;
651     Po = Column.Po*p_factor;
652     Pul = Column.Pul*p_factor;
653
654     Mcr = Column.Mcr*m_factor;
655     Mn =  Column.Mn*m_factor;
656     Mo = Column.Mo*m_factor;
657     Mul = Column.Mul*m_factor;
658
659     curvature = zeros(numel(list_of_moments),1);
660     i = 1;
661     for moment = list_of_moments'
662         if 0<= moment && moment<= Mcr
663             m = (Mcr- 0)/(Pcr - 0);
664             curvature(i) = (moment - 0)/m + 0;
665         elseif Mcr<= moment && moment<= Mn
666             m = (Mn- Mcr) /(Pn - Pcr);
667             curvature(i) = (moment - Mcr)/m + Pcr;
668         elseif Mn<= moment && moment<= Mo
669             m = (Mo- Mn) /(Po - Pn);
670             curvature(i) = (moment - Mn)/m + Pn;
671         elseif Mo<= moment && moment<= Mul
672             m = (Mul- Mo) /(Pul - Po);
673             curvature(i) = (moment - Mo)/m + Po;
674         else
675             curvature(i) = Pul;
676         end
677         i = i+1;
```

```
678     end
679
680 end
681
682 function phi = find_curvature_of(Column, list_of_moments, normalized)
683     arguments
684         Column
685         list_of_moments
686         normalized = false
687     end
688     list_of_moments = reshape(list_of_moments,[1,numel(list_of_moments)]);
689
690     if normalized == true
691         V = list_of_moments';
692         N = Column.Normalized_M';
693         A = repmat(N,[1 length(V)]);
694         [~,ids] = min(abs(A-V'));
695         phi = Column.Normalized_P(ids);
696     else
697         V = list_of_moments';
698         N = Column.Moment';
699         A = repmat(N,[1 length(V)]);
700         [~,ids] = min(abs(A-V'));
701         phi = Column.Curvature(ids);
702     end
703 end
704
705 function test_moment_curvature(Column)
706         moment = 1:600000:Column.Mul/Column.norm_M;
707         curvature = zeros(1,numel(moment));
708         curvature_approx = zeros(1,numel(moment));
709         for i = 1:numel(moment)
710             curvature(i) = find_curvature_of(Column, moment(i));
711             curvature_approx(i) = four_line(Column, moment(i), 0);
712         end
713         figure(); hold on;
714         plot(curvature_approx,moment/1000,'-','LineWidth',2,"DisplayName","Approximate 4 Line Curve"); grid on;
715         plot(curvature,moment/1000,'-','LineWidth',2,"DisplayName","Real M-\phi Curve");
716         title("Moment Curvature"); legend("Location","Best");
717         ylabel("Moment [kip-in]"); xlabel("Curvature [1/in]");
718         set(gcf, 'Position',  [0, 800, 1300,500])
719 end
720
721 function [V_minus_P_delta, displacements] = plot_force_displacement_real(Column)
722     clc;
723     % Find moments along the curve
724     lp1 = Column.Confined_Concrete.lp1; % [in]; height of plastic hinge 1
725     lp2 = Column.Confined_Concrete.lp2; % [in]; height of plastic hinge 2
726     height = 24*12; % [in]; overall height
727     no_levels = 4; % Numer
728
729     height_levels = (height-lp1)*(1:no_levels)/no_levels + lp1;
730     height_levels = [0, lp1, height_levels]';
731     ratio = (1-height_levels/height);
732     height_levels(2) = lp1 + lp2;
733     arm = [ height ,(height-lp1)/no_levels/2.*(2*no_levels-1:-2:1)];
734     widths = [lp1+lp2 , (height-lp1)/no_levels*ones(1,no_levels)];
```

```
735     i = 1;
736
737 %     moments_points = [0, Column.Mcr, Column.My, Column.MACI, Column.Mn, Column.Ms, Column.Mo, Column.Mul] /Column.norm_M;
738     moments_points = 1:600000:Column.Mul/Column.norm_M;
739     displacements = zeros(1,numel(moments_points));
740
741     for moment_base = moments_points
742         level_moment = moment_base * ratio;
743         curvature = find_curvature_of(Column, level_moment', 0);
744         mean_curvature = mean([curvature(1:end-1);curvature(2:end)]);
745         mean_curvature(1) = curvature(1);
746         displacements(i) = sum(mean_curvature.* widths .*arm);
747         i = i + 1;
748     end
749
750     P = -Column.total_axial_load;
751     P_Delta = P*displacements/height;
752
753     V = moments_points/ height;
754     V_minus_P_delta = V - P_Delta;
755
756     figure(); hold on;
757     plot(displacements,V/1000,'-o',"DisplayName","F:\Delta");
758     plot(displacements,V_minus_P_delta/1000,'-o',"DisplayName","F:\Delta - P\Delta Effect");
759     plot(displacements,P_Delta/1000,'-o',"DisplayName","P\Delta Effect");
760     xlabel('Lateral Displacement \Delta [in]'); ylabel('Force [kips]'); title('Force Displacement Curve');grid on;
761     legend("Location","best");
762     set(gcf, 'Position',  [0, 800, 1300,500])
763 end
764
765 function [V_minus_P_delta, displacements_approx] = plot_force_displacement_approx(Column)
766         % Find moments along the curve
767         lp1 = Column.Confined_Concrete.lp1; % [in]; height of plastic hinge 1
768         lp2 = Column.Confined_Concrete.lp2; % [in]; height of plastic hinge 2
769         height = Column.height; % [in]; overall height
770         no_levels = 4; % Numer
771
772         height_levels = (height-lp1)*(1:no_levels)/no_levels + lp1;
773         height_levels = [0, lp1 + lp2, height_levels]';
774         ratio = (1-height_levels/height);
775         arm = [ height ,(height-lp1)/no_levels/2.*(2*no_levels-1:-2:1)];
776         widths = [lp1+lp2 , (height-lp1)/no_levels*ones(1,no_levels)];
777         i = 1;
778
779         moments_points = 1:600000:Column.Mul/Column.norm_M;
780         displacements_approx = zeros(1,numel(moments_points));
781
782         for moment_base = moments_points
783             level_moment = moment_base * ratio;
784             curvature_approx = four_line(Column, level_moment,0);
785             mean_curvature_approx = mean([curvature_approx(1:end-1)';curvature_approx(2:end)']);
786
787             mean_curvature_approx(1) = curvature_approx(1);
788             displacements_approx(i) = sum(mean_curvature_approx.* widths .*arm);
789             i = i + 1;
790         end
791
```

```
792         P = -Column.total_axial_load;
793         V = moments_points/ height;
794         P_Delta = P*displacements_approx/height;
795         V_minus_P_delta = V - P_Delta;
796
797         figure(); hold on;
798         plot(displacements_approx,V/1000,'-o',"DisplayName","F:\Delta");
799         plot(displacements_approx,V_minus_P_delta/1000,'-o',"DisplayName","F:\Delta - P\Delta Effect");
800         plot(displacements_approx,P_Delta/1000,'-o',"DisplayName","P\Delta Effect");
801         xlabel('Lateral Displacement \Delta  [in]'); ylabel('Force [kips]'); title('Force Displacement Curve Using Approximate 4-Pt M\phi Curve'); ↙
grid on;
802         legend("Location","best");
803         set(gcf, 'Position',  [0, 800, 1300,500])
804 end
805
806 function plots = plot_moment_curvature(list_of_columns, normalized)
807     arguments
808         list_of_columns
809         normalized = false
810     end
811     plots = [];
812 %     figure;
813     hold on; grid on; grid minor; legend("Location","Best");
814     col = [ "#0072BD",  "#D95319",  "#EDB120", "#77AC30",   "#A2142F"];
815     i = 1;
816     for Column = list_of_columns
817         id = find_points(Column);
818         if normalized == false
819             title("Moment Curvature")
820             ylabel("Moment [kip-in]"); xlabel("Curvature [1/in]");
821             P = plot(Column.Curvature, Column.Moment/1000,'Color',col(i),'DisplayName',Column.name,"LineWidth",3);P.Color(4) = 0.6;
822             plots = [plots,P];
823         else
824             title("Normalized Moment Curvature")
825             ylabel("Normalized Moment $\frac{M}{D^3\cdot f'_c}$","Interpreter","latex");
826             xlabel(" Normalized Curvature $\frac{\phi \cdot D}{\lambda\epsilon_y}$","Interpreter","latex");
827             M_factor = 1/(Column.Section.diameter^3*Column.Confined_Concrete.fc);
828             phi_factor = Column.Section.diameter/Column.Section.lamba/Column.Long.ey;
829             P = plot(Column.Curvature* phi_factor, Column.Moment*M_factor,'Color',col(i),'DisplayName',Column.name,"LineWidth",3);P.Color ↙
(4) = 0.6;
830             plots = [plots,P];
831         end
832         i = i+1;
833     end
834 set(gcf, 'Position',  [0, 800, 1300,500])
835 xlim([0, inf]);
836 end
837
838 function print_figure(file_name)
839     % Saves the figures in a consistent manner
840     orient(gcf,'landscape');
841     folder = '..\figures\';
842     name = 'Figure ' +string(file_name);
843     print(folder+name,'-dpdf','-fillpage','-PMicrosoft Print to PDF','-r600','-painters')
844     print(folder+name,'-dsvg','-PMicrosoft Print to PDF','-r600','-painters')
845 end
```

```
function [fcu]= Mander_unconfined_concrete_split(strain, Concrete)
  % Return stress in confined concrete
  %% Input
  % Concrete data structure
     % fc is the modulus of the concrete
     % ecu is the maximum compressive strain
     % ec is the strain at the maximum compressive force
     % lambda_c is the softening branch normalization factor
  % strain is the current strain in the concrete fiber
   %% Output
   % fcu is the stress in the unconfined concrete fiber
   %%
   fc = Concrete.fc;
   n_E = 1 + 3600/fc;
   i = 1;

   fcu = zeros(1,numel(strain));
   for es = strain
      e_ec = es/Concrete.ec;
      if 0 <= es && es <= Concrete.et % Elastic Tension
         fcu(i) = Concrete.Ec *es;
      elseif es >= Concrete.et % Post Tensile cracking concrete
         fcu(i) = 0.7*Concrete.ft/(1+sqrt(500*es));
      elseif es <= 0 && es >= Concrete.ec % Elastic and hardening range
         fcu(i) = -(1-(1-e_ec)^n_E) *fc;
      elseif es < Concrete.ec && es >= Concrete.ecu % Post peak compression
         r_c = 1/(1-1/n_E);
         L_c = Concrete.lambda_c;
         fcu(i) = -(1+(1/L_c)*(e_ec-1))/(r_c-1+(1+(1/L_c)*(e_ec-1))^r_c)*r_c*fc;
      else
         fcu(i) = 0;
      end
      i = i +1;
   end
end
```

```
function [fc_con]= Mander_confined_concrete_split(strain, Concrete)
  % Return stress in confined concrete
  %% Input
  % Concrete data structure
     % fcc is the confine concrete compressive strength
     % ecc is the compressive strain at fcc
     % r_cc is the power term for the curve
     % lambda_c is the softening branch normalization factor
  % strain is the current strain in the concrete fiber
   %% Output
   % fc is the stress in the confined concrete fiber
   %%
   fcc = Concrete.fcc;
   ecc = Concrete.ecc;
   r_cc = Concrete.r_cc;
   i = 1;
   fc_con = zeros(1,numel(strain));
   for es = strain
      if 0 <= es && es <= Concrete.et % Elastic Tension
         fc_con(i) = Concrete.Ec *es;
      elseif es >= Concrete.et % Post Tensile cracking concrete
         fc_con(i) = 0.7*Concrete.ft/(1+sqrt(500*es));
      elseif es <= 0 && es >= ecc % Elastic and hardening range
         L_c = 1;
         fc_con(i) = -(1+1/L_c*(es/ecc-1))/(r_cc-1+(1+1/L_c*(es/ecc-1))^r_cc)*r_cc*fcc;
      elseif es <= ecc && es >= Concrete.ecu  % Post peak
         L_c = Concrete.lambda_c;
         fc_con(i) = -(1+1/L_c*(es/ecc-1))/(r_cc-1+(1+1/L_c*(es/ecc-1))^r_cc)*r_cc*fcc;
      else
         fc_con(i) = 0;
      end
      i = i + 1;
   end
end
```

```
function[fs]=Mander_steel_split(strain, Steel)
% Obtains steel stress using Mander's reinforcing steel model.
%% Input
% Es is the Young's modulus of steel
% fy is the yield stress of steel
% P is the power of the function for the strain hardening region
% esh is the strain at onset of strain hardening
% esu is the uniform strain
% strain is the strain at the fiber
% fsu is the ultimate yield strength
%% Output
% fs is the stress at strain
%%

Es = Steel.Es;
fy = Steel.fy;
P = Steel.P;
esh = Steel.esh;
esu = Steel.esu;
fsu = Steel.fsu;
i = 1;

fs = zeros(1,numel(strain));
for es = strain
if es>0
    if es <=fy/Es
        fs(i)=es*Es;
    else
        if es<=esh
            fs(i)=fy;
        else
            if es<esu
                fs(i)=fsu-(fsu-fy)*(((esu-es)/(esu-esh)))^P;
            else, fs(i)=0;
            end
        end
    end
else
    if es > -fy/Es
        fs(i)=es*Es;
    else, fs(i)=-fy;
    end
end
i = i +1;
end
end
```

```
function [y, Ac, Acc] = areas_circle(Section)
% Obtains fiber discretization geometry for circular section
%% Input
% A section structure with the following:
%   D is the diameter of the section
%   ds is the confined diameter of the section
%   t is the width of the concrete fiber
%   n is the total number of fibers
%% Output
% y is the distance from the bottom of the section to the centroid of each
% fiber (vector)
% Ac is the area of unconfined concrete of eachbar fiber (vector)
% Acc is the area of confined concrete of each fiber (vector)
%%

D = Section.diameter;
ds = Section.diameter_confined;
t = Section.fiber_width;
n = Section.no_of_fibers;

R = D/2;    r = ds/2;
y_coverb = (D - ds)/2;
y_covert = D - y_coverb;
y = zeros(n,1);
Ac = zeros(n,1);    Acc = zeros(n,1);

for ff = 1:n
    y(ff) = t*(ff-0.5);
    if (y(ff) > y_coverb) && (y(ff) < y_covert)
        xcc = sqrt(r^2 - (y(ff)-R)^2);
        Acc(ff) = 2*xcc*t;
    else, xcc = 0;
    end
    xc = sqrt(R^2 - (y(ff)-R)^2);
    bc = xc - xcc;
    Ac(ff) = 2*bc*t;
end
end
```

```
function[de,e]=brute_force(error,tol,de,e)
% Bisection method to update value of strain at centroid
%% Input
% error is the current value of the error
% tol is the tolerance value
% de is the current strain change
% e is the strain at the centroid at the current step (just one value, not
% the complete vector)
%% Output
% de is the updated value of strain change
% e is the updated value of strain at the centroid
if abs(error)>tol
    if (error<0 && de<0 || error>0 && de>0)
        de=de*-0.5;
        e=e+de;
    else
        e=e+de;
    end
end
end
```