Spring 1-1-2011

Implementation of Lumped Plasticity Models and Developments in an Object Oriented Nonlinear Finite Element Code

Christopher L. Segura
University of Colorado at Boulder, segurac@colorado.edu

Implementation of Lumped Plasticity Models and Developments in an Object Oriented Nonlinear Finite Element Code

by

Christopher L. Segura

B.S., Civil Engineering, University of Colorado Boulder,

Boulder, CO 2009

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Civil, Environmental and Architectural Engineering

2011

This thesis entitled:

Implementation of Lumped Plasticity Models and Developments in an Object Oriented Nonlinear Finite Element Code

written by Christopher L. Segura

has been approved for the Department of Civil, Environmental and Architectural Engineering

Prof. Victor Saouma

Prof. Franck Vernerey

Prof. Abbie Liel

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Abstract

Segura, Christopher L. (M.S., Civil, Environmental and Architectural Engineering)

Implementation of Lumped Plasticity Models and Developments in an Object Oriented Nonlinear Finite Element Code

Thesis directed by Professor Victor E. Saouma

Numerical simulation tools capable of modeling nonlinear material and geometric behavior are important to structural engineers concerned with approximating the strength and deformation capacity of a structure. While structures are typically designed to behave linear elastic when subjected to building code design loads, exceedance of the linear elastic range is often an important consideration, especially with regards to structural response during hazard level events (i.e. earthquakes, hurricanes, floods), where collapse prevention is the primary goal. This thesis addresses developments made to Mercury, a nonlinear finite element program developed in MATLAB for numerical simulation and in C++ for real time hybrid simulation. Developments include the addition of three new constitutive models to extend Mercury's lumped plasticity modeling capabilities, a constitutive driver tool for testing and implementing Mercury constitutive models, and Mercury pre and post-processing tools.

Mercury has been developed as a tool for transient analysis of distributed plasticity models, offering accurate nonlinear results on the material level, element level, and structural level. When only structural level response is desired (collapse prevention), obtaining material level results leads to unnecessarily lengthy computational time. To address this issue in Mercury, lumped plasticity capabilities are developed by implementing two lumped plasticity flexural response constitutive models and a column shear failure constitutive model. The models are chosen for implementation to address two critical issues evident in structural testing: column shear failure and strength and stiffness

degradation under reverse cyclic loading. These tools make it possible to model post-peak behavior, capture strength and stiffness degradation, and predict global collapse.

During the implementation process, a need was identified to create a simple program, separate from Mercury, to simplify the process of implementing a new constitutive model. A constitutive driver tool with a graphical user interface is developed to address this issue, providing benefits for Mercury development and classroom learning.

A Mercury pre and post-processor graphical user interface is also implemented. The developed tool is a standalone application which allows Mercury users to visualize numerical models for verification and view analysis results without the need to transfer information. Mercury analysis may also be run from within the application. All necessary pre-process, analysis, and post-process procedures are, therefore, combined into the program, collectively referred to as Mercury++.

Acknowledgements

I would like to express my appreciation to Professor Victor Saouma for his guidance, continual support, and patience as I worked to understand and develop the MATLAB based Mercury platform. I would also like to thank Dr. Dae-Hung Kang for his correspondence, helping me to gain an understanding of Mercury processes.

I would especially like to thank Mr. Siamak Sattar for producing OpenSees validation results. Also, thank you to Mr. Sattar and Professor Abbie Liel for taking the time to discuss the process of lumped plasticity modeling and implementation of the constitutive models discussed herein.

Table of Contents

List of Tables

List of Figures

Notation

A_g	gross cross-sectional area
A_{st}	transverse reinforcing steel cross-sectional area
b	member cross-section width
c	hysteretic degradation rate factor
E	Young's modulus
$E_{adjusted}$	lumped plasticity model, linear elastic element adjusted elastic modulus
E_{axial}	zero-length element, axial spring elastic (Young's) modulus
E_{bp}	break-point reloading modulus
E_c	post-capping (strain softening) modulus
E_{hys}	hysteretic energy capacity (= $\gamma\ \sigma_y\ \varepsilon_y$) as defined by Rahnama and Krawinkler (Rahnama & Krawinkler, 1993)
E_i	hysteretic energy dissipated during loading excursion "i" ($= \Delta\varepsilon \left[\sigma_{i-1} + \frac{1}{2}(\sigma_i - \sigma_{i-1})\right]$)
E_s	strain hardening modulus
E_{shear}	zero-length element, transverse spring elastic (Young's) modulus
E_t	target reloading modulus
E_{tan}	tangent modulus
E_u	unloading modulus
f'_c	concrete compressive strength
F_y	yield force
G	shear modulus
h	member cross-section height
$K_{adjusted}$	lumped plasticity model, linear elastic element adjusted initial stiffness
K_{BC}	column initial stiffness

$K_{combined}$	lumped plasticity model, linear elastic and nonlinear lumped plastic combined initial rotational stiffness
K_{deg}	zero-length element post-peak degraded stiffness defined by Elwood & Moehle (Elwood & Moehle, 2003)
K_e	initial elastic stiffness
K_{linear}	lumped plasticity model, linear elastic element initial stiffness
K_{LP}	lumped plasticity model, nonlinear rotational element initial stiffness
$K_{rotation}$	initial rotational stiffness
K_s	strain hardening stiffness
K_{shear}	initial shear stiffness
K_{stiff}	Limit State model, zero-length transverse spring initial stiffness
K^t_{deg}	post-peak degraded stiffness defined by Elwood & Moehle (Elwood & Moehle, 2003)
K_{unload}	unloading stiffness
L	element length
M_r	residual moment
M_u	ultimate (peak) moment
M_y	yield moment
P	axial force
V_{fail}	shear force at point of shear failure
V_r	residual shear strength
V_u	shear capacity
β	hysteretic dissipation parameter defined by Rahnama & Krawinkler (Rahnama & Krawinkler, 1993)
γ	hysteretic energy capacity factor ($E_{hys} = \gamma \sigma_y \varepsilon_y$)

$\gamma_{adjusted}$	lumped plasticity model, calibrated hysteretic capacity factor
Δ_{BC}	column lateral deformation
Δ_s	interstory lateral deformation at point of shear failure as defined by Elwood & Moehle (Elwood & Moehle, 2003)
Δ_{spring}	zero-length spring deformation
Δ_y	yield displacement
ε_{10}	point of inflection between negative stress unloading and positive stress reloading (Figure 3.10)
ε_5	point of inflection between positive stress unloading and negative stress reloading (Figure 3.10)
ε_{bp}	break-point strain
ε_r	residual strain
ε_t	strain at target reloading point
ε_u	strain at peak point
ε_y	yield strain
θ_r	residual rotation
θ_u	ultimate (peak) rotation
θ_y	yield rotation
κ_d	break-point determination target strain (displacement) factor
κ_f	break-point determination target stress (force) factor
λ	residual strength factor ($V_r = \lambda V_{fail}$)
ν	nominal shear strength (= V_u/A_g)
ρ''	transverse reinforcement ratio (= A_{st}/bh)
σ_{bp}	break-point stress

σ_r	residual stress
σ_{ref}	post-capping branch reference stress (Figure 3.17)
$\sigma_{ref,bp}$	break-point determination reference stress (Figure 3.10)
σ_t	stress at target reloading point
σ_u	ultimate (peak) stress
σ_y	yield stress

1 Introduction

This thesis details developments made to Mercury (Kang, 2010), a nonlinear finite element program specifically developed for transient analysis of reinforced concrete structures. A version of Mercury has been developed in MATLAB for numerical simulation and another version has been developed in C++ for pseudo-static and hard real time hybrid simulation. The developments discussed, herein, have been made to the MATLAB version.

1.1 Motivation

An entire field of research is concerned with identifying and predicting critical structural failure modes. Extensive research in this field has produced numerical simulation tools capable of capturing structural strength loss and predicting failure. Mercury has been developed as a tool for two-dimensional and three-dimensional distributed plasticity modeling using layer and fiber sections to capture nonlinear material and nonlinear geometric response (Saouma, 2011). Distributed plasticity modeling provides accurate results when geometry, material properties, and external forces are known. In the field of performance based earthquake engineering (PBEE), however, uncertain force conditions often warrant a lengthy numerical simulation process and simplification of the analysis method is often desired to shorten the process (Krawinkler, 1999). Distributed plasticity models may be very computationally intensive due to their complexity. Structural engineers, thus, often choose to simplify complex models by lumping inelastic effects at critical locations (lumped plasticity modeling) and calibrating models to produce results similar to those expected from a distributed plasticity model. This modeling choice results in time savings by providing an overall look at performance on a structural level rather than on a material level as provided by distributed plasticity modeling. Implementation of numerical simulation

tools developed for lumped plasticity modeling is vital for providing Mercury these modeling capabilities.

Use of numerical simulation tools (such as Mercury) entails attaining a certain level of familiarity with the program and its capabilities. Until the capabilities are understood, the program may be viewed as somewhat elusive and hard to work with, preventing its usage. Even when an acceptable level of comfort has been established, users may be deterred from using the program if it is difficult to verify input files and to obtain and use results. Development of visualization tools to promote program familiarity, to offer simple verification, and to provide instant access to results, therefore, is key to the development of the MATLAB based Mercury platform.

1.2 Thesis Work Outline

Chapter 2 provides a description of Mercury including element and constitutive model choices. A brief description of nonlinear modeling is provided with a discussion of distributed and lumped plasticity modeling.

The theory and Mercury implementation of two hysteresis constitutive models is presented in Chapter 3. First, the importance of implementation of such a model is discussed with a focus on the behavioral response evident in structural testing. A detailed description is provided to explain the basic response throughout the duration of several loading cycles. A description of four stiffness and strength degradation modes follows, providing an in depth explanation of the strength loss captured under reverse cyclic loading. Implementation of a new Mercury constitutive model is then explained and details are provided for implementation of the hysteresis models. Validation of the newly implemented models is performed by comparing Mercury analysis results to those of a similar OpenSees (OpenSees, 2005) model.

Chapter 4 contains the theory and Mercury implementation details for a shear failure constitutive model. Motivation for use of the shear failure model is discussed first, highlighting its benefits to both lumped and distributed plasticity modeling. Next, details of column response are provided leading up to detection of failure and the corresponding response after shear failure is detected by the model. Mercury implementation is then explained, followed by a validation section.

Chapters 5 and 6 explain the development of two visualization tools developed for Mercury: a constitutive driver and a Mercury pre and post-processor, collectively referred to as Mercury++. Chapter 5 provides an explanation of the constitutive driver as an independent tool. Information on developing GUIs in MATLAB is included in Chapter 6 as a starting point for those interested in the development process. A full description is then provided for each of the individual user interfaces used to create a Mercury input file and to obtain analysis results with the pre and post-processors.

Finally, chapter 7 includes concluding remarks and suggestions for future work.

2 Introduction to Mercury and Nonlinear Modeling

Because of the constant advancement of research, numerical simulation tools must be adaptable to keep from becoming obsolete. MATLAB is often chosen as a suitable programming language for research tools due to its simplicity as a high level language. For this reason, developments are made to Mercury in MATLAB. Recent updates to the MATLAB based Mercury platform take advantage of MATLAB's object oriented programming capabilities. The benefit of Mercury in a MATLAB object oriented format is twofold as developers are able to take advantage of the simplicity of MATLAB while also being able to easily add to object classes (i.e. new constitutive models). Implementation, into Mercury, of a new constitutive model or new element, thus, requires no maintenance to the Mercury source code.

The full release version of Mercury contains a total of seven constitutive models to simulate the response of elastic, steel, and concrete materials (Kang, 2010). Three steel constitutive models exist in Mercury:

1. Combined isotropic and kinematic hardening model
2. Simplified bilinear model with isotropic hardening
3. Modified Giuffre-Menegotto-Pinto model

In addition, three constitutive models are included for modeling concrete response:

1. Modified Kent and Park model
2. Anisotropic damage model with effective damage and stiffness recovery
3. Anisotropic damage model with effective damage, stiffness recovery, and permanent strains

Mercury also contains seven distinctive element choices. The elements are divided amongst truss elements, beam-column elements, and zero-length lumped plasticity elements (Kang, 2010).

1. Truss element (truss)
2. Elastic beam-column element (beam-column)
3. Stiffness-based beam-column element (beam-column)
4. Flexibility-based beam column element (beam-column)
5. Flexibility-based beam column element with element iterations (beam-column)
6. Zero-length uniaxial element (lumped plasticity)
7. Zero-length section element (lumped plasticity)

As such, Mercury yields results at the material (section) level, element level, and structural level, accounting for nonlinearities over the entire length of elements (Saouma, 2011). An example of a two-dimensional distributed plasticity model is shown in Figure 2.2 for the reinforced concrete cantilever column shown in Figure 2.1.

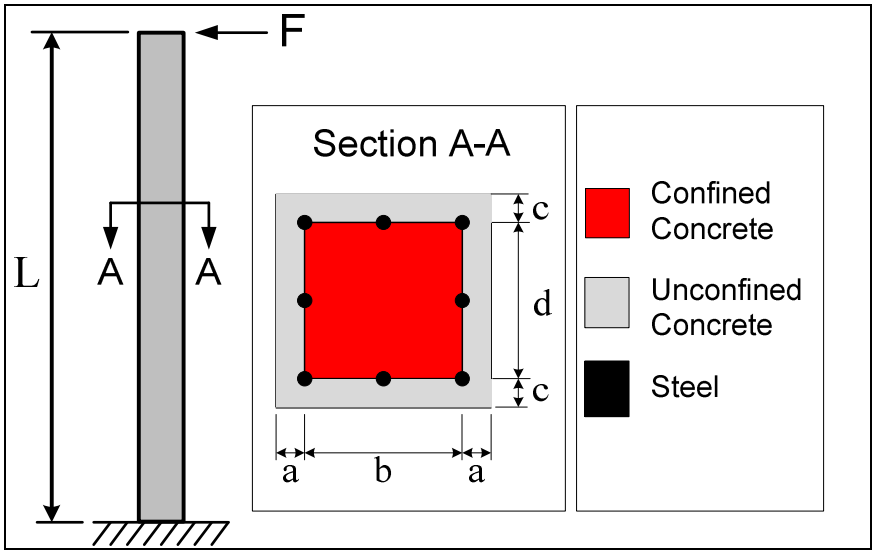

Figure 2.1 Mercury Example Model

Using layered sections, the cross-sectional properties are divided amongst unconfined cover concrete, confined concrete, and reinforcing steel (Figure 2.2). A single flexibility-based element or several identical stiffness-based elements are used for the Mercury model. Assigning the Modified Kent and Park constitutive model for confined and unconfined concrete and the Modified Giuffre-Menegetto-Pinto model for reinforcing steel, material nonlinearities are captured for each layer along the length of the column.

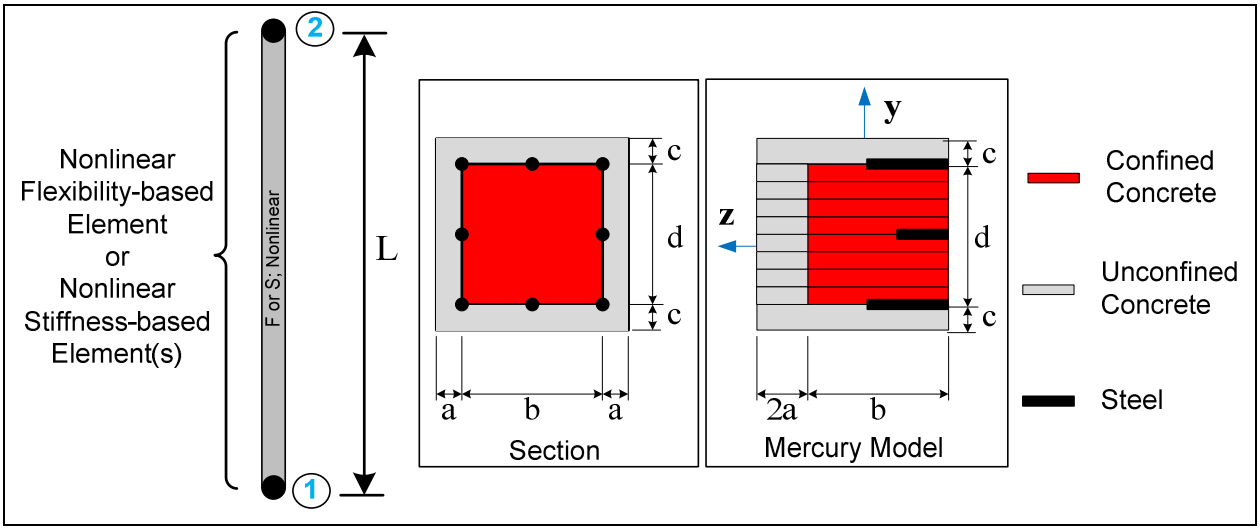

Figure 2.2 Mercury 2-D Layer Section Model

As a comparison, a two-dimensional lumped plasticity model is shown in Figure 2.3 for the same cantilever column (Figure 2.1). Zero-length lumped plasticity elements are used to model nonlinear behavior at the base and at the top of the column while beam-column elements are modeled as linear elastic along the element length (Saouma, 2011). The initial stiffness of the lumped plasticity element is set to a value much larger than that of the linear elastic element and the stiffness of the linear elastic element is adjusted to preserve pre-yield behavior. As a result, the combined stiffness of the elastic element and lumped plasticity element is comparable to that of the distributed plasticity model. See Chapter 3.5 for a more in-depth discussion about lumped plasticity modeling.

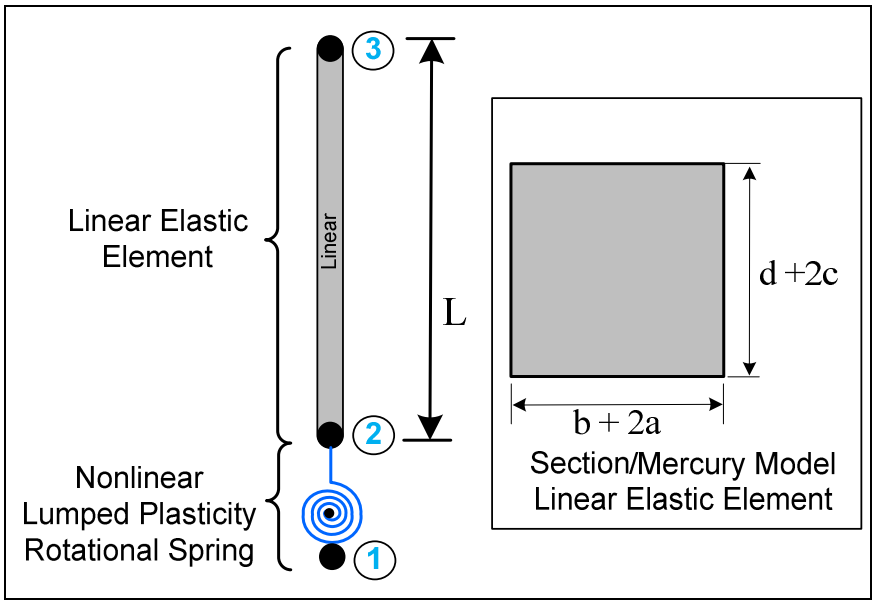

Figure 2.3 Lumped Plasticity Model

3 Ibarra Hysteretic Models Incorporating Energy Dissipation

This chapter introduces several constitutive models for capturing strength and stiffness degradation that results from reverse cyclic loading, referred to as hysteresis models. A thorough description is then provided for two reinforced concrete hysteresis models chosen for implementation into Mercury, based on the research of Ibarra (Ibarra, 2003): the peak-oriented model (Chapter 3.2.1) and the pinching model (Chapter 3.2.2). They are differentiated only by the inclusion of "pinching" behavior along the reloading branch (see Figure 3.5). The implemented peak-oriented and pinching models include four cyclic strength degradation modes (Chapter 3.3), presented by Ibarra as the most critical sources of strength loss evident in structural laboratory testing.

3.1 Introduction

A hysteretic constitutive model is one that captures certain material responses that occur as a structural element is subjected to cyclic loads. Some idealized hysteretic responses are: elastic loading, strain hardening, elastic unloading, and elastic reloading. A strain hardening constitutive model, for example, captures each of these idealized responses over the duration of a single loading cycle. In Figure 3.1, a hysteresis cycle is shown for a strain hardening material. Elastic loading is shown between points 0 and 1, followed by strain hardening from 1-2, then elastic unloading (2-3) upon a strain reversal. Elastic reloading (3-4) is followed by strain hardening reloading (4-5), elastic unloading (5-6), elastic reloading (6-7), and strain hardening reloading (7-8).

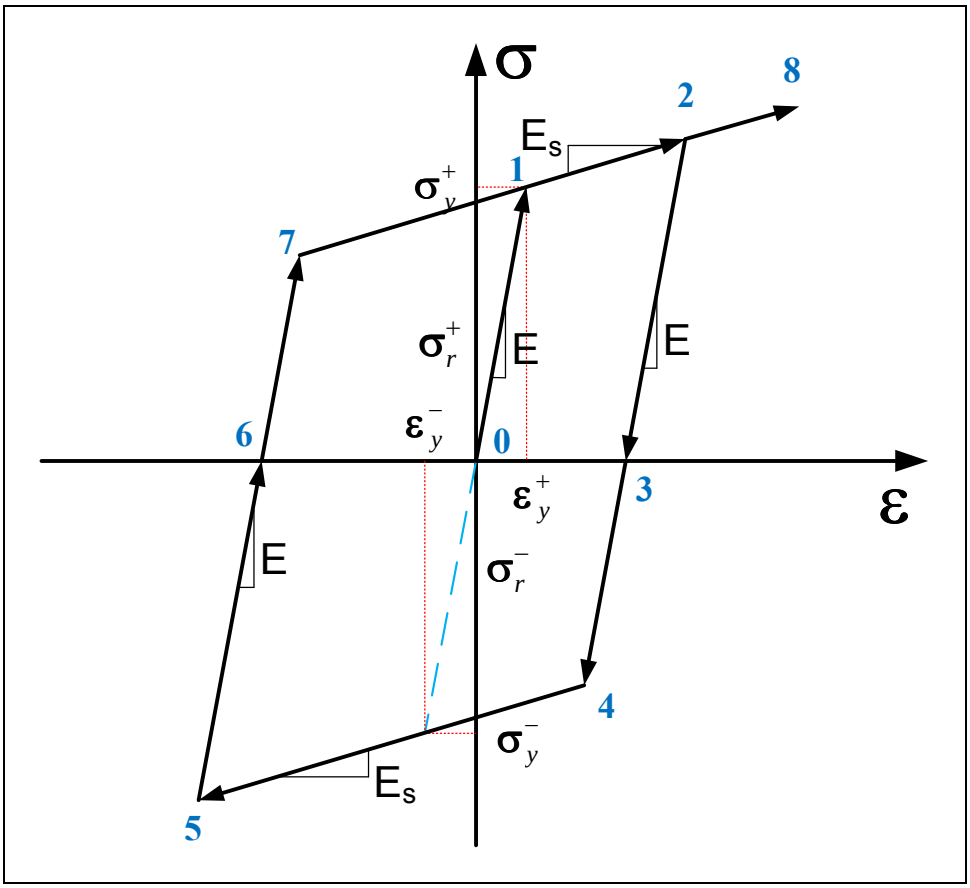

Figure 3.1 Hysteresis Cycle

This idealized response pattern, however, does not capture hysteretic behavior typical of most materials, especially that of reinforced concrete. Over the course of one or several hysteresis cycles, idealization of elastic unloading and elastic reloading, for example, is not an accurate assumption.

An important aspect of numerical simulation is capturing behavioral response that occurs as a structure approaches collapse. While most structures undergo some post-yield strengthening (i.e. strain hardening), the peak-strength point is typically followed by strength degradation leading to the point of collapse. This type of strength loss is shown as a strain softening branch (Figure 3.2) from the peak point (point 2) to the point of failure (point 3), resulting from highly inelastic monotonic loading. An effective hysteretic constitutive model should be capable of capturing this behavior.

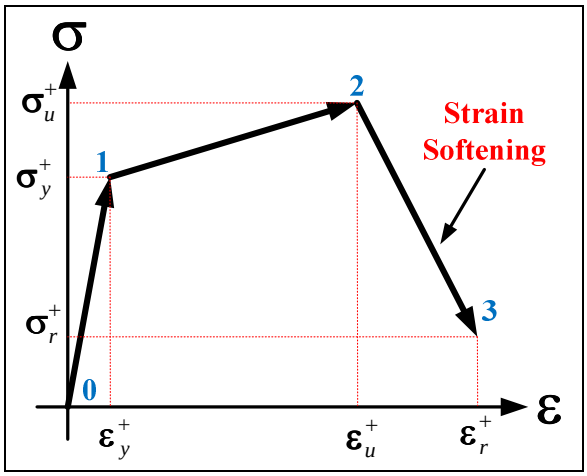

Figure 3.2 Strain Softening Strength Loss

Research indicates strength capacity degradation occurs over the duration of a reverse cyclic loading history, regardless of material, signifying the importance of a computational model capable of capturing this hysteretic strength degradation. This type of strength loss results in constitutive model strength indicator points (i.e. peak strength point, yielding point) translating toward the origin over the course of a cyclic loading history, leading to failure (Saouma, 2011). Figure 3.3 details this process, indicating degraded strength properties at a point X as X' and X". It is important to point out that the yield points (3 and 6), peak strength points (3 and 8) and residual strength points (4 and 9) translate toward the origin causing the point of failure to occur at a strain value less than that of the original residual strain.

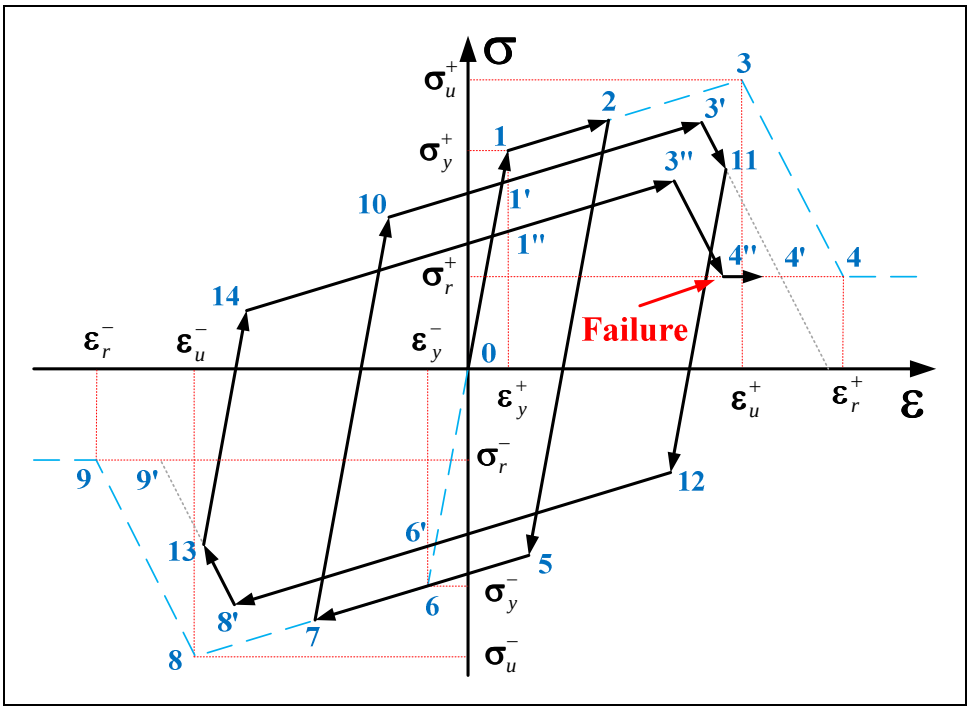

Figure 3.3 Reverse Cyclic Strength Degradation

Several models have been developed to capture hysteretic strength loss. They can generally be divided into two groups: piecewise linear models and smooth models with continuously changing stiffness. The Bilinear and Modified Kent and Park models implemented in Mercury (Kang, 2010) are piecewise linear models capable of capturing some hysteretic behavioral characteristics. The Bilinear model, however, does not capture post-peak behavior and no strength deterioration is accounted for under reverse cyclic loading. While the Modified Kent and Park model *does* captures post-peak behavior, strength deterioration is not accounted for by the model. The models proposed by Clough (Clough, 1966) and Takeda (Takeda et al., 1970) are piecewise linear models that account for reloading stiffness and unloading stiffness, respectively, based on the maximum deformation experienced in the direction of loading in previous cycles. The Wen-Buoc model (Wen, 1976) is a smooth model capable of capturing stiffness deterioration. Sivaselvan and Reinhorn (Sivaselvan & Reinhorn, 2000) proposed a smooth model incorporating stiffness and strength degradation; however, the model does not include a post-

peak softening branch. Each model effectively captures characteristics apparent in structural testing, and a combination of these characteristics ultimately lays the foundation for the hysteretic models implemented in Mercury. In order to provide these capabilities, two constitutive models are implemented, based on the research of Ibarra (Ibarra, 2003), as computational tools for predicting structure response as a function of hysteretic energy dissipated throughout the duration of the loading cycles.

Three hysteretic models are presented by Ibarra, one of which has not been chosen for implementation. The first is a modified bilinear model, following similar hysteretic rules as those of the *Bilinear* model already implemented in Mercury (not including the strength and stiffness degradation modes presented in Chapter 3.3). The modified bilinear model has not been implemented in Mercury. The other two models are modifications of Clough's model, referred to as the peak-oriented model, differentiated from one another only by the inclusion of pinching effects in the so-called pinching model. The two models are implemented in Mercury, specifically for simulating the hysteretic behavior of reinforced concrete. Both the peak-oriented and pinching models follow a similar backbone (Figure 3.4) from which strength is deteriorated. The backbone follows an initial elastic branch up to the yield point (0-1), a strain hardening branch to the peak (ultimate) strength point (1-2), a softening branch to the ultimate deformation capacity (2-3), and a residual strength branch beyond the point of deformation capacity (3-4). The backbone represents the loading path under monotonic loading, when no hysteretic strength loss occurs. For the purpose of illustration, the following figures express the constitutive models in terms of stress-strain; however, the description applies to force-displacement and moment-curvature relations as well. The subscripts *r*, *y*, and *u* correspond to the residual, yield, and ultimate capacities respectively. The subscript *s* corresponds to the strain hardening branch and the subscript *c* corresponds to the post-capping (softening) branch.

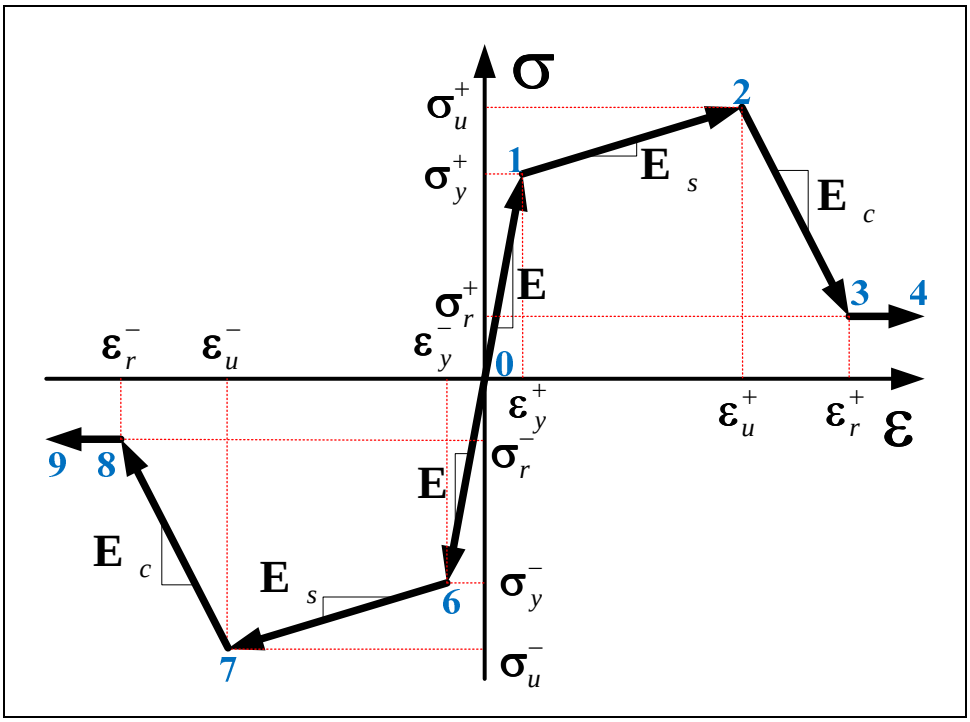

Figure 3.4 Hysteresis Model Backbone

The stress-strain relation is separated into four regions for both the positive and negative loading paths: an elastic branch, a strain hardening branch, a strain softening branch, and a residual strength branch. A summary of the monotonic backbone is presented in Table 3-1, including determination of the tangent modulus.

Path	Description	Loading in Positive Direction		Loading in Negative Direction	
		E_{tan}	Strain (ε)	E_{tan}	Strain (ε)
0- 1 & 0- 6	Elastic Loading	$E^+ = \frac{\sigma_y^+}{\varepsilon_y^+}$	$0 \le \varepsilon < \varepsilon_y^+$	$E^- = \frac{\sigma_y^-}{\varepsilon_y^-}$	$\varepsilon_y^+ < \varepsilon < 0$
1- 2 & 6- 7	Strain Hardening	$E_s^+ = \frac{(\sigma_u^+ - \sigma_y^+)}{(\varepsilon_u^+ - \varepsilon_y^+)}$	$\varepsilon_y^+ \le \varepsilon < \varepsilon_u^+$	$E_s^- = \frac{(\sigma_u^- - \sigma_y^-)}{(\varepsilon_u^- - \varepsilon_y^-)}$	$\varepsilon_u^- < \varepsilon \le \varepsilon_y^-$
2- 3 & 7- 8	Strain Softening	$E_c^+ = \frac{(\sigma_r^+ - \sigma_u^+)}{(\varepsilon_r^+ - \varepsilon_u^+)}$	$\varepsilon_u^+ \le \varepsilon < \varepsilon_r^+$	$E_c^- = \frac{(\sigma_r^- - \sigma_u^-)}{(\varepsilon_r^- - \varepsilon_u^-)}$	$\varepsilon_r^- < \varepsilon \le \varepsilon_u^-$
3- 4 & 8- 9	Residual Strength	0	$\varepsilon \ge \varepsilon_r^+$	0	$\varepsilon \le \varepsilon_r^-$

Table 3-1 Tangent Modulus

3.2 Hysteresis Models

Figure 3.5 displays the basic cycle path of the peak-oriented and pinching models adopted for implementation in Mercury. Similarities between the models are labeled as black arrows, and differences as blue (peak-oriented) and red (pinching) arrows. The models are bounded by the monotonic backbone. The first cycle displays elastic loading followed by strain hardening. Upon a strain reversal, elastic unloading occurs (2 to 3). Both models operate under target reloading where the reloading path aims to the state of maximum deformation in all previous cycles. Because deformation has not yet occurred in the negative stress direction, target loading for the first cycle aims to the negative yield point (3-5). For the peak-oriented model the reloading path directly targets this maximum deformation state (3-5). For the pinching model a pinching "break-point" (4) is first targeted. The break-point represents the point at which sectional cracks are filled and some frictional resistance is restored. Beyond the break-point the reloading path aims to the target state (4-5).

Continuing with loading beyond the target state, strain hardening occurs (5-6) and a strain reversal results in elastic unloading (6-7). Target reloading on the positive side then aims to the point of maximum deformation in previous cycles (2). For the peak-oriented model the reloading path directly targets the maximum deformation state (7-2). For the pinching model reloading first aims to the break-point (7-8) and then directs to the target state (8-2). Determination of the break-point is discussed in Section 3.2.2.

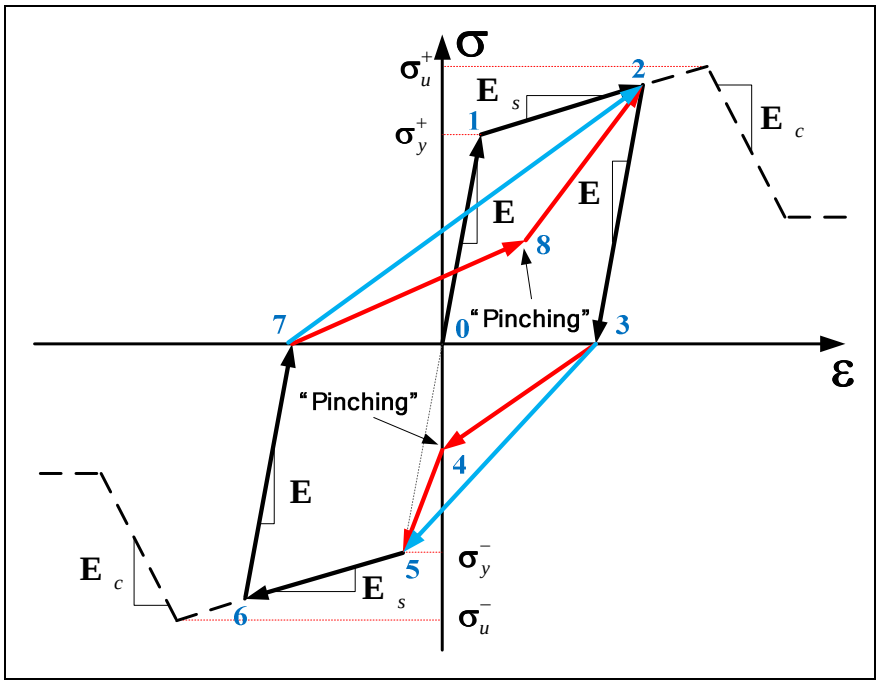

Figure 3.5 Peak-Oriented and Pinching Models

3.2.1 Peak-Oriented Model

The basic cyclic loading path rules for the peak-oriented model are shown in Figure 3.6. The term "peak-oriented" refers to the point of maximum deformation in previous loading cycles (points 2 and 6 in Figure 3.5). Reloading, in the peak-oriented model, occurs straight to this point. Referring to Figure 3.6, elastic loading occurs from 0-1, followed by loading along the strain hardening branch from 1-11. A strain reversal results in elastic unloading between 11 and 5. Cyclic loading is directed toward a target state representative of the stress-strain state at the maximum deformation experienced in all previous cycles (peak-oriented maximum deformation point). Initially, the target state is the yield point. Because yielding has occurred in the positive loading direction (point 1), upon a strain reversal, the target state is stored as the point at which the reversal occurred (point 11) *if this point exceeds the deformation in all previous loading cycles*. Reloading, for the first excursion in the negative loading direction, targets the yield point (point 6). Degraded reloading occurs between 5 and 6, displaying the stiffness degradation inherent in the peak-oriented model. Loading along the strain hardening branch (6-13) is followed by a

strain reversal and elastic unloading (13-10). The target state in the negative loading direction is stored at 13. Loading for the subsequent cycle aims to this target state. Reloading along the positive branch, thus, targets point 11. The subscript *t* corresponds to the target state. It is important to point out that the target loading modulus (E_t) should not be confused with the tangent modulus (E_{tan}); although, when loading directly to the target state, they are the same.

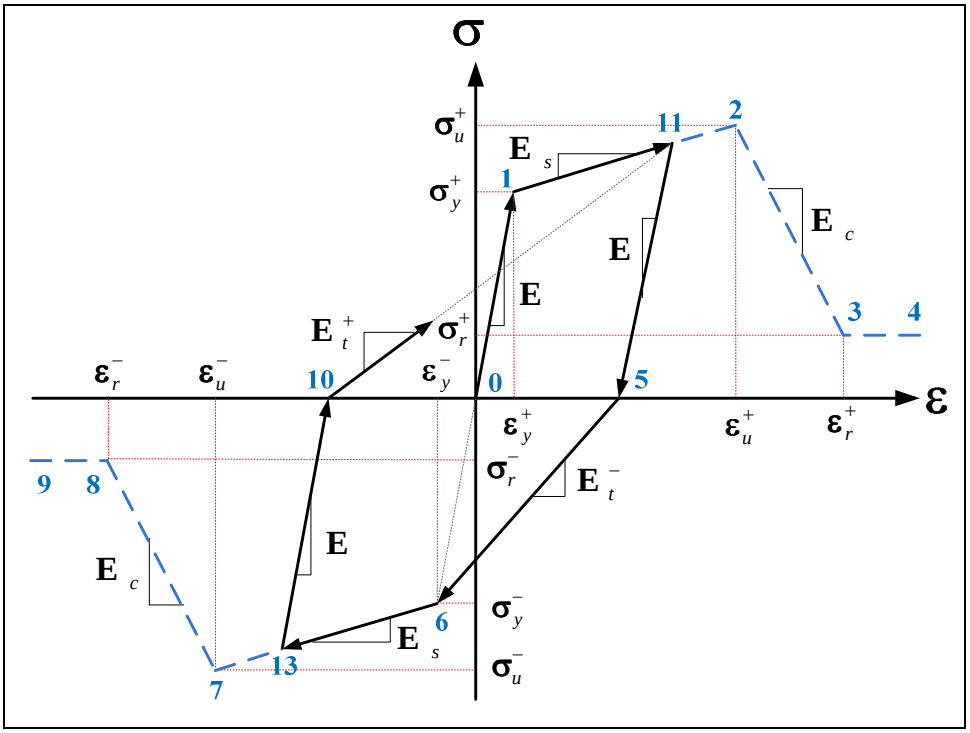

Figure 3.6 Peak-Oriented Model Basic Cycle

Referring to Figure 3.7, when a strain reversal occurs on the positive branch (point 11'), the positive target state is updated if the deformation at the reversal exceeds the previous maximum deformation state. Elastic unloading occurs (11'-5'), and degraded reloading occurs toward the negative target point (13).

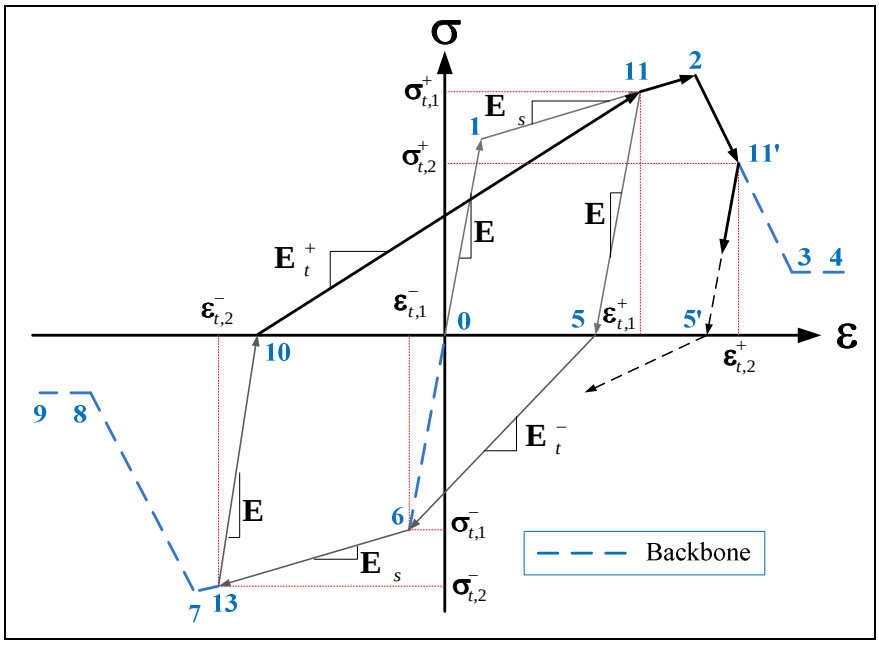

Figure 3.7 Load Reversal and Target State Update

Over a cyclic loading history increasing in magnitude, the peak-oriented model follows the monotonic backbone towards the residual stress branch as shown in Figure 3.8.

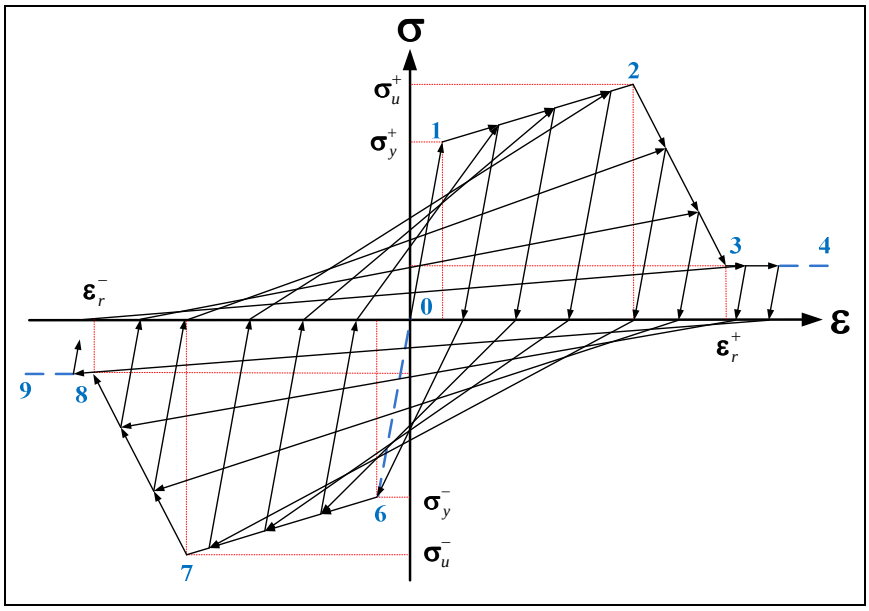

Figure 3.8 Increasing Cyclic Load History Following Peak-Oriented Model

3.2.2 Pinching Model

The pinching model accounts for reloading behavior inherent to reinforced concrete. Evident in structural testing, as sectional cracking occurs during inelastic loading, a lower frictional resistance is provided in the direction of loading as cracks remain open. As loading continues and cracks are allowed to fill, some frictional resistance is restored. This results in a "pinched strength" reloading branch resulting from the reduced frictional resistance. The pinching model follows the same monotonic backbone as the peak-oriented model and the same "peak-oriented" reloading, targeting the point of maximum deformation in previous cycles. Reloading does not directly target the maximum deformation state, however. Instead, as sectional cracks fill, the tangent modulus targets a so-called pinching "break-point" (points 14 and 15 in the negative and positive loading directions, respectively). The break-point represents the point at which sectional cracks are filled and some frictional resistance is restored. Beyond the break-point, the tangent modulus aims to the target state. Similar to the peak-oriented model, elastic (0-1) and strain hardening (1-11) loading is followed by a strain reversal and elastic unloading (11-5). The positive target state is stored at point 11. The degraded reloading first targets the negative break-point (14) at which point some of the original stiffness (modulus) is restored and reloading aims to the target point (6), the yield point during the first excursion. Loading follows along the strain hardening branch (6-13) and when a strain reversal occurs, the negative target state is updated (13) and elastic unloading occurs between 13 and 10. Reloading in the positive direction targets the positive break-point (10-15) first, followed by strengthened reloading to the target (15-11).

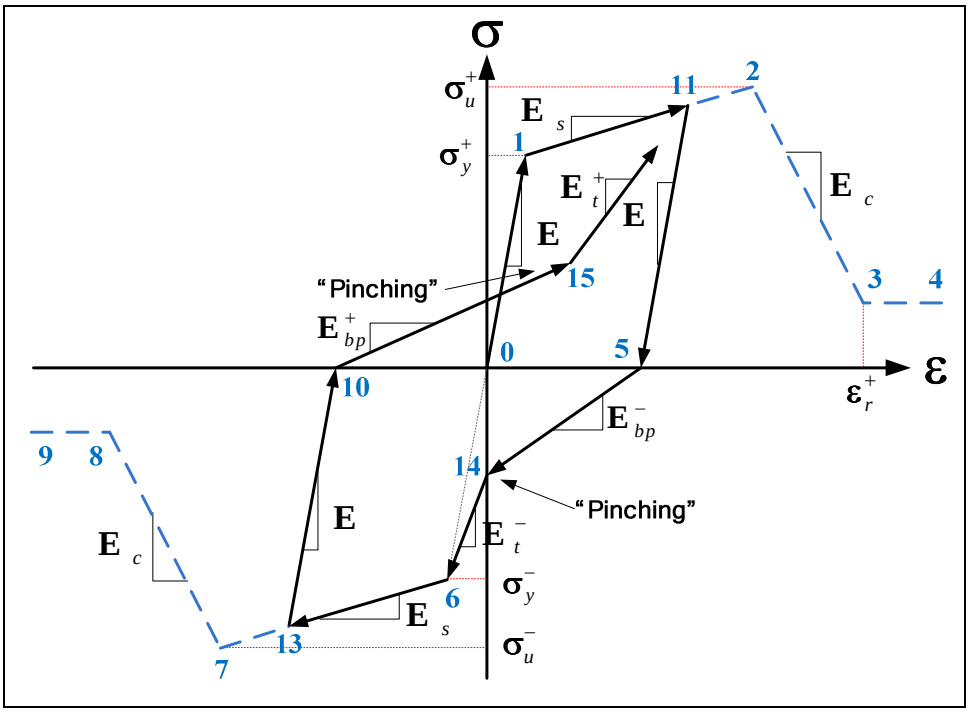

Figure 3.9 Pinching Model Basic Cycle

The break-point state is determined using the parameters κ_f and κ_d, which factor the target stress and the strain at the inflection point between unloading and reloading in the opposite direction (ε_5 and ε_{10}), respectively. The break-point strain is determined directly from the target strain by (3.1).

$$\varepsilon_{bp}^{+/-} = (1-\kappa_d)\varepsilon_{5/10}^{+/-} \tag{3.1}$$

The break-point stress is determined by extrapolating a reference stress ($\sigma_{ref,bp}^{+/-}$), at the target strain (Figure 3.10). The reference value is determined by (3.2).

$$\sigma_{ref,bp}^{+/-} = \kappa_f \sigma_t^{+/-} \tag{3.2}$$

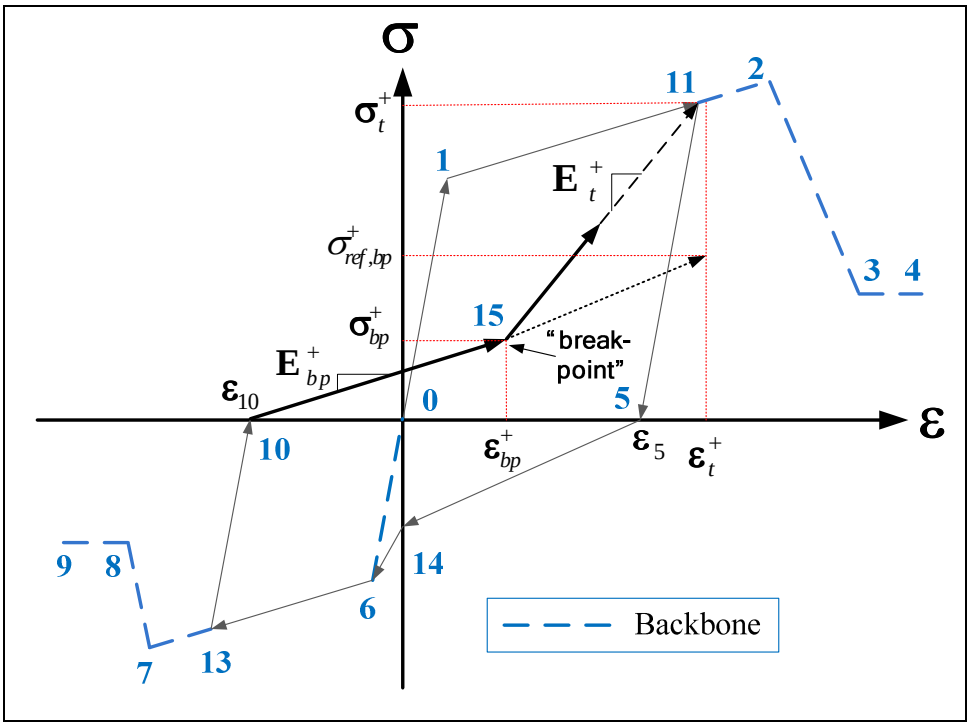

Figure 3.10 Break-Point Determination

For clarity, determination of the break-point values on the positive loading branch (point 15) is explained first. The positive break-point strain and break-point reference stress are determined by (3.3) and (3.4).

$$\varepsilon_{bp}^{+} = \left(1 - \kappa_d\right) \varepsilon_5^{+} \tag{3.3}$$

$$\sigma_{ref,bp}^{+} = \kappa_f \sigma_t^{+} \tag{3.4}$$

The tangent modulus to the positive break-point (E_{bp}^{+}) is determined by (3.5) and used to compute the break-point stress (σ_{bp}^{+}).

$$E_{bp}^{+} = \frac{\sigma_{ref,bp}^{+}}{\left(\varepsilon_t^{+} - \varepsilon_{10}\right)} \tag{3.5}$$

$$\sigma_{bp}^{+} = E_{bp}^{+}(\varepsilon_{bp}^{+} - \varepsilon_{10}) \tag{3.6}$$

Then, the tangent modulus to the target state is determined using the stress-strain state at the break-point.

$$E_{t}^{+} = \frac{(\sigma_{t}^{+} - \sigma_{bp}^{+})}{(\varepsilon_{t}^{+} - \varepsilon_{bp}^{+})} \tag{3.7}$$

Determination of the negative break-point state values follows the same process; however, it is important to once again point out that the break-point values are a function the target state, dependent on the maximum directional deformation experienced in previous cycles. Since inelastic loading was first reached on the positive stress branch (point 11), inelastic deformation has not yet occurred on the negative branch, thus the target state is the negative yield state (point 6) until yielding deformation has been exceeded in the negative direction. In this case, the first break-point is determined for the negative loading direction and the break-point, therefore, falls on the vertical axis, corresponding to zero strain.

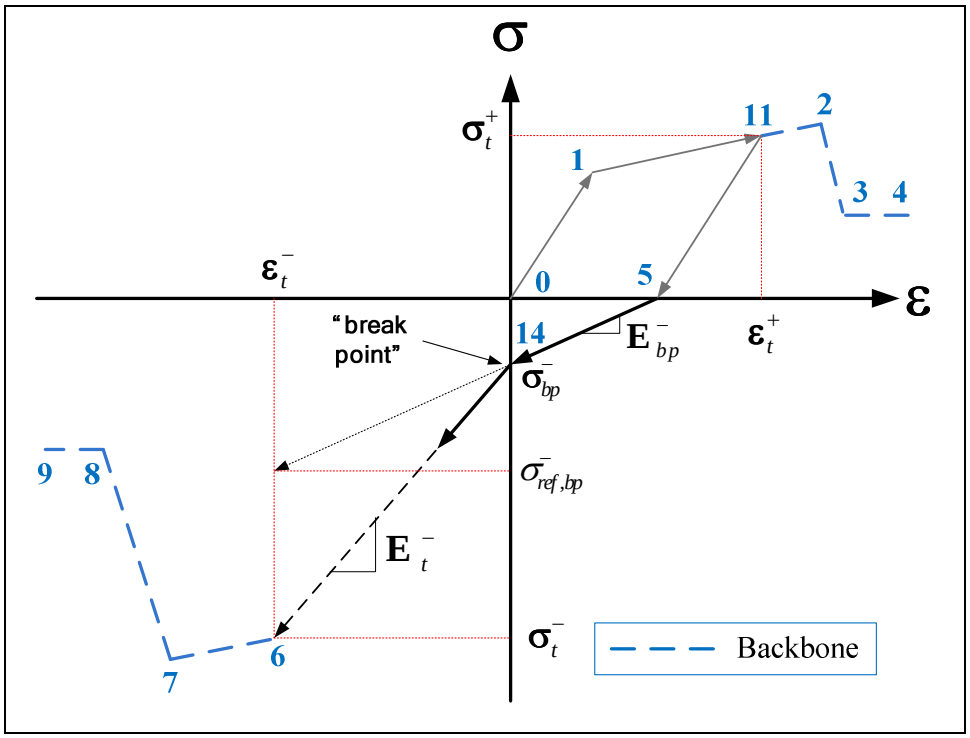

Figure 3.11 Determination of First Break-Point

Using this information the first break-point stress-strain values are computed below.

$$\varepsilon_{bp}^- = 0 \tag{3.8}$$

$$\sigma_{ref,bp}^- = \kappa_f \sigma_t^- \tag{3.9}$$

$$E_{bp}^- = \frac{\sigma_{ref,bp}^-}{(\varepsilon_t^- - \varepsilon_5)} \tag{3.10}$$

$$\sigma_{bp}^- = -E_{bp}^- \varepsilon_5 \tag{3.11}$$

During the next cycle (Figure 3.12), when unloading occurs on the positive stress branch (11'-5') and reloading begins on the negative branch (5'-14'), the break-point values are determined as a function of

the negative target state (point 13) following the process described for determination of the positive break-point.

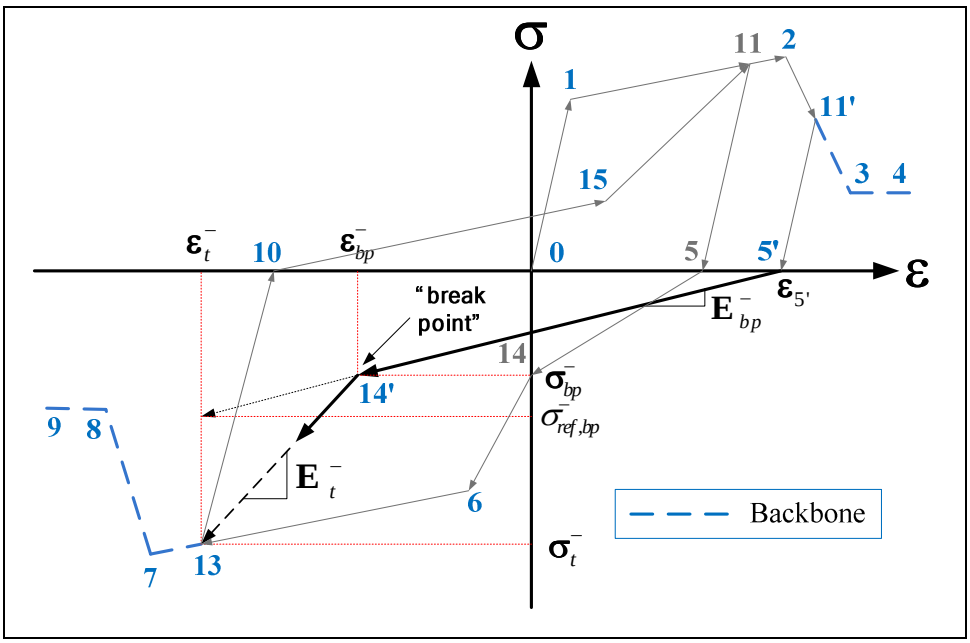

Figure 3.12 Break-Point Update

$$\varepsilon_{bp}^- = (1-\kappa_d)\varepsilon_{10}^- \tag{3.12}$$

$$\sigma_{ref,bp}^- = \kappa_f \sigma_t^- \tag{3.13}$$

$$E_{bp}^- = \frac{\sigma_{ref,bp}^-}{(\varepsilon_t^- - \varepsilon_{5'})} \tag{3.14}$$

$$\sigma_{bp}^- = E_{bp}^-(\varepsilon_{bp}^- - \varepsilon_{5'}) \tag{3.15}$$

$$E_t^- = \frac{(\sigma_t^- - \sigma_{bp}^-)}{(\varepsilon_t^- - \varepsilon_{bp}^-)} \tag{3.16}$$

When a strain reversal occurs along the reloading path (Figure 3.13) and elastic unloading (16-10') leads to reloading beyond the break point (15), reloading aims directly to the target state (11').

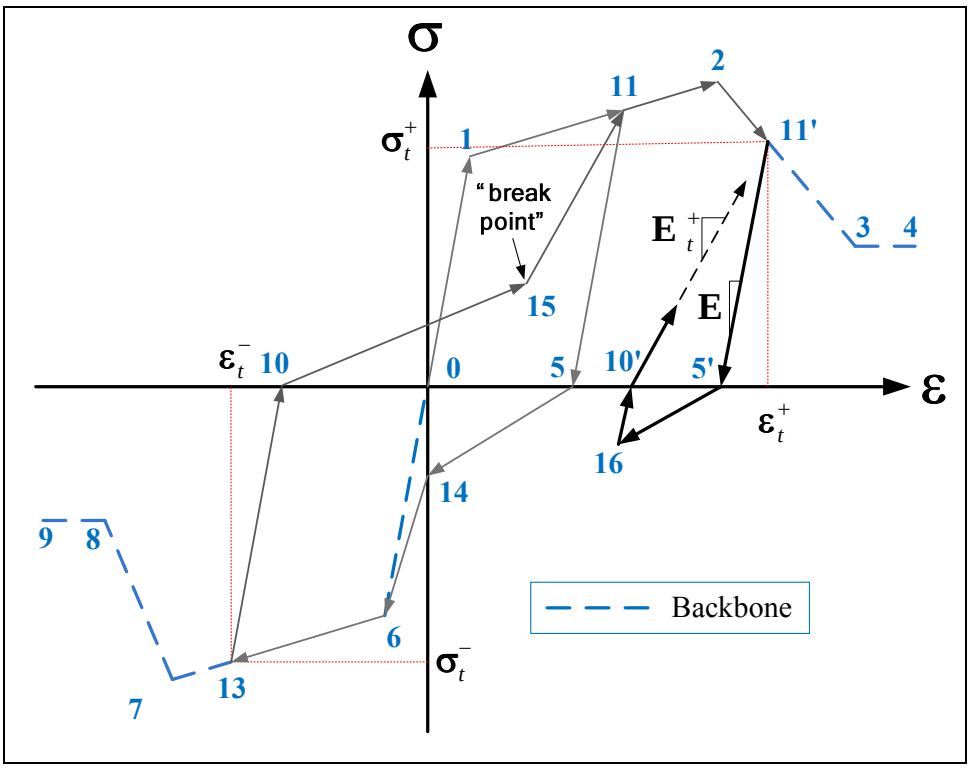

Figure 3.13 Reloading Beyond Break-Point

Over a cyclic loading history increasing in magnitude, the pinching model follows the monotonic backbone towards the residual stress through the "pinched strength" path shown in Figure 3.14.

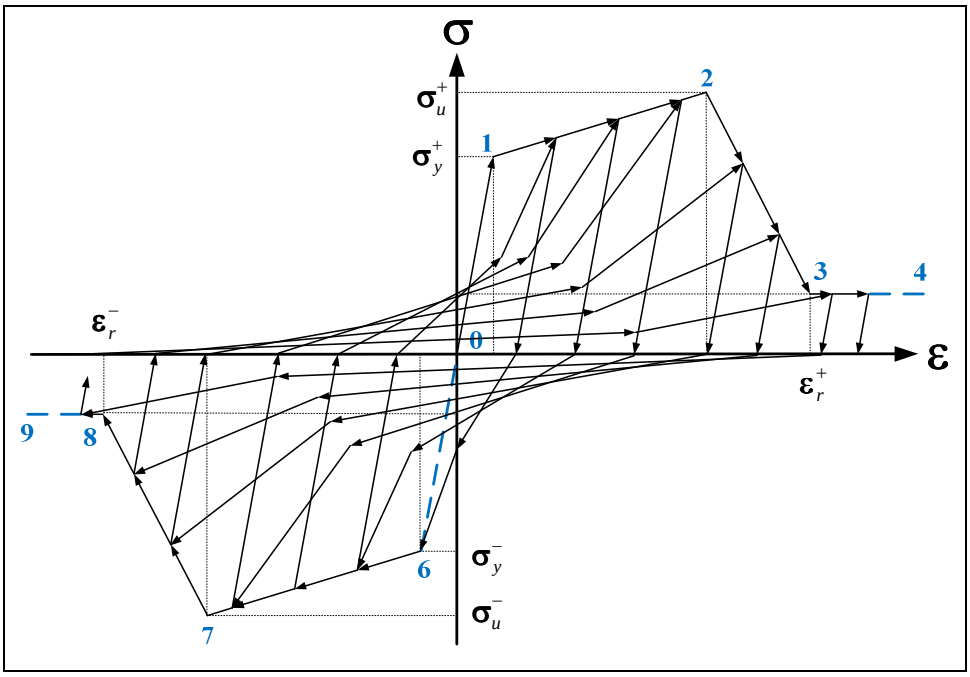

Figure 3.14 Increasing Cyclic Load History Following Pinching Model

3.3 Strength Degradation Modes

The hysteretic models introduced in Chapter 3.2 are effective models for simulating post-yielding behavior. Laboratory results indicate, however, that as a structure is subjected to reverse cyclic loads, the structure undergoes strength loss and reloading typically does not occur fully to the monotonic backbone. To account for this behavior, four strength degradation modes are proposed by Ibarra (Ibarra, 2003). It is important to point out that the strength degradation modes presented in this section are intended to be used in conjunction with a hysteretic constitutive model and not as standalone model(s). As such, each of these four degradation modes is included in the peak-oriented and pinching models implemented in Mercury.

The strength degradation modes account for structural capacity loss due to:

1. Yield strength (basic strength) degradation
2. Peak strength (post-capping) degradation

3. Unloading stiffness (modulus) degradation
4. Reloading stiffness (modulus) degradation

Figure 3.15 provides results from tests performed at the University of California, San Diego (Ibarra, 2003). The blue line presents results of a monotonic loading test, defining the monotonic backbone. Reverse cyclic testing results are plotted in red and indicators are included to describe the effects of each of the four degradation modes. Degradation mode 1 displays a decreasing yield point as the structure endures cyclic excursions. Mode 2 displays a translation in the softening branch, directly affecting the peak strength. Mode 3 displays deterioration of the unloading stiffness (modulus) and mode 4 displays a strain shift.

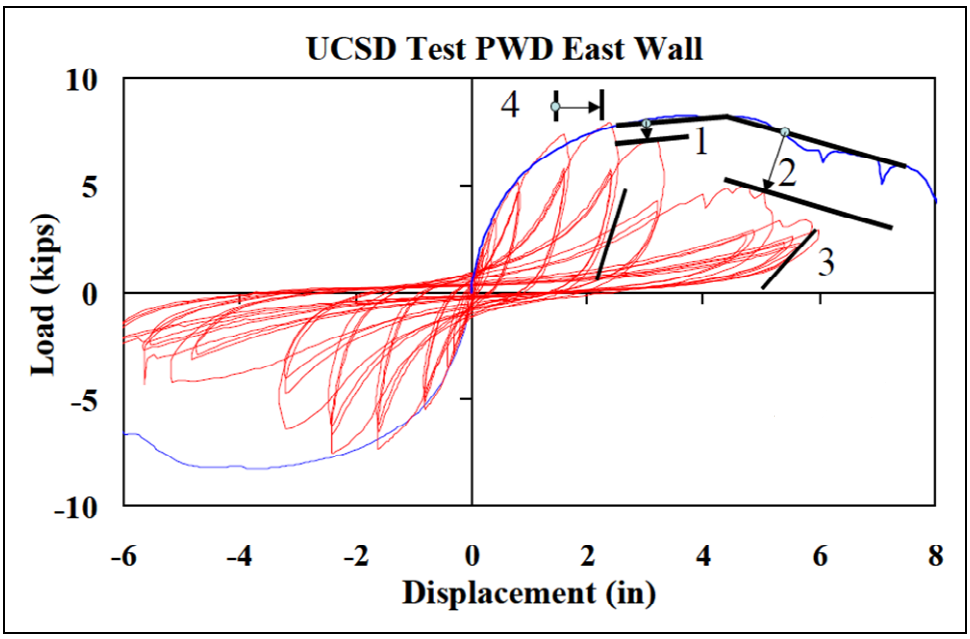

Figure 3.15 UCSD Structural Laboratory Results(Ibarra, 2003)

Each of the four degradation modes has been implemented using the hysteretic dissipation parameter β introduced by Rahnama and Krawinkler(Rahnama & Krawinkler, 1993) to degrade a user defined hysteretic energy capacity (E_{hys}). The hysteretic capacity is defined as a factor of the energy necessary to reach yielding.

$$\beta_i = \left(\frac{E_i}{E_{hys} - \sum_{j=1}^{i} E_j} \right)^c \tag{3.17}$$

$$E_{hys} = \gamma \sigma_y \varepsilon_y \tag{3.18}$$

where

E_i = hysteretic energy dissipated over loading excursion "i"

$\sum E_j$ = hysteretic energy dissipated over all previous loading excursions

c = degradation rate parameter

γ = hysteretic energy capacity factor

A loading excursion begins when the horizontal axis is met and ends when the horizontal axis crossed again. Therefore, separate loading excursions occur on the positive and negative stress branches and are used to update strength parameters on the opposite branch. The dissipation parameter is initially a small value ($\beta_i \ll 1$), increasing as energy is dissipated throughout a loading history. As the total energy dissipated approaches the reference capacity, the dissipation parameter approaches infinity indicating structural collapse. Over the duration of the loading history, values of β should lie within the range of $0 \leq \beta_i \leq 1$ for computational purposes. Values below zero and greater than one indicate that the reference hysteretic energy capacity has been exhausted and structural capacity degradation will result in loss of structural load carrying capacity. Rahnama and Krawinkler suggest a range $1 \leq c \leq 2$ for the deterioration rate parameter.

The degradation modes begin to deteriorate structural capacity once yielding has been achieved in at least one direction. Updates are made for each of the modes, with the exception of the unloading

stiffness mode, when the horizontal axis of the model is crossed, corresponding to a change from positive to negative stress or vice versa. Updates are made independently for each stress state (positive or negative) when loading crosses the horizontal axis and proceeds in that direction of loading. For example, when unloading occurs on the positive branch and loading begins on the negative branch, updates are made for basic strength capacity, post-capping capacity, and reloading stiffness (modulus) on the negative side. Updates for the unloading stiffness (modulus), on the other hand, are made when a strain reversal occurs. Figures presented for the degradation modes highlight the pinching model, although the idea is extended to the peak-oriented model by eliminating the break-points and reloading directly to the target points.

3.3.1 Basic Strength Degradation Mode

The basic strength degradation mode accounts for the reduction in strength as reloading occurs toward the strain hardening branch, physically representative of mode 1 in Figure 3.15. It shifts the strain hardening branch (Figure 3.16) toward the origin by reducing the yield stress (1 to 1') and the hardening modulus ($E_{s,o}^{\mp}$ to $E_{s,1}^{\mp}$) using the dissipation parameter β_s. Updates are made to the yield stress and hardening modulus when the horizontal axis is crossed (point 5 or point 10) for the direction of loading only. Again, the figures presented in this section detail incorporation of the degradation modes into the pinching model, although, they are similarly implemented in the peak-oriented model. The only difference is the additional reloading branch that targets the pinching break-points.

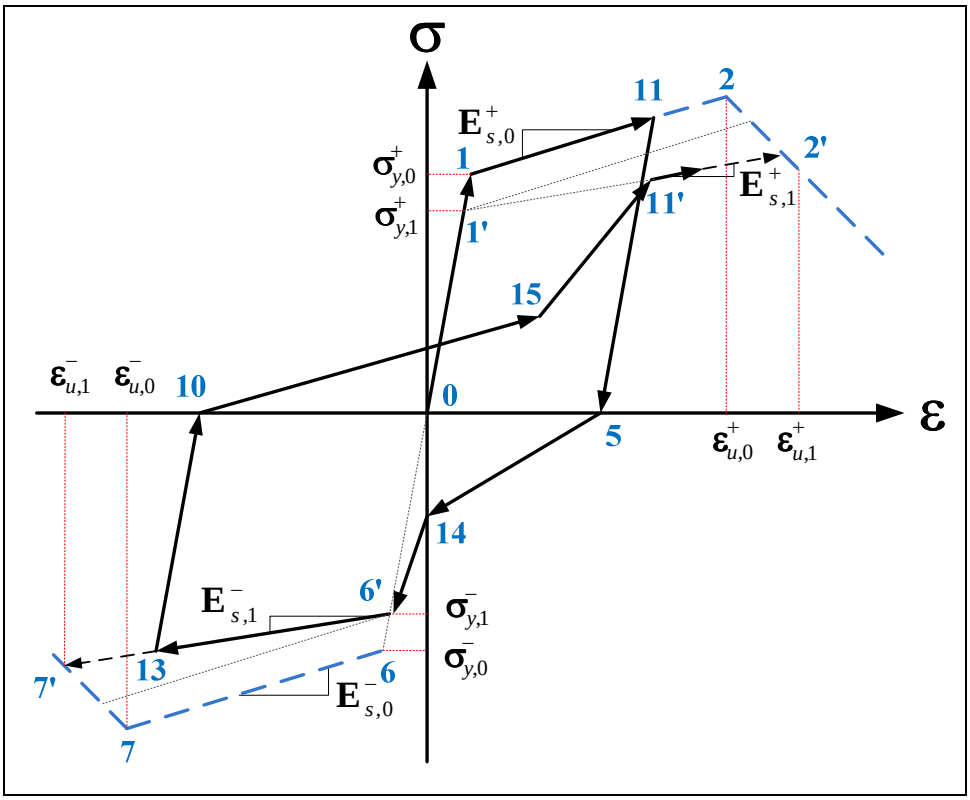

Figure 3.16 Basic Strength Degradation Mode

The degraded values are determined by (3.19) and (3.20).

$$\sigma_{y,i}^{+/-} = (1-\beta_{s,i})\sigma_{y,i-1}^{+/-} \tag{3.19}$$

$$E_{s,i}^{+/-} = (1-\beta_{s,i})E_{s,i-1}^{+/-} \tag{3.20}$$

where

$\beta_{s,i}$ = basic strength dissipation parameter for time step "i"

$\sigma_{y,i-1}^{+/-}$ = directional yield stress for time step "i-1"

$\sigma_{y,i}^{+/-}$ = directional yield stress for time step "i"

$E_{s,i-1}^{+/-}$ = directional strain hardening modulus for time step "i-1"

$E_{s,i}^{+/-}$ = directional strain hardening modulus for time step "i"

As previously stated, independent updates are made to positive and negative values during the step when loading crosses the horizontal axis and begins in that direction. The basic strength dissipation parameter ($\beta_{s,i}$) makes use of a hysteretic capacity related to basic strength, determined by the hysteretic capacity factor γ_s.

$$E_{hys,s} = \gamma_s \sigma_y \varepsilon_y \tag{3.21}$$

$$\beta_{s,i} = \left(\frac{E_i}{E_{hys,s} - \sum_{j=1}^{i} E_j} \right)^c \tag{3.22}$$

Notice that based on the requirement $0 \leq \beta_{s,i} \leq 1$, the values of the yield stress and hardening slope will never degrade below zero. As a result of the degradation of the yield stress and hardening modulus, the peak stress (2 to 2') and target stress (11 to 11') are each reduced and the loading path cycles more rapidly toward the residual branch.

3.3.2 Post-Capping Strength Degradation Mode

The post-capping degradation mode accounts for the reduction in strength as reloading occurs toward the post-capping (softening) branch, physically representative of mode 2 in Figure 3.15. This degradation mode shifts the softening branch toward the origin by shifting the peak strength point (2 to 2'). This is done by degrading a reference stress value that represents the peak stress value extrapolated back to the vertical axis (i.e. point of zero strain) by the softening modulus. Updates are made when the horizontal axis is crossed for the direction of loading only. Figure 3.17 shows this strength degradation mode for the Mercury implemented pinching model.

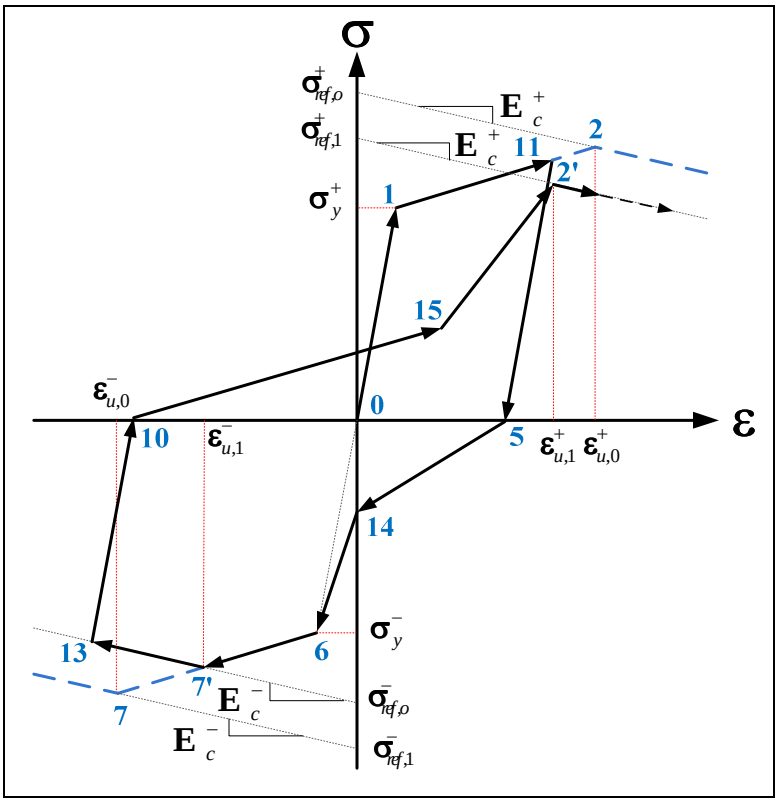

Figure 3.17 Post-Capping Strength Degradation Mode

The degraded reference stress is determined using the dissipation parameter $\beta_{c,i}$ corresponding to the capping strength degradation mode.

$$E_{hys,c} = \gamma_c \sigma_y \varepsilon_y \tag{3.23}$$

$$\beta_{c,i} = \left(\frac{E_i}{E_{hys,c} - \sum_{j=1}^{i} E_j} \right)^c \tag{3.24}$$

$$\sigma^{+/-}_{ref,i} = (1 - \beta_{c,i}) \sigma^{+/-}_{ref,i-1} \tag{3.25}$$

where

γ_c = hysteretic energy factor associated with capping strength degradation

$\beta_{c,i}$ = capping strength dissipation parameter for time step “i"

$\sigma_{ref,i-1}^{+/-}$ = directional reference stress for time step “i-1”

$\sigma_{ref,i}^{+/-}$ = directional reference stress for time step “i”

It is important to point out that the softening modulus is held constant and the peak stress is degraded only by altering the reference stress. Once the reference stress has been degraded, updates must be made to the peak stress, the target stress-strain state, and the residual strain. Over time, the reduced values will cause subsequent loading to target the residual branch.

3.3.3 Unloading Stiffness Degradation Mode

The unloading stiffness degradation mode accounts for degradation mode 3 in Figure 3.15. It is unique in comparison to the other degradation modes. First, it does not update independently for the positive and negative stress branches. Instead, updates are made for both directions, meaning that unloading stiffness degradation may occur more rapidly than the other modes. Also, degradation does not occur when loading crosses the horizontal axis. Updates are made to the unloading stiffness when a strain reversal occurs, at the time step when unloading first occurs (Figure 3.18). This mode degrades the unloading modulus by using the associated dissipation parameter $\beta_{k,i}$.

$$E_{hys,k} = \gamma_k \sigma_y \varepsilon_y \tag{3.26}$$

$$\beta_{k,i} = \left(\frac{E_i}{E_{hys,k} - \sum_{j=1}^{i} E_j} \right)^c \tag{3.27}$$

$$E_{u,i} = (1 - \beta_{k,i}) E_{u,i-1} \tag{3.28}$$

where

γ_k = hysteretic energy factor associated with unloading stiffness degradation

$\beta_{k,i}$ = unloading stiffness dissipation parameter for time step "i"

$E_{u,i-1}$ = unloading modulus for time step "i-1"

$E_{u,i}$ = unloading modulus for time step "i"

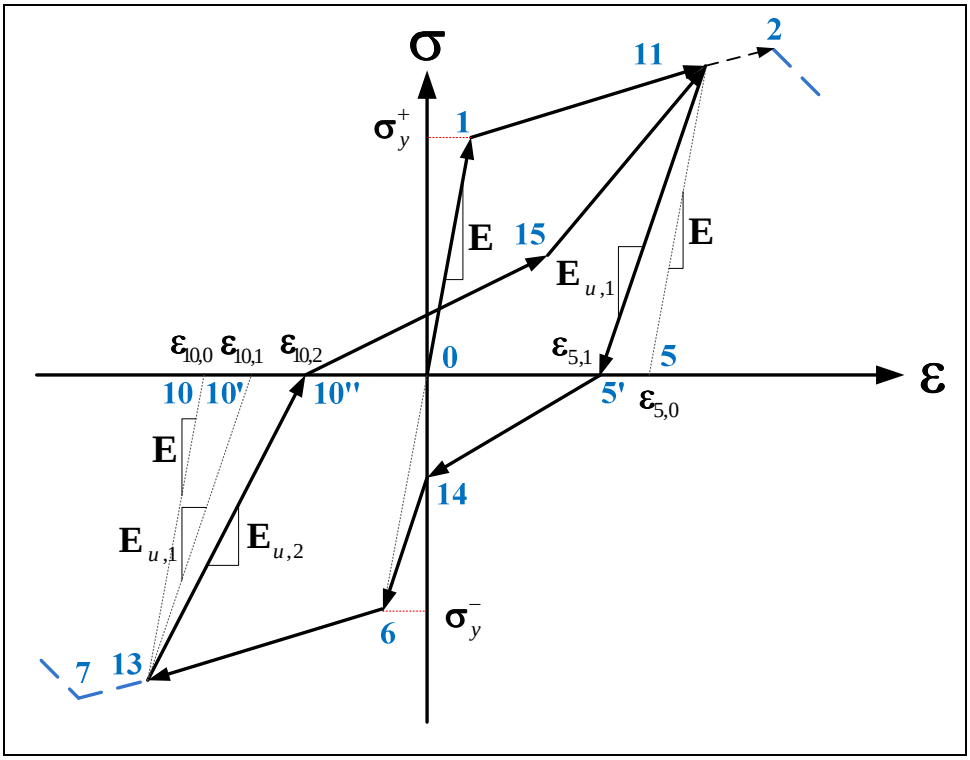

Figure 3.18 Unloading Stiffness Degradation Mode

Over one complete cycle, the unloading modulus is degraded twice from a value equal to the initial elasticity. By doing so, the strain values at point 5 and point 10 are shifted toward the origin (5-5' and 10-10'-10"), resulting in accelerated unloading. Because this mode may update twice as much as other modes, an appropriate γ value associated with this mode should be up to twice that of the other modes if comparable energy dissipation is expected in both directions (Ibarra, 2003). If an adequate γ value is

not chosen, exhaustion of the hysteretic capacity will occur prematurely and the value of $\beta_{k,i}$ will approach infinity and fall below zero well before exhaustion of hysteretic capacity takes place for the other modes.

Another unique characteristic of the unloading stiffness degradation mode is that updates are only made if unloading has occurred completely on one branch (positive or negative) and reloading has commenced on the opposite branch (Figure 3.19). When unloading is not completed and reloading begins on the same side, it is considered an interruption and the updated value ($E_{u,2a}$ in Figure 3.19) is disregarded. When another strain reversal occurs, the disregarded degraded value is replaced by a new value ($E_{u,2b}$ in Figure 3.19) based on the new dissipated energy.

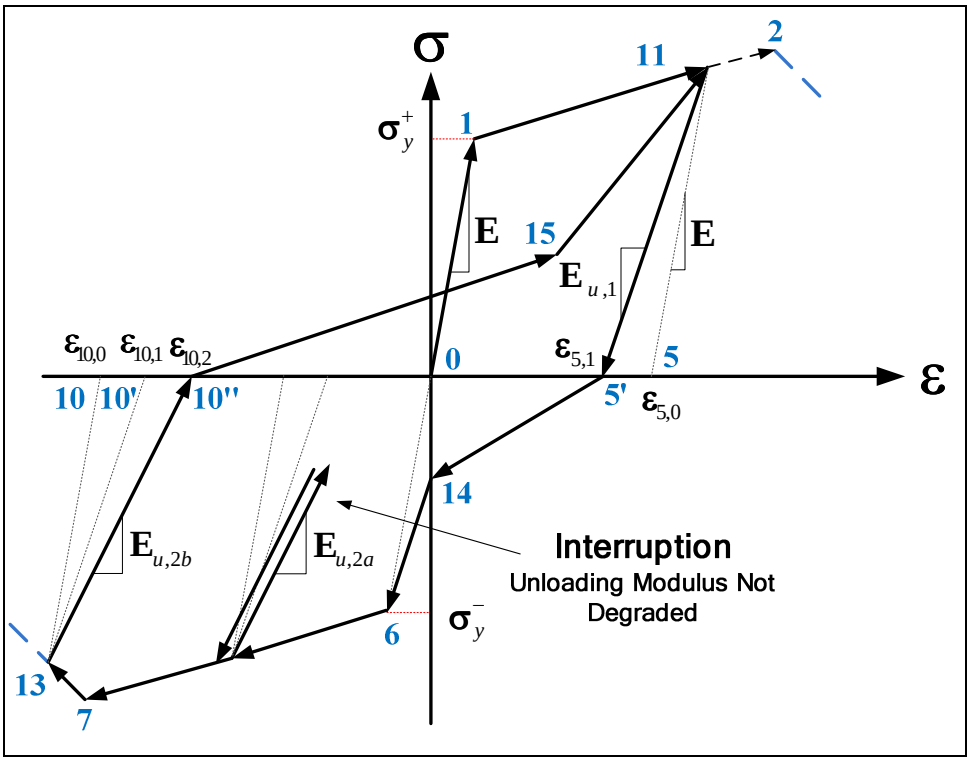

Figure 3.19 Unloading Interruption

Once complete unloading has taken place, the current degraded value is stored and degraded upon the next strain reversal. As the unloading stiffness is degraded, it is apparent that increasing hysteretic energy dissipation will occur, resulting in rapid degradation of the other modes.

3.3.4 Accelerated Reloading Stiffness Degradation Mode

The accelerated reloading stiffness degradation mode accounts for mode 4 in Figure 3.15. The accelerated reloading mode increases the absolute value of the directional target strain value. By doing so, the target points (11-11') and break-points (15-15' and 14-14') shift away from the origin, causing the reloading modulus to decrease. Updates are made when the horizontal axis is crossed for the direction of loading only.

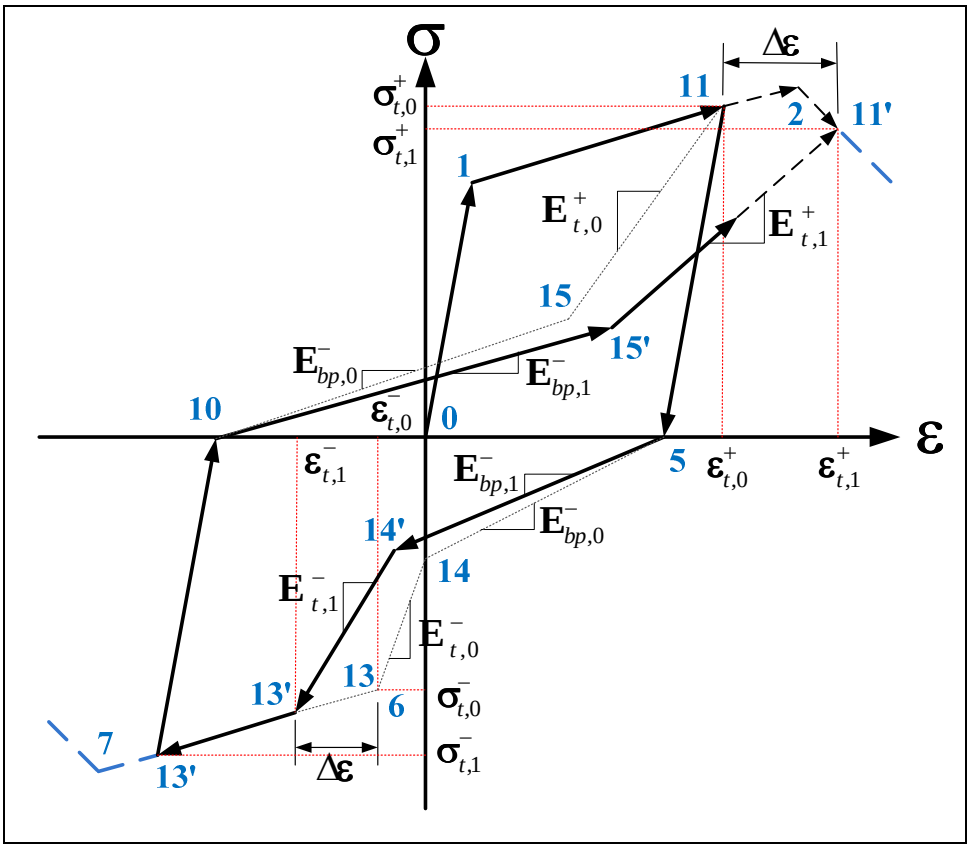

Figure 3.20 Accelerated Reloading Stiffness Degradation Mode

The target strain is shifted by adjusting the value using the accelerated reloading dissipation parameter $\beta_{a,i}$.

$$E_{hys,a} = \gamma_a \sigma_y \varepsilon_y \tag{3.29}$$

$$\beta_{a,i} = \left(\frac{E_i}{E_{hys,a} - \sum_{j=1}^{i} E_j} \right)^c \tag{3.30}$$

$$\varepsilon_{t,i}^{+/-} = (1 + \beta_{a,i}) \varepsilon_{t,i-1}^{+/-} \tag{3.31}$$

where

γ_a = hysteretic energy factor associated with accelerated reloading stiffness degradation

$\beta_{a,i}$ = accelerated reloading dissipation parameter for time step “i"

$\varepsilon_{t,i-1}^{+/-}$ = directional target strain for time step “i-1”

$\varepsilon_{t,i}^{+/-}$ = directional target strain for time step “i”

As a result of the target strain shift, reloading is accelerated and the loading path targets the residual branch much more rapidly as hysteretic energy dissipation increases.

3.3.5 Summary

The peak-oriented and pinching hysteretic constitutive models discussed in Chapters 3.2.1 and 3.2.2, respectively, have each been chosen for implementation into Mercury. Both models follow the same monotonic backbone that includes an elastic loading branch, a strain hardening branch, a post-capping branch, and a residual strength branch. Each model adheres to peak-oriented reloading toward the point of maximum deformation experienced in previous loading cycles, the only difference being that

the pinching model includes a second reloading branch that targets the pinching break-point before targeting the maximum deformation state. To account for hysteretic strength loss, the strength and stiffness degradation modes presented in sections 3.3.1-3.3.4 have been chosen to be included in both the implemented peak-oriented and pinching models.

Chapter 3.4 describes the implementation of both models including general information about the process of implementing a new constitutive model in Mercury.

3.4 Mercury Implementation

Description of the Mercury implementation begins with a discussion about the process of implementing a new constitutive model in Mercury. Chapter 3.4.1 is intended to provide a reader with all the information necessary to develop a constitutive model externally of the Mercury source code. Details about the implemented peak-oriented and pinching models follow, providing input variable descriptions and coded warnings to signal exhaustion of hysteretic energy. A validation section is included with a comparison to the OpenSees (OpenSees, 2005) implementation and a parametric study.

3.4.1 Adding a New Constitutive Model to Object Oriented Mercury

Coding of the hysteresis models was completed taking advantage of MATLAB's object oriented programming capabilities. Mercury defines materials as an object, requiring minimal modification to the Mercury process in order to implement a new constitutive model. MATLAB uses the *handle* class structure to reference a particular object and the *classdef* structure to define a subclass to handle specific object events. The constitutive model operates by defining a material subclass definition (*classdef*) associated with a particular element or element fiber. When an element is defined in a Mercury input file, the associated constitutive model tag relates to this material subclass definition to

handle material object events. The peak-oriented model and pinching model have been assigned the subclass identifications *Hysteresis* and *HysteresisPinch* respectively.

When defining a new *classdef* in MATLAB, the subclass name is included in the first line (Figure 3.21). Within the subclass definition, a *properties* section is used to define all subclass update properties, and a *methods* section is used to obtain user input variables and handle object events.

```
1   classdef Hysteresis < handle
2
3       properties
4
5           % update properties defintion
6
7       end
8
9       methods
10
11          % object event handling methods defintion
12
13      end
14
15  end
```

Figure 3.21 MATLAB Object Subclass Definition

The *properties* section stores uniaxial stress-strain variables, updates variables at each time step, and loads variables committed from the previous time step (Figure 3.22). Mercury developers use the *properties* section to handle update variables as variable modifications occur from the event handling of the *methods* section.

```
    properties (SetAccess=protected,Hidden)
        % Input variables
        input_var_1;
            ...
        input_var_i;
    end

    properties (SetAccess=public,GetAccess=public)
        % Storage for output
        uni_strain_stress    % Storage of uniaxial strain and stress
    end

    properties (SetAccess=public,GetAccess=public,Hidden)
        % update variables
        update_var_1;
            ...
        update_var_i;
        % commit variables
        com_var_1;
            ...
        com_var_i;
    end
```

Figure 3.22 Mercury Properties Definition

The *methods* section (Figure 3.23) makes use of a series of functions to determine state variables at each increment. In the *methods* section, Mercury developers define the primary model code to handle events (state determination). A description of each of the functions in the methods section is provided in Table 3-2.

No.	Function Name	Description
1	function mat = Hysteresis	Extracts user input variables and prompts initialization
2	function Initialization	Initializes values for all update variables
3	function commit	Stores variables from previous time step to be used in current step
4	function revert	Stores variables from previous time step to return to if convergence issues are met
5	function materialStateDetermination	Main developer function - describes event handling to determine state variables at current increment

Table 3-2 Methods Functions

```
    methods (Access=public)
        function mat = Hysteresis(info) % obtain input variables
            mat.input_var_1     = info{1};
                ...
            mat.input_var_i     = info{i};
            mat.Initialization(); % prompt initialization
        end

        function Initialization(mat)
            mat.update_var_1    = ###; % user defined
                ...
            mat.update_var_i    = ###; % user defined
        end % function Initialization

        function commit(mat) % store values at previous step for current step
            mat.com_update_var_1    = mat.update_var_1;
                ...
            mat.com_update_var_i    = mat.update_var_i;
        end

        function revert(mat) % store values at previous step in case of convergence issues
            mat.update_var_1    = mat.com_update_var_1;
                ...
            mat.update_var_i    = mat.com_update_var_i;
        end

        function materialStateDetermination(mat, TrialStrain)

            #### % user defined event handling

        end
```

Figure 3.23 Mercury Methods State Variable Determination

3.4.2 Hysteresis and HysteresisPinch User Input Variables

This section details user input for the Mercury implementation of the peak-oriented model presented in Chapter 3.2.1 and the pinching model presented in Chapter 3.2.2, both including the strength and stiffness degradation modes discussed in Chapter 3.3 (Ibarra, 2003). For the implemented models, user input variables are defined such that the user specifies the stress-strain states at the yield point, peak capacity point, and residual point, on both the positive and negative sides of the model (points 1,2,3,6,7,and 8 in Figure 3.6 and Figure 3.9). The user also defines the hysteretic energy capacity factor (γ) and deterioration rate value (c) (Chapter 3.3). For the HysteresisPinch (pinching) model, the reloading stress and strain (κ_f and κ_d in Chapter 3.2.2) factors are also user inputs. A summary of the

user input variables for both the Hysteresis (peak-oriented) and HysteresisPinch (pinching) models is provided in Table 3-3.

Variable	Description	Model	
		Peak-Oriented	Pinching
$\sigma_y^{+/-}$	Positive & Negative Yield Points	✓	✓
$\varepsilon_y^{+/-}$		✓	✓
$\sigma_u^{+/-}$	Positive & Negative Peak Points	✓	✓
$\varepsilon_u^{+/-}$		✓	✓
$\sigma_r^{+/-}$	Positive & Negative Residual Points	✓	✓
$\varepsilon_r^{+/-}$		✓	✓
γ	Hysteretic Capacity Factor	✓	✓
c	Deterioration Rate Factor	✓	✓
κ_f	Reloading Break-Point Stress Factor	☐	✓
κ_d	Reloading Break-Point Strain Factor	☐	✓

Table 3-3 Hysteresis and HysteresisPinch Input Variables

From these inputs, the initial elastic modulus, hardening modulus, and softening modulus are determined and variables are initialized for degradation. For the purpose of this implementation, it is assumed that γ is equal for each of the modes except the unloading stiffness mode for which γ is twice that of the other modes, as suggested by Ibarra (Ibarra, 2003).

$$\gamma_s = \gamma_c = \gamma_a \tag{3.32}$$

$$\gamma_k = 2\gamma_{s,c,a} \tag{3.33}$$

A single value, therefore, is input by the user for the hysteretic capacity factor (γ).

3.4.3 Strength Degradation in Mercury

A complete description of the strength and stiffness degradation that occurs, when the horizontal axis of the stress-strain plot is crossed, is provided in Figure 3.24 for updates to variables on the positive loading branch. When the horizontal axis is crossed (point 10), and the stress state changes sign, the yield stress ($\sigma_{y,o}^{+}$), hardening modulus ($E_{s,o}^{+}$), reference stress ($\sigma_{ref,o}^{+}$), and target strain ($\varepsilon_{t,o}^{+}$) are each updated for the direction of loading. As a result, the yield strain ($\varepsilon_{y,o}^{+}$), stress-strain values at the peak point ($\sigma_{u,o}^{+}$ and $\varepsilon_{u,o}^{+}$), residual strain ($\varepsilon_{r,o}^{+}$), target stress ($\sigma_{t,o}^{+}$), and target reloading modulus (E_{t}^{+}) are each degraded. For the HysteresisPinch model, the stress-strain values at the break-point (σ-ε at point 15), and the break-point reloading modulus (E_{bp}^{+}) are degraded as well. Similar strength and stiffness degradation occurs for variables on the negative stress branch when unloading of the positive branch (points 11-5 in Figure 3.24) and the horizontal axis is crossed at point 5, leading to loading of the negative stress branch.

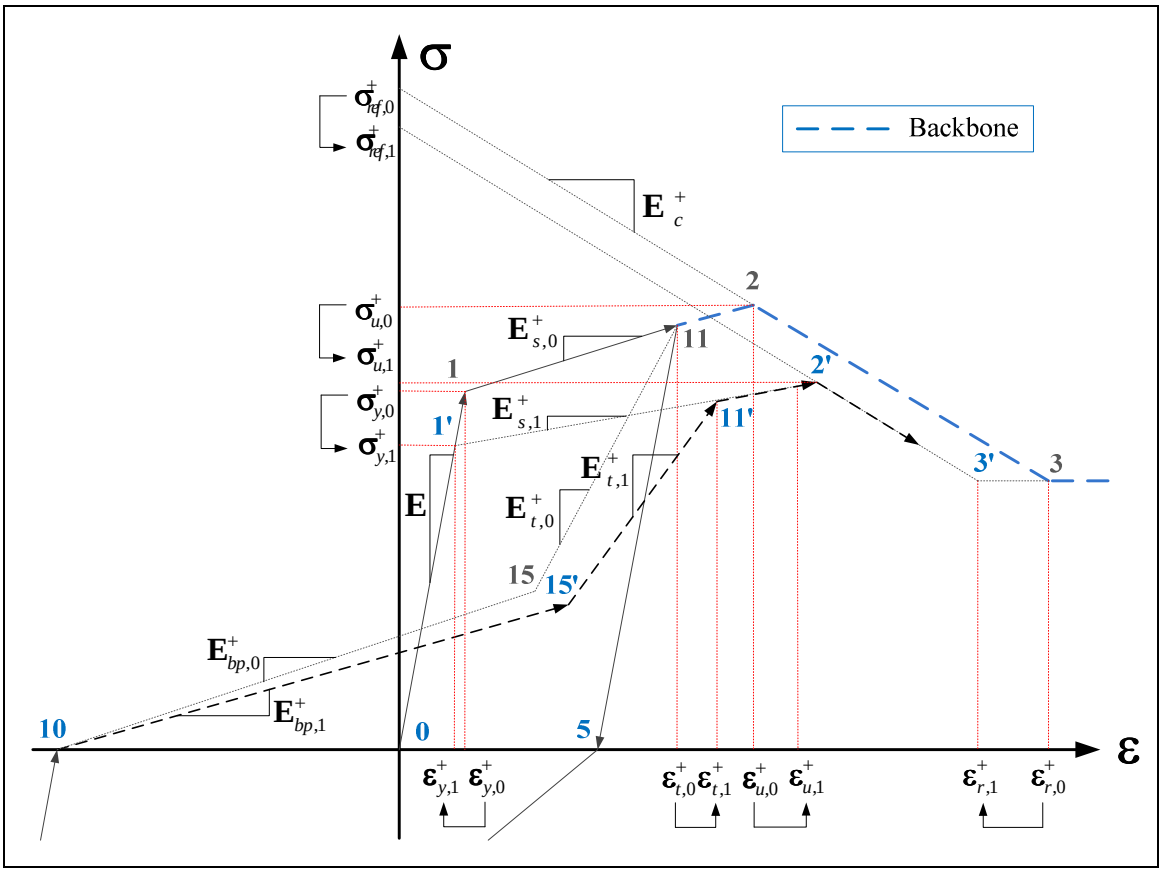

Figure 3.24 Mercury Variable Degradation

Figure 3.24 illustrates the strength degradation that occurs when the horizontal axis is crossed in the implemented peak-oriented (*Hysteresis*) and pinching (*HysteresisPinch*) models, accounting for basic strength degradation (Chapter 3.3.1), post-capping strength degradation (Chapter 3.3.3), and accelerated reloading stiffness degradation (Chapter 3.3.4). Unloading stiffness degradation, however, does not occur when the horizontal axis is crossed. Instead, the unloading modulus is updated when a strain reversal occurs (Figure 3.25 points 11 and 13) and updates are not made separately for the positive and negative branches (see Chapter 3.3.3). In other words, the same unloading modulus is used and degraded for unloading in both directions. For degradation of the unloading modulus, the implemented model updates a flag when unloading begins. The flag signals Mercury to prevent further

degradation of the unloading modulus until unloading has completed and reloading has begun in the opposite direction (i.e. unloading from 11-5 or 13-10).

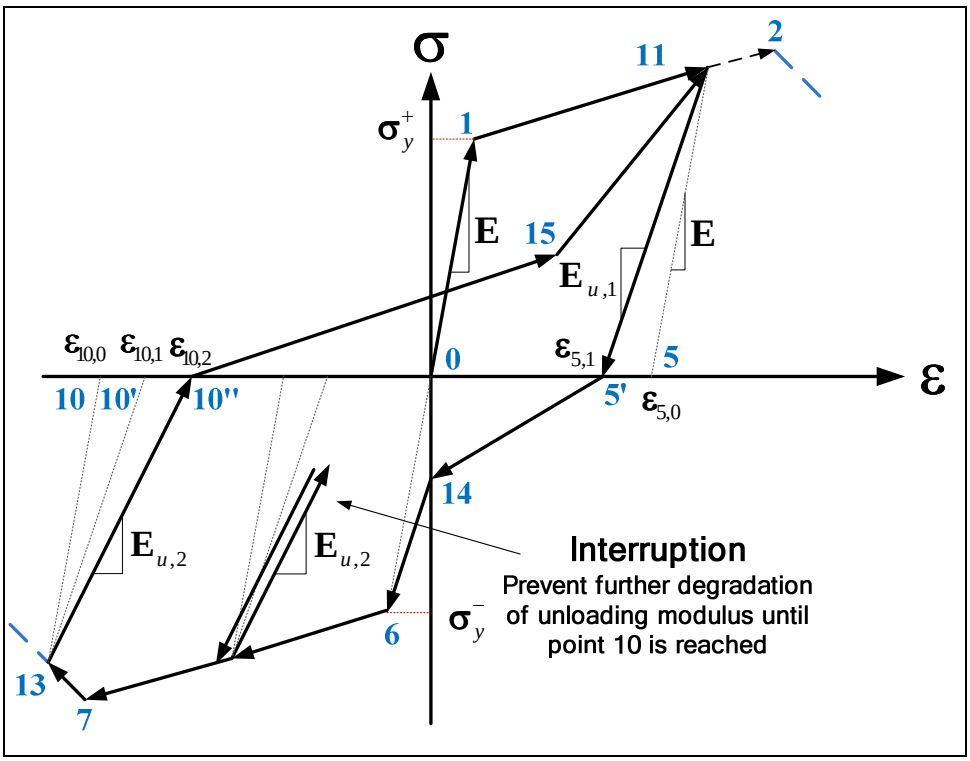

Figure 3.25 Implemented Unloading Degradation Rules

Over the duration of several loading cycles, as the dissipated energy approaches or exceeds the hysteric capacity, the value of β, for the degradation, tends toward infinity or becomes negative. In this case, a value of 1 is assigned for β (in the implemented Mercury model) and a message is displayed to inform the user that hysteretic capacity has been exhausted (Figure 3.26). Structural failure is expected during this analysis time step and post-process results should indicate rapid loading toward the residual strength.

```
     745 step:   0.012515 second
     746 step:   0.010819 second
     747 step:   0.010749 second
     748 step:   0.010120 second
     749 step:   0.012589 second
**************************
Hysteretic Energy Exhausted
**************************
**************************
Hysteretic Energy Exhausted
**************************
     750 step:   0.014298 second
     751 step:   0.011989 second
     752 step:   0.011900 second
     753 step:   0.013559 second
```

Figure 3.26 Hysteretic Energy Exhaustion in Mercury

3.4.4 Validation

For the purpose of validation, a two-dimensional truss member is subjected to cyclic axial displacements. First, the hysteretic dissipation capacity factor (γ) is set to infinity to verify that the model follows the monotonic backbone. Recall that strength and stiffness degradation uses the dissipation parameter β which is a function of the hysteretic capacity (E_{hys}) related to β by Equation(3.17).

$$\beta_i = \left(\frac{E_i}{E_{hys} - \sum_{j=1}^{i} E_j} \right)^c$$

Recall also, from Equation(3.18), that the hysteretic capacity is the energy necessary to reach yielding, factored by γ.

$$E_{hys} = \gamma \sigma_y \varepsilon_y$$

Thus, by setting γ to infinity, no strength and stiffness degradation is included and the validation model should reload to the monotonic backbone.

After validation of the monotonic backbone, the implemented peak-oriented model is compared to a similar model implemented in OpenSees (OpenSees, 2005) to ensure that a similar strength degradation pattern occurs when γ is set to a finite value.

3.4.4.1 Monotonic Backbone Validation

The truss member constitutive model properties are summarized in Figure 3.27 for the monotonic backbone validation model. Note that the values κ_f and κ_d apply to the pinching (*HysteresisPinch*) model only.

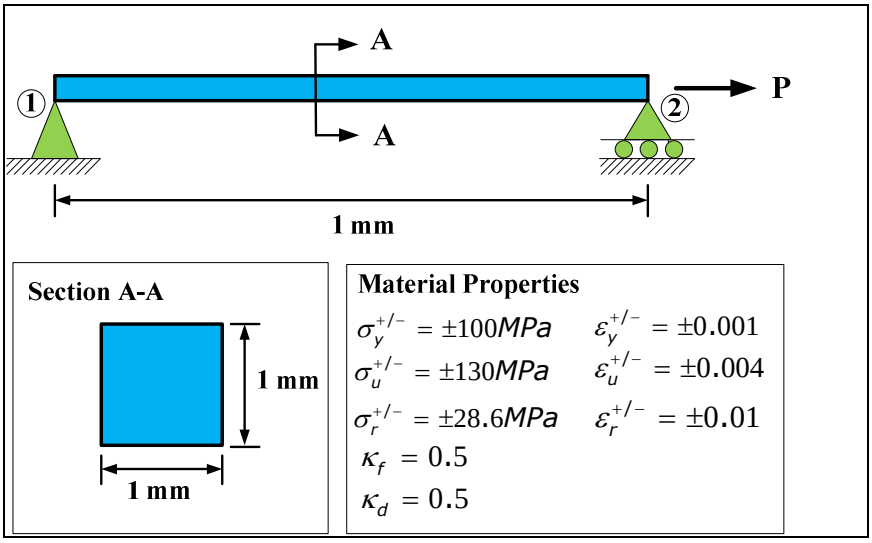

Figure 3.27 Two-Dimensional Truss Member Properties

The truss member is subjected to the axial displacements shown in Figure 3.28, up to a maximum displacement of 12 times that of the member yield displacement. This value was chosen for this particular member because the assigned residual displacement is 10 times that of the yield (Figure 3.27), thus loading should occur to the residual branch.

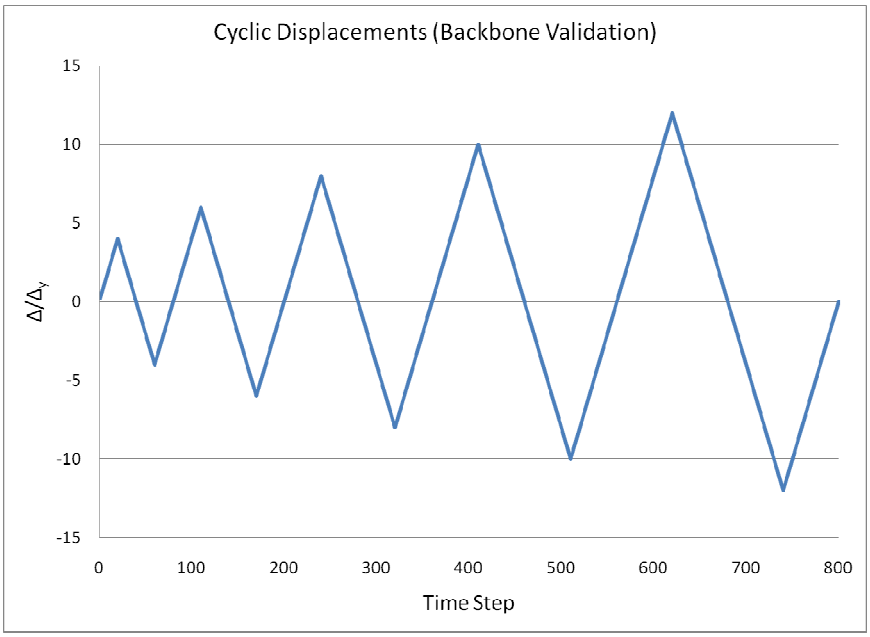

Figure 3.28 Backbone Validation Prescribed Displacements

In order to verify that each of the models follows the monotonic backbone, the value of the hysteretic capacity factor ( ) is set to infinity, eliminating strength degradation. Both models adequately capture the cyclic loading history, eventually loading along the residual branch (Figure 3.29 and Figure 3.30).

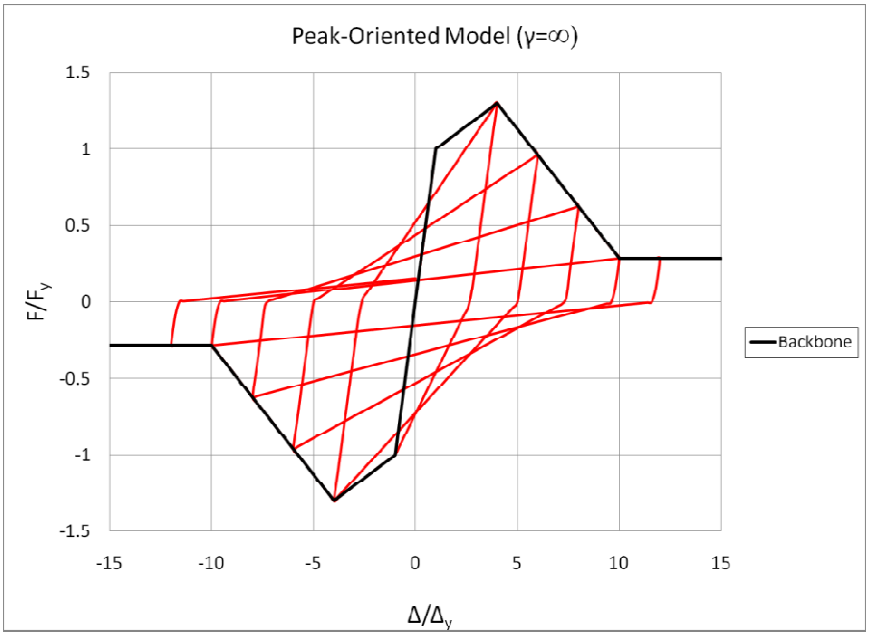

Figure 3.29 Peak-Oriented Model, No Strength Degradation

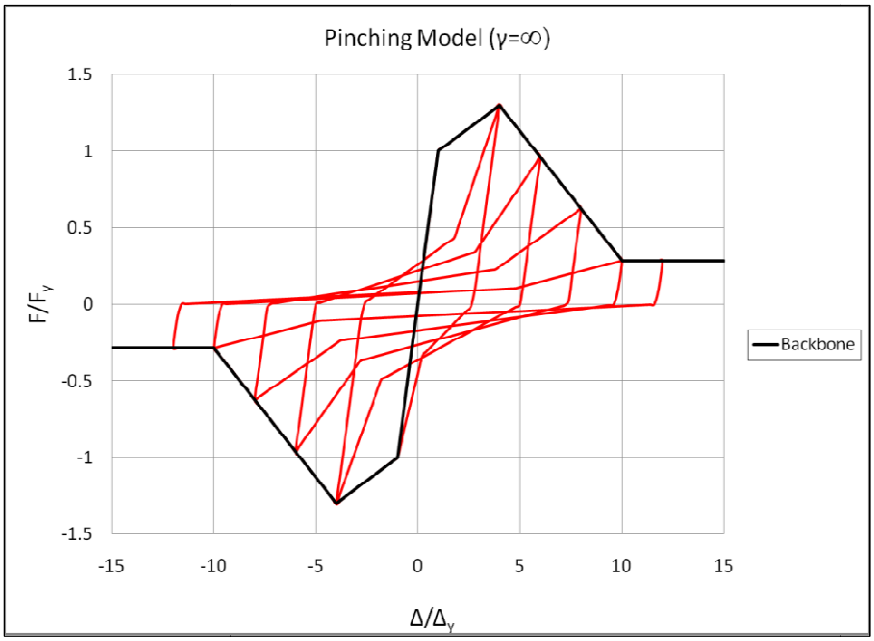

Figure 3.30 Pinching Model, No Strength Degradation

3.4.4.2 Comparison to OpenSees Implementation

A similar model is used for the OpenSees comparison; however, a finite value is assigned for the hysteretic capacity factor (γ) to incorporate the strength and stiffness degradation modes. The material and hysteresis properties are summarized in Figure 3.31. The properties do not match the validation model in Section 3.4.4.1. An attempt was made to use the same properties in OpenSees; however, the OpenSees analysis failed to converge. An example models was available with the material properties summarized in Figure 3.31.

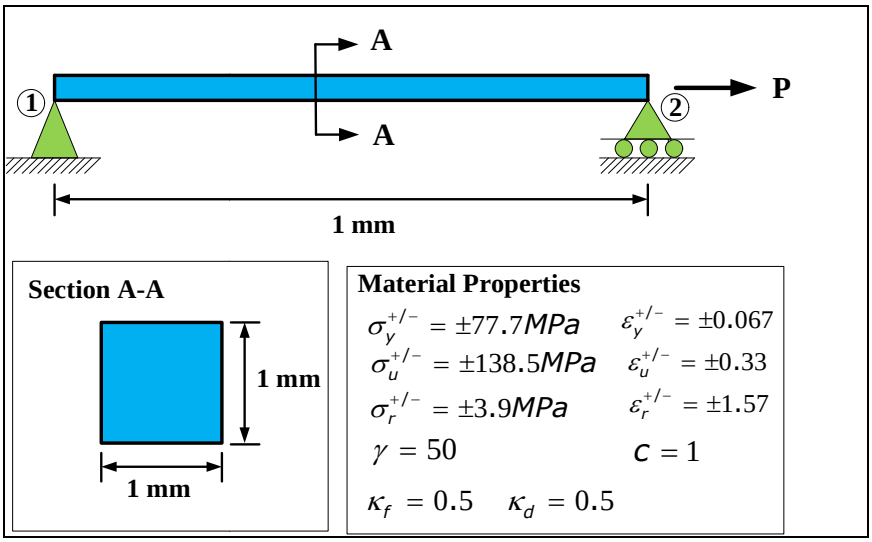

Figure 3.31 OpenSees Comparison Model

The member is subjected to axial displacements up to 8 times the yield, following the loading pattern in Figure 3.32.

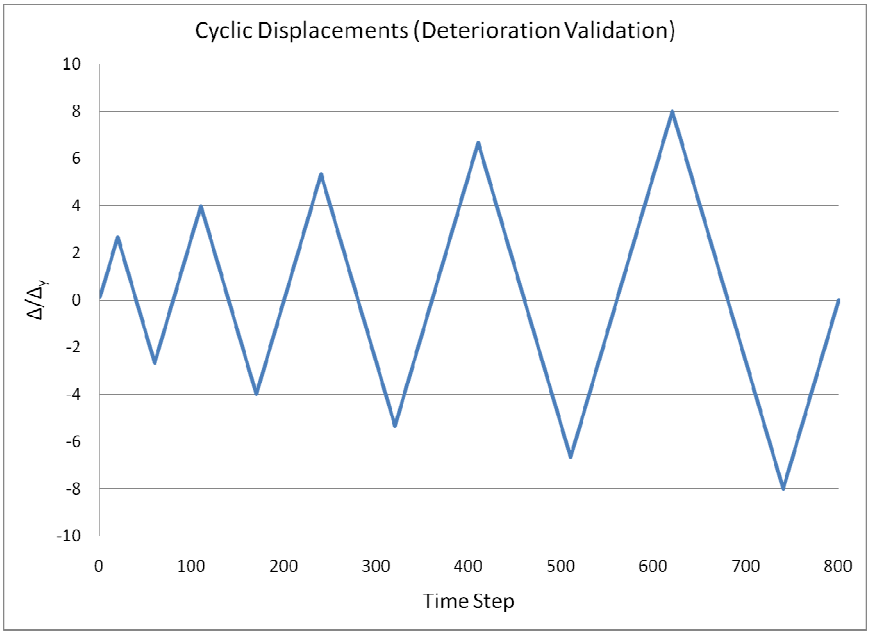

Figure 3.32 Prescribed Displacements

The peak-oriented models are essentially identical (Figure 3.33), undergoing the same strength degradation pattern over the duration of the loading history. Because of this, it is impossible to distinguish between the two models in the comparison plot.

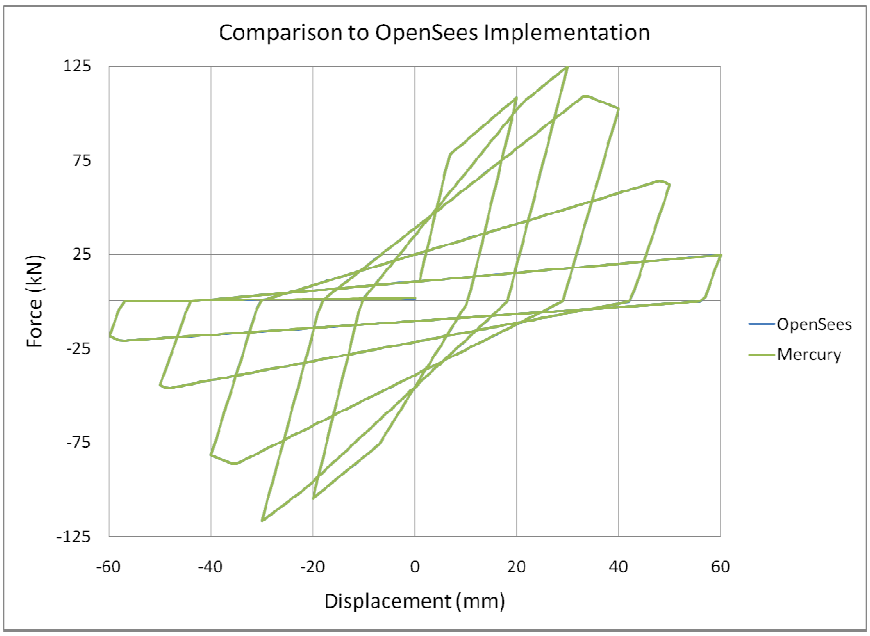

Figure 3.33 Peak-Oriented Model Comparison to OpenSees Clough Material

Comparison of the pinching model implementations (Figure 3.34), on the other hand, does not produce such a close match. The OpenSees model (*Pinching uniaxial material*) does not provide a user input variable for the break-point strain factor (κ_d). Instead, only the stress factor (κ_d) is taken as user input. The reloading path beyond the break-point restores the initial elastic modulus rather than following a degraded reloading modulus as suggested by the Ibarra pinching model (Section 3.2.2). The OpenSees implementation, thus, does not follow the reloading rules suggested by Ibarra. As a result, the OpenSees and Mercury implementations determine the break-point strain different from one another.

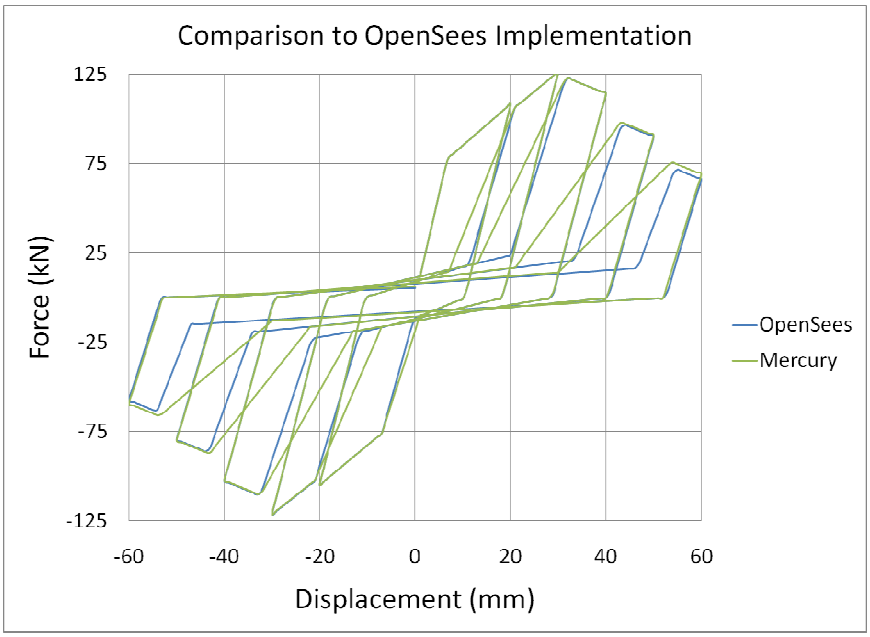

Figure 3.34 Pinching Model Comparison to OpenSees Pinching Material

3.4.5 Parametric Study

In this section, the hysteretic capacity factor (γ) and the deterioration rate (c) are examined to illustrate the effects of varying these degradation mode variables.

3.4.5.1 Hysteretic Capacity Factor (γ)

Strength degradation rates are examined by varying the values of the hysteretic energy capacity factor ( ). By doing so, the rate at which strength and stiffness degradation occurs is compared. Values of 50 and 100 are assigned for γ and the member is subjected to the cyclic displacements shown in Figure 3.32, up to a maximum of 8 times the yield displacement. The deterioration rate factor (c) is held constant at 1. The model and material properties used are the same as those in Figure 3.27. In both cases, lower values of result in more rapid exhaustion of hysteretic capacity and quicker strength degradation (Figure 3.35 and Figure 3.36). The monotonic backbone is shown to illustrate the magnitude of strength loss.

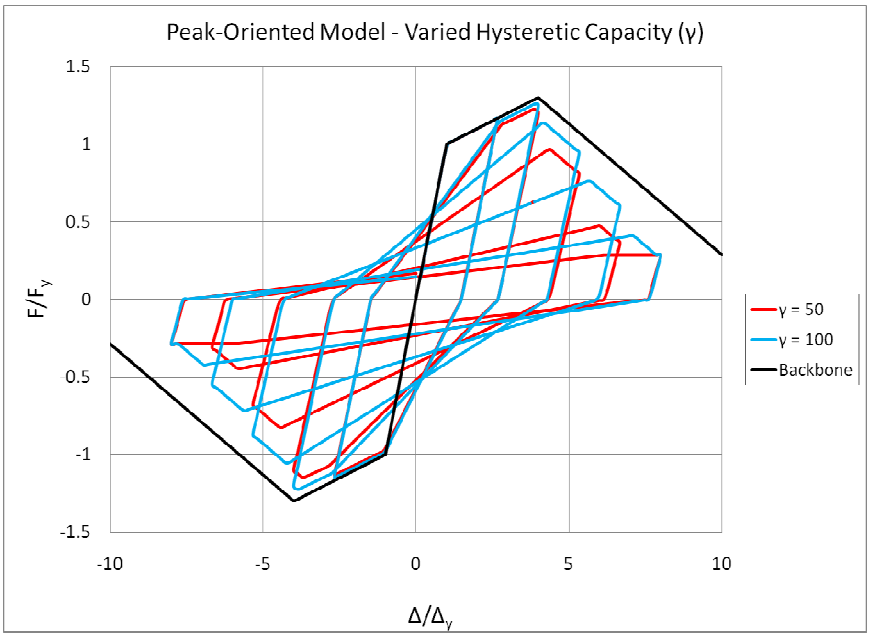

Figure 3.35 Varying Hysteretic Capacity (Peak-Oriented)

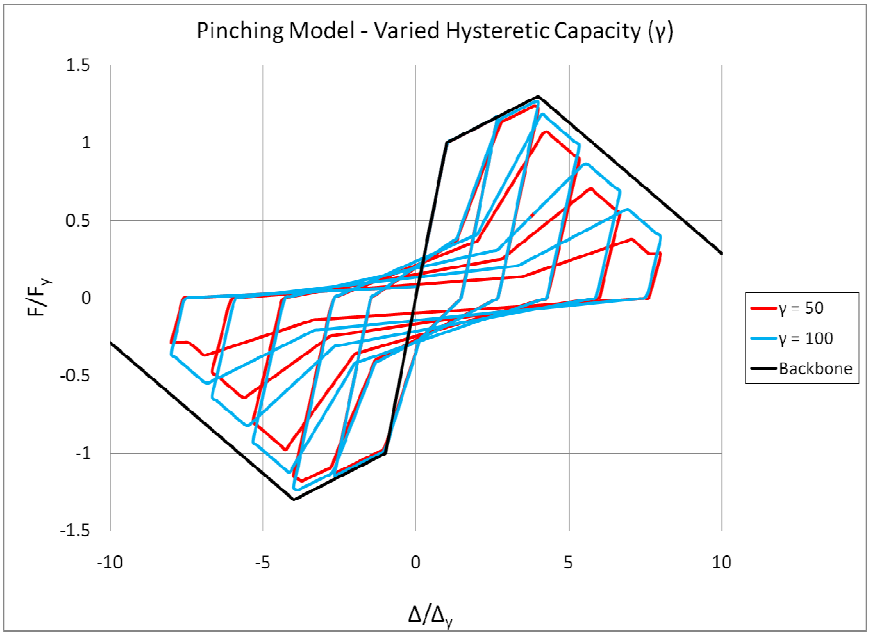

Figure 3.36 Varying Hysteretic Capacity (Pinching)

3.4.5.2 Deterioration Rate Factor (c)

Similarly, values of the rate deterioration factor (c) are varied for the model shown in Figure 3.27.

Values of 1 and 2, the bounds suggested by Rahnama and Krawinkler (Rahnama & Krawinkler, 1993), are

examined, and the value of is held constant at 50. As expected, lower values of result in more rapid strength degradation (Figure 3.37 and Figure 3.38). Again, the monotonic backbone is included to illustrate the magnitude of strength loss.

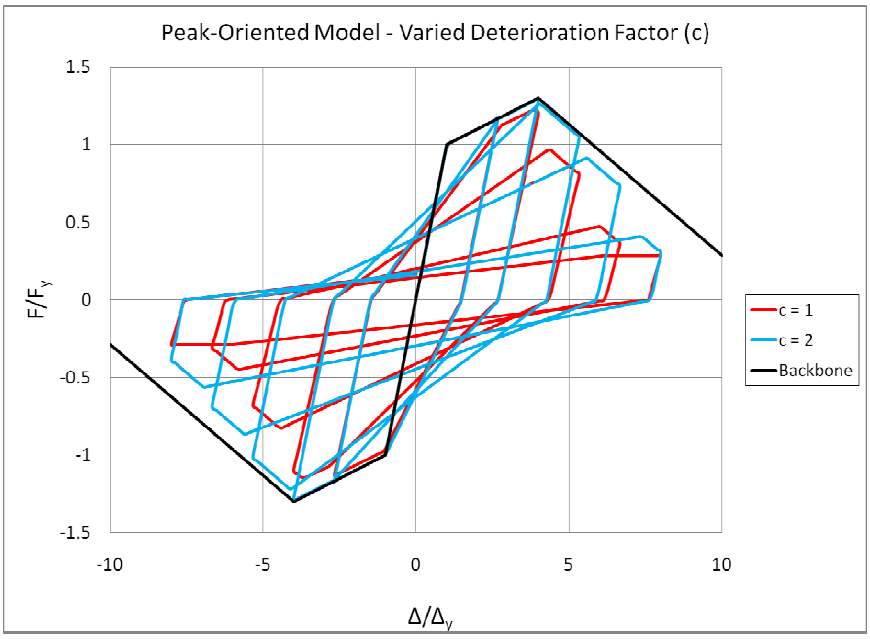

Figure 3.37 Varying Deterioration Rate (Peak-Oriented)

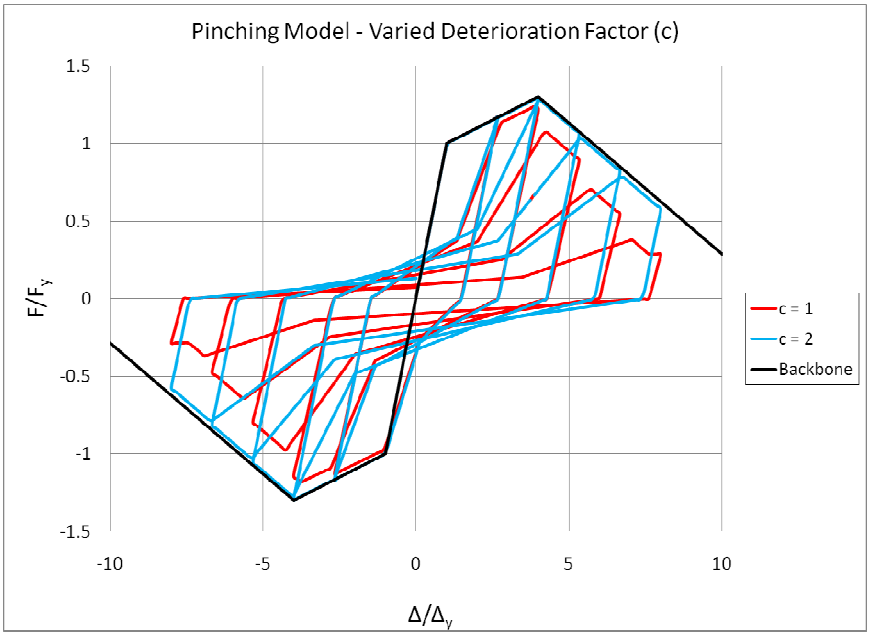

Figure 3.38 Varying Deterioration Rate (Pinching)

3.5 Lumped Plasticity Modeling in Mercury

This section is included as a description for assembling and calibrating a lumped plasticity model, in Mercury, using the implemented peak-oriented (*Hysteresis*) model. It is not included as further validation; instead, it is a starting point for calibrating constitutive properties to model flexural behavior of a beam-column element.

3.5.1 Modeling Description

A simple lumped plasticity model was introduced in Chapter 2 for a cantilever column. A similar model is used to explain lumped plasticity modeling using the implemented hysteretic models discussed in Chapter 3.4. The cantilever column in Figure 3.39 is modeled as a lumped plasticity element in Mercury (see Figure 3.42). The model is subjected to reverse cyclic displacements applied at the top of the column (Figure 3.40).

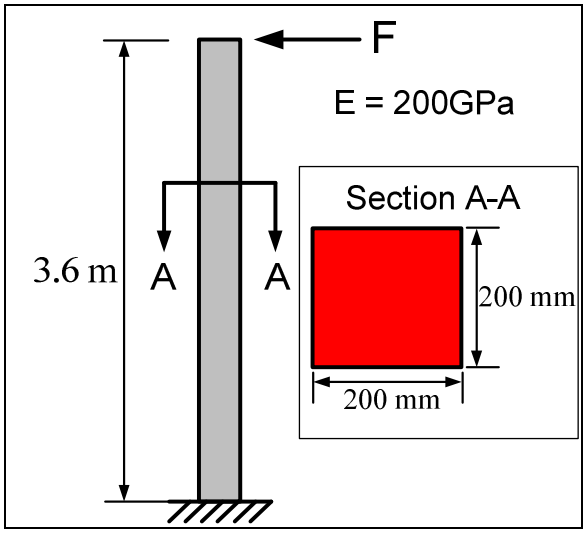

Figure 3.39 Prototype Column

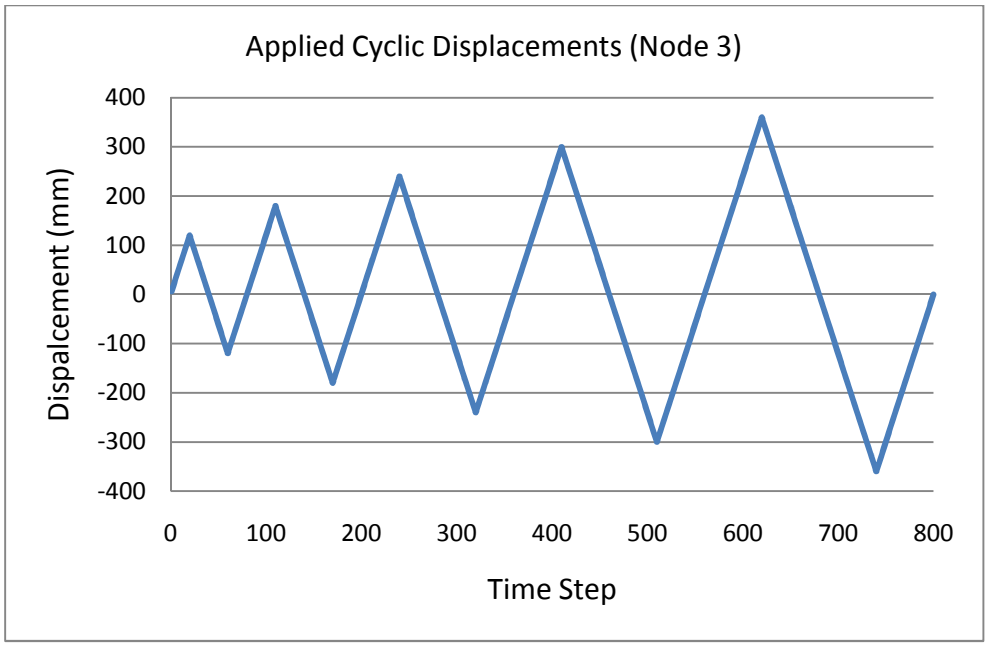

Figure 3.40 Applied Displacements

Figure 3.42 shows a Mercury lumped plasticity model for the cantilever column. The model consists of a linear elastic beam-column element and a zero-length element. For simplicity, a second zero-length element is not included in the model at node 3. In Mercury, zero-length elements are a series of three uniaxial springs: an axial spring, a transverse shear spring, and a rotational spring. The *Hysteresis* (peak-oriented) constitutive model is to be assigned to the zero-length rotational spring with the goal of achieving the combined moment-rotation column response shown in Figure 3.41.

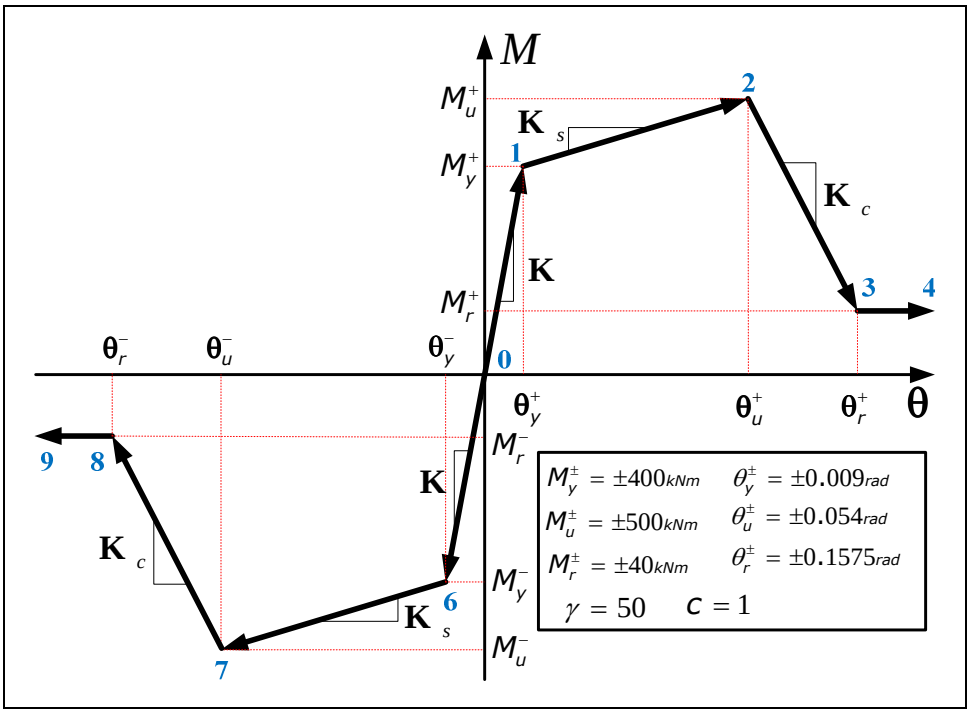

Figure 3.41 Combined Response

When shear and axial deformations are to be neglected in the zero-length element, the stiffness of both the axial and shear spring is set to a high value. To do so, a linear elastic constitutive model with a high stiffness value is assigned. Careful selection of the stiffness values for the rigid springs must be made. A low value will result in the addition of unwanted axial and shear deformations. On the other hand, a very high value may result in convergence issues. Model calibration, therefore, should involve testing of the input values to determine acceptable stiffness values at which unwanted deformations do not occur and convergence is met (Saouma, 2011).

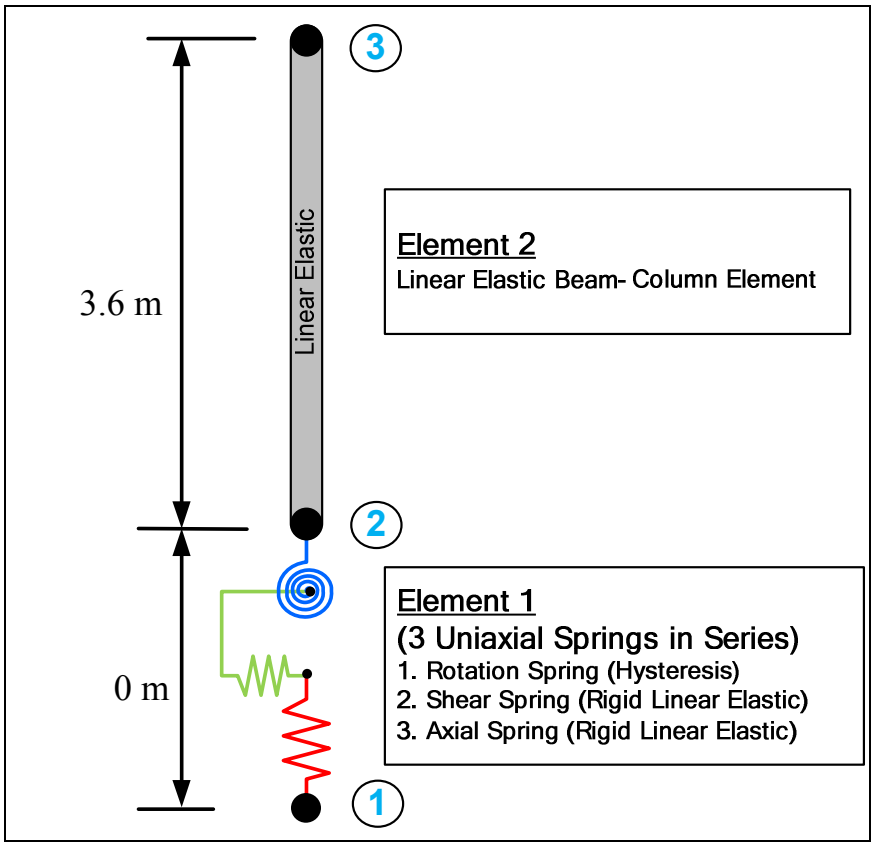

Figure 3.42 Hysteresis Lumped Plasticity Model

3.5.2 Model Calibration

The lumped plasticity model requires careful determination of the lumped plasticity element strength point values (i.e. θ_y, θ_u, θ_r) to accurately capture the desired response. Continuing with the discussion in Chapter 2, the combined stiffness of the linear elastic beam-column element and the lumped plasticity zero-length element should be equal to the stiffness of a distributed plasticity model. Using the Young's Modulus value of 200 GPa and assuming double curvature (Ibarra, 2003), the rotational stiffness of the element in Figure 3.39 is determined by (3.34). This value represents the desired calibration rotational stiffness.

$$K_{rotation} = \frac{6EI}{L} \tag{3.34}$$

$$K_{rotation} = 4.44x10^4 \frac{kNm}{rad} \tag{3.35}$$

Accounting for the stiffness contributions of the linear and lumped plasticity elements, the calibrated stiffness equality is shown in (3.38) by setting the combined stiffness equal to $K_{rotation}$.

$$\frac{1}{K_{combined}} = \frac{1}{K_{LP}} + \frac{1}{K_{linear}} \tag{3.36}$$

$$K_{combined} = K_{rotation} = 4.44x10^4 \frac{kNm}{rad} \tag{3.37}$$

$$K_{combined} = \frac{1}{\frac{1}{K_{LP}} + \frac{1}{K_{linear}}} = 4.44x10^4 \frac{kNm}{rad} \tag{3.38}$$

If the initial stiffness of the linear element is set equal to $K_{rotation}$, the initial stiffness of the lumped plasticity element must be set to infinity (a very high value) in order to satisfy (3.38).

$$K_{linear} = K_{rotation} = 4.44x10^4 \frac{kNm}{rad} \tag{3.39}$$

$$\frac{1}{K_{LP}} = \frac{1}{K_{combined}} - \frac{1}{K_{linear}} = \frac{1}{K_{rotation}} - \frac{1}{K_{rotation}} = 0 \tag{3.40}$$

$$K_{KP} = \infty \tag{3.41}$$

This is not a desired option as a very high stiffness will lead to convergence issues in Mercury. If, instead, the initial stiffness of the lumped plasticity element is set to a value 10 times that of the linear elastic element, the combined initial stiffness is defined by (3.44).

$$K_{LP} = 10K_{linear} \tag{3.42}$$

$$\frac{1}{K_{combined}} = \frac{1}{K_{rotation}} + \frac{1}{10K_{rotation}} = \frac{11}{10K_{rotation}} \tag{3.43}$$

$$K_{combined} = \frac{10K_{rotation}}{11} = 4.04x10^4 \frac{kNm}{rad} \tag{3.44}$$

It is apparent that (3.38) is not yet satisfied. An adjustment may be made to K_{linear} in order to improve the model initial stiffness.

$$K_{adjusted} = \frac{11K_{rotation}}{10} \tag{3.45}$$

Then, setting the lumped plasticity element initial stiffness to a value of $10K_{adjusted}$, the combined stiffness is calibrated.

$$\frac{1}{K_{combined}} = \frac{1}{K_{adjusted}} + \frac{1}{10K_{adjusted}} = \frac{10}{11K_{rotation}} + \frac{10}{110K_{rotation}} = \frac{1}{K_{rotation}} \tag{3.46}$$

$$K_{combined} = K_{rotation} \tag{3.47}$$

This process derives the calibration equations suggested by Ibarra (Ibarra, 2003),

$$K_{linear} = \frac{n+1}{n} K_{rotation} \tag{3.48}$$

$$K_{LP} = (n+1)K_{rotation} \tag{3.49}$$

where $n \gg 1$. Using $n = 10$, the stiffness of the linear element is determined from (3.34) and used to calculate the lumped plasticity rotational spring stiffness.

$$K_{linear,10} = \frac{11}{10} K_{rotation} = 4.89x10^4 \frac{kNm}{rad} \tag{3.50}$$

$$K_{LP,10} = 10K_{linear} = 4.89x10^5 \frac{kNm}{rad} \tag{3.51}$$

If desired, the initial stiffness of the lumped plasticity element may be set to 100 times that of the linear elastic element. Notice, in this case, the initial stiffness of the linear element is closer to the value of $K_{rotation}$.

$$K_{linear,100} = \frac{101}{100} K_{rotation} = 4.49x10^4 \frac{kNm}{rad} \tag{3.52}$$

$$K_{LP,100} = 100K_{linear} = 4.49x10^6 \frac{kNm}{rad} \tag{3.53}$$

In Mercury, the *Elastic* constitutive model (assigned to the linear element) receives the Young's modulus as an input. The adjustment made in (3.48), therefore, is made to the elastic modulus, directly affecting the rotational stiffness.

$$E_{linear} = \frac{n+1}{n} E \tag{3.54}$$

For the cases when the rotation spring element initial stiffness is set to 10 and 100 times that of the linear element, the elastic modulus of the linear element is set to the value determined by (3.55).

$$\begin{aligned} E_{linear,10} &= 220GPa \\ E_{linear,100} &= 202GPa \end{aligned} \tag{3.55}$$

The rotational spring yield rotation may be determined using the initial stiffness (K_{LP}) and the yield moment (M_y).

$$\theta_y = \frac{M_y}{K_{LP}} \tag{3.56}$$

$$\theta_{y,10} = 0.00082 rad$$
$$\theta_{y,100} = 0.000089 rad \tag{3.57}$$

Similar to the initial stiffness, the hardening stiffness (K_s) and the post-capping stiffness (K_c) must each be determined for the rotational element to provide the desired response Figure 3.41. The desired values are calculated in (3.58) and (3.59).

$$K_s = \frac{M_u - M_y}{\theta_u - \theta_y} = 2.22x10^3 \frac{kNm}{rad} \tag{3.58}$$

$$K_c = \frac{M_r - M_u}{\theta_r - \theta_u} = -4.44x10^3 \frac{kNm}{rad} \tag{3.59}$$

Arranging (3.36) to solve for the rotational element stiffness, the values of K_s and K_c are determined for the cases when the initial rotation spring stiffness is 10 and 100 times the linear element initial stiffness.

$$\frac{1}{K_{LP}} = \frac{1}{K_{combined}} - \frac{1}{K_{linear}} \tag{3.60}$$

$$K_{s,10} = 2.33x10^3 \frac{kNm}{rad}$$
$$K_{s,100} = 2.34x10^3 \frac{kNm}{rad} \tag{3.61}$$

$$K_{c,10} = -4.07x10^3 \frac{kNm}{rad}$$
$$K_{c,100} = -4.04x10^3 \frac{kNm}{rad} \tag{3.62}$$

The peak and residual rotations of the rotational spring are then determined.

$$\theta_u = \theta_y + \frac{M_u - M_y}{K_s} \tag{3.63}$$

$$\begin{aligned} \theta_{u,10} &= 0.0438 rad \\ \theta_{u,100} &= 0.0429 rad \end{aligned} \tag{3.64}$$

$$\theta_r = \theta_u + \frac{M_r - M_u}{K_c} \tag{3.65}$$

$$\begin{aligned} \theta_{r,10} &= 0.157 rad \\ \theta_{r,100} &= 0.157 rad \end{aligned} \tag{3.66}$$

Recall from Chapter 3.4.2, the hysteretic capacity (E_{hys}) is a function of the yield point ($E_{hys} = \gamma M_y \theta_y$). By altering the yield point, the hysteretic capacity factor (γ) must also be modified to properly degrade the strength and stiffness modes.

$$\gamma_{adjusted} = \gamma \left[\frac{\theta_y}{\theta_{y,adjusted}} \right] \tag{3.67}$$

$$\begin{aligned} \gamma_{adjusted,10} &= 550 \\ \gamma_{adjusted,100} &= 5050 \end{aligned} \tag{3.68}$$

The Mercury input file, for the case when the lumped plasticity element initial stiffness (K_{LP}) is set equal to 10 times that of the linear element, is shown in Figure 3.43. Notice that the elastic Modulus for the linear elastic beam-column element is set to 220GPa.

```
%% Basic Information
unit    = {'kN', 'mm', 'sec'};
strmode = {2, 3};
%% Node Information
nodcoord   = {1, 0,   0;
              2, 0,   0;
              3, 0, 3600};
%% Boundary Information
constraint = {1, 1, 1, 1};
%% Material Information
materials = {
    {1, 'Hysteresis', 4e5, 0.00082, -4e5, -0.00082, 5e5, 0.0438, -5e5, -0.0438, 4e4, 0.157, -4e4, -0.157, 550, 1, 0}
    {2, 'Elastic', 220, 0, 0}       % elastic beam-column
    {3, 'Elastic', 1000, 0, 0}        % rigid axial spring
    {4, 'Elastic', 1000, 0, 0}       % rigid shear spring
            };
%% Section Information
sections = {1, 'General2D', {2, 40000, 0, 1.33333333e8};
            };
%% Element Information
elements = { {1, 'zerolength2d', 1, 2, 3, 4, 1, pi/2}
             {2, 'elasticbeamcolumn', 2, 3, 1, 0} };
%% Force Information
DispInput = load('cyclic_Siamak.txt');
row = size(DispInput,1);
div = 1;
total = floor(row/div);
factor = 6;
for i = 1:total
    DispCell{i} = factor*DispInput(i*div);
end
forces = { 1, 'DispCtrl', {3, 1, DispCell
                           };
         };
clear DispInput;clear row;clear div;clear DispCell
% ========================================================================
%% Iteration Information
iterations = {1, {'NewtonRaphson', 10, 1.0e-8, 'dispNorm'}
              2, {'ModifiedNewtonRaphson', 20, 1.0e-8, 'dispNorm'}
              3, {'Initial', 30, 1.0e-8, 'dispNorm'}
              };
```

Figure 3.43 Lumped Plasticity Initial Stiffness 10X Elastic Stiffness

A comparison plot of the base shear vs. the lateral drift is provided to display both cases ($K_{LP}=10K_{linear}$ and $K_{LP}=100K_{linear}$). The similarity of the results for the two calibrated tests indicates that a value of n=10 is suitable for Equations (3.48) and (3.49).

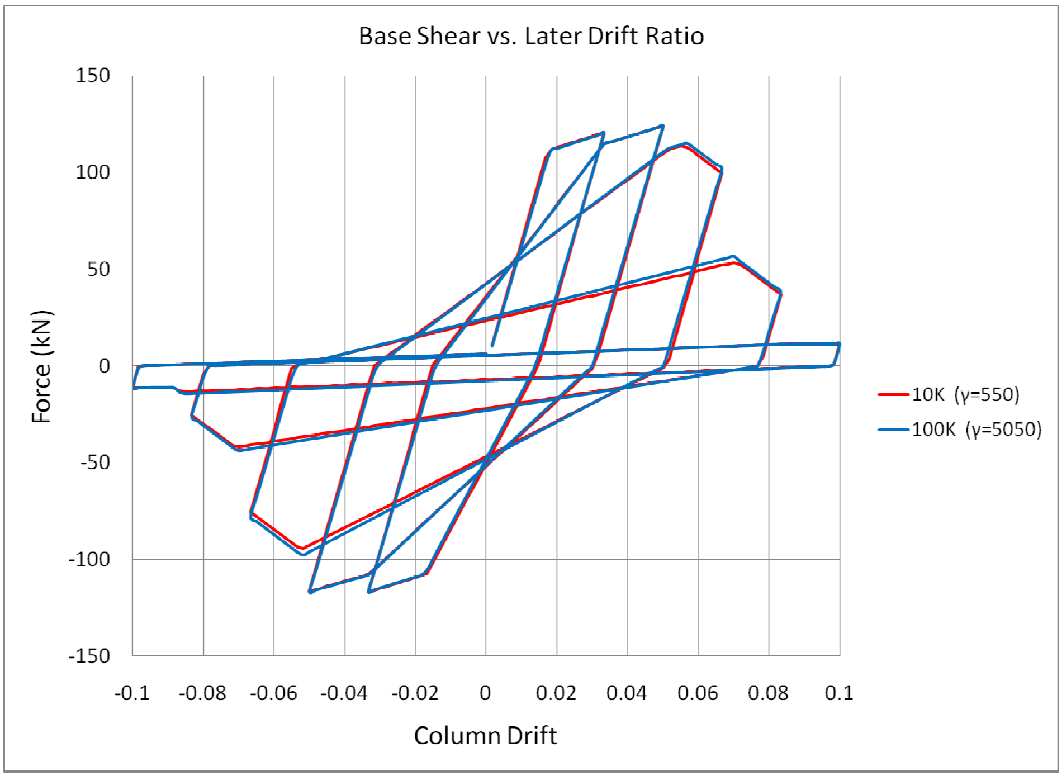

Figure 3.44 Initial Stiffness Calibration

Adjustments were not made to the degradation rate factor (c) in this section because the assigned model value is 1. When values other than 1 are assigned for the degradation rate, calibration must also incorporate an adjustment to this variable.

3.6 Chapter Summary

This chapter detailed two hysteretic constitutive models adopted for implementation in Mercury. The implemented peak-oriented (Chapter 3.2.1) and pinching (Chapter 3.2.2) models account for cyclic strength loss due to the four strength and stiffness modes suggested by Ibarra (Ibarra, 2003):

1. Basic strength degradation—a *strength* degradation modes which translates the strain hardening branch by reducing the yield strength and hardening modulus (stiffness)
2. Post-capping strength degradation—a *strength* degradation mode that translates the post-peak branch by reducing the peak strength

3. Unloading stiffness degradation—a *stiffness* degradation mode that reduces the unloading modulus (stiffness)
4. Accelerated reloading stiffness—a *stiffness* degradation mode that reduces the reloading modulus (stiffness) by increasing the target deformation

Chapter 4 describes the implementation of a column shear failure constitutive model.

4 Shear Failure Model (Limit State Material)

4.1 Introduction

The lumped plasticity model outlined in Chapter 3.5 included flexural strength degradation only, assigning rigid linear elastic material properties to the shear and axial springs. Shear and axial failure modes, therefore, are not captured in the model, although their effects are critical given the severity of structural response under both shear and axial loss of strength. In fact, column shear failure is the most severe of failure modes, rapidly leading to loss of axial load carrying capacity due to reduced sectional shear resistance.

A computational tool capable of predicting the onset of structural failure is a critical tool for engineers in the field of performance-based earthquake engineering (PBEE). This is especially true when considering the vulnerability of older reinforced concrete structures to collapse due to shear and axial failure. These structures are particularly vulnerable to collapse because their light transverse reinforcement does not promote ductile behavior under transverse loading, a problem in older (pre 1970s) reinforced concrete structures designed previous to current seismic design provisions. As a tool for predicting the behavior of such a structure, Elwood and Moehle (Elwood & Moehle, 2003) developed an empirical model to capture the onset of failure and the subsequent strength degradation leading to collapse. Limit state models for both shear and axial failure have been suggested based on the column interstory drift at which failure is expected for the failure mode of interest.

The shear limit state model has been chosen for implementation in Mercury as a computational fuse for simulating shear failure of reinforced concrete columns. The model has applications to both distributed

and lumped plasticity modeling, although it is undoubtedly most beneficial as a complimentary component to a lumped plasticity model similar to the model discussed in Chapter 3.5.

4.2 Shear Limit State Model

The shear limit state material model is adopted as a means to provide another tool, in addition to the strength degradation models (Chapter 3.4.2), to investigate structure loss of strength. It is important to point out that the model is empirical in nature and, therefore, limited in its applications. The model applies to columns with low transverse reinforcement ratios ($\rho'' \leq 0.002$) expected to fail in shear after flexural yielding has occurred. A shear zero-length element is used to define a shear response constitutive model. Developing Figure 3.42 to include the shear limit state model, a lumped plasticity model would include an additional zero-length element at the top of the column (node 3) with Mercury's *Hysteresis* model assigned to the rotational spring, a rigid linear elastic model assigned to the axial spring, and the limit state model assigned to the shear spring Figure 4.1.

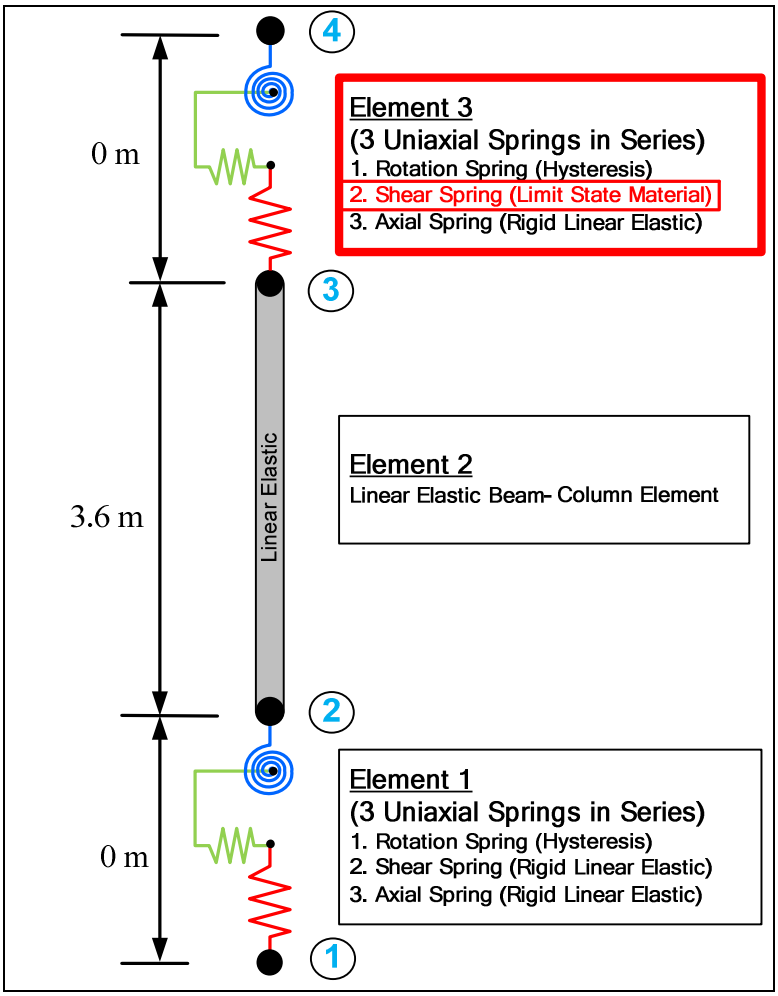

Figure 4.1 Lumped Plasticity Model with Shear Failure

The Limit State shear model defines the force-deformation curve as three lines (Figure 4.2): an initial elastic branch (0-1), a degrading (softening) strength branch (1-2), and a residual strength branch (2-3).

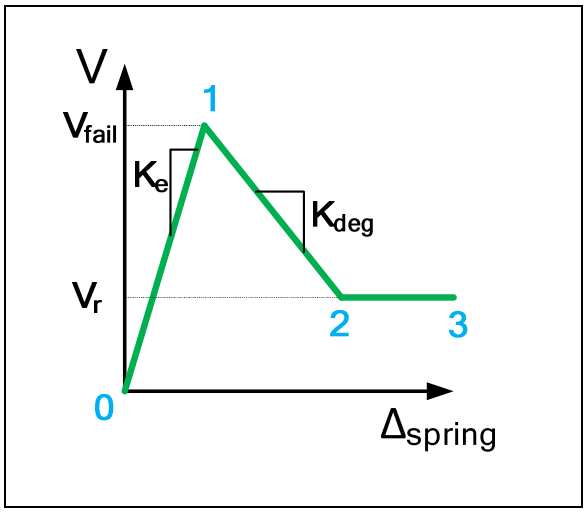

Figure 4.2 Shear Spring Force-Deformation Relationship

A shear failure (drift capacity) surface (Figure 4.3) is defined by Equation (4.1), suggested by Elwood and Moehle (Elwood & Moehle, 2003). The failure surface defines the lateral drift $\left(\frac{\Delta}{L}\right)$ at which the column is expected to fail in shear *after flexural yielding has occurred*. It is a function of the transverse reinforcement ratio (ρ"), shear strength (v), concrete compressive strength (f'_c), axial load (P), and gross sectional area (A_g). Because the model is empirical, the failure surface is defined separately for S.I. and English units

$$\frac{\Delta_s}{L} = \frac{3}{100} + 4\rho'' - \frac{1}{40}\frac{v}{f'_c} - \frac{1}{40}\frac{P}{A_g f'_c} \geq \frac{1}{100} \quad (S.I.Units)$$
$$\frac{\Delta_s}{L} = \frac{3}{100} + 4\rho'' - \frac{1}{500}\frac{v}{f'_c} - \frac{1}{40}\frac{P}{A_g f'_c} \geq \frac{1}{100} \quad (English\,Units) \qquad (4.1)$$

where

Δ_s = column interstory displacement at shear failure

L = column height

ρ'' = transverse reinforcement ratio

A_g = column cross-sectional area [mm^2 or in^2]

ν = shear capacity (V/A_g) [MPa or psi]

f'_c = concrete compressive strength [MPa or psi]

P = column axial load

The model operates by comparing the shear force-deformation response at the top of the column (Figure 4.1 node 4) to the drift capacity surface. Figure 4.3 separates the column and shear spring responses. The column lateral deformation (Δ_{BC}) is the displacement at node 4 and the spring deformation (Δ_{spring}) is the horizontal displacement from node 3 to node 4. Until the column reaches the shear failure surface (Figure 4.3 point 2_c), the shear spring response is linear elastic (0_s-1_s). Once the failure surface is reached, the shear spring stiffness is degraded (1_s-2_s) from the peak-point (V_{fail}) at which shear failure was predicted by the failure surface.

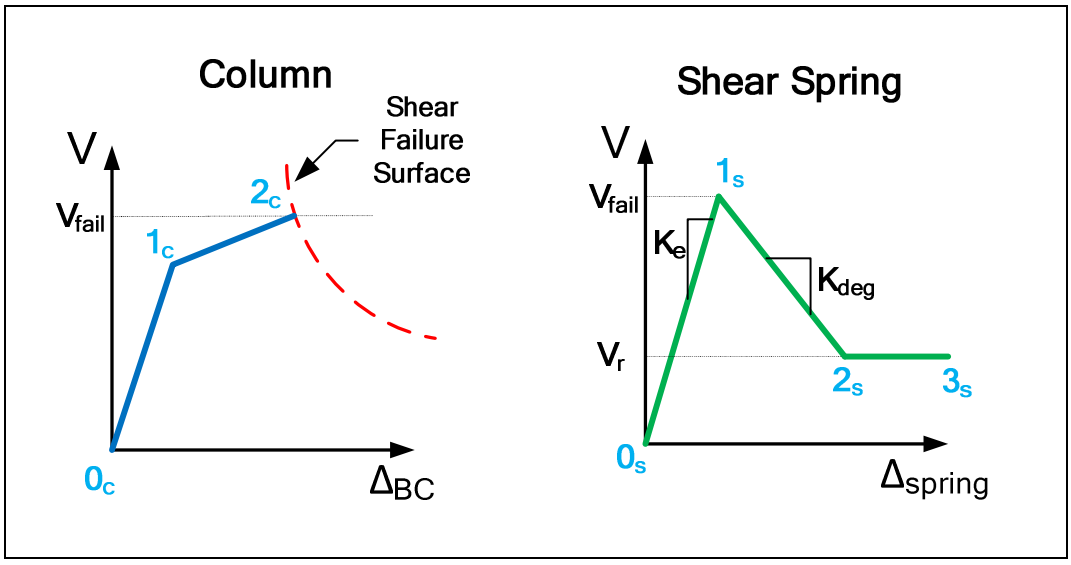

Figure 4.3 Shear Spring Response

Figure 4.4 details the column response (Δ_{BC}), the spring response (Δ_{spring}), and the total response (Δ_{total}): a) before the failure surface is met, b) during the step when the failure surface is met, and c) when the

shear stiffness is degraded. The blue beam-column element is composed of element 1, element 2, and the axial and rotational springs of element 3 (Figure 4.1). If the initial stiffness of the shear spring is set to a very high value, the pre-failure total response is equal to the column flexural response ($\Delta_{total} = \Delta_{BC}$). During the time step when failure is detected by the shear failure surface, the shear spring stiffness becomes negative and the total response is a combination of the column flexural response and the uniaxial shear spring response as shear deformations are no longer negligible in the model. When the failure surface is met and the shear spring stiffness becomes negative, as lateral deformations increase due to the degrading shear stiffness, the column element is allowed to unload. This results in reduced column flexural deformation as the spring deformation increases.

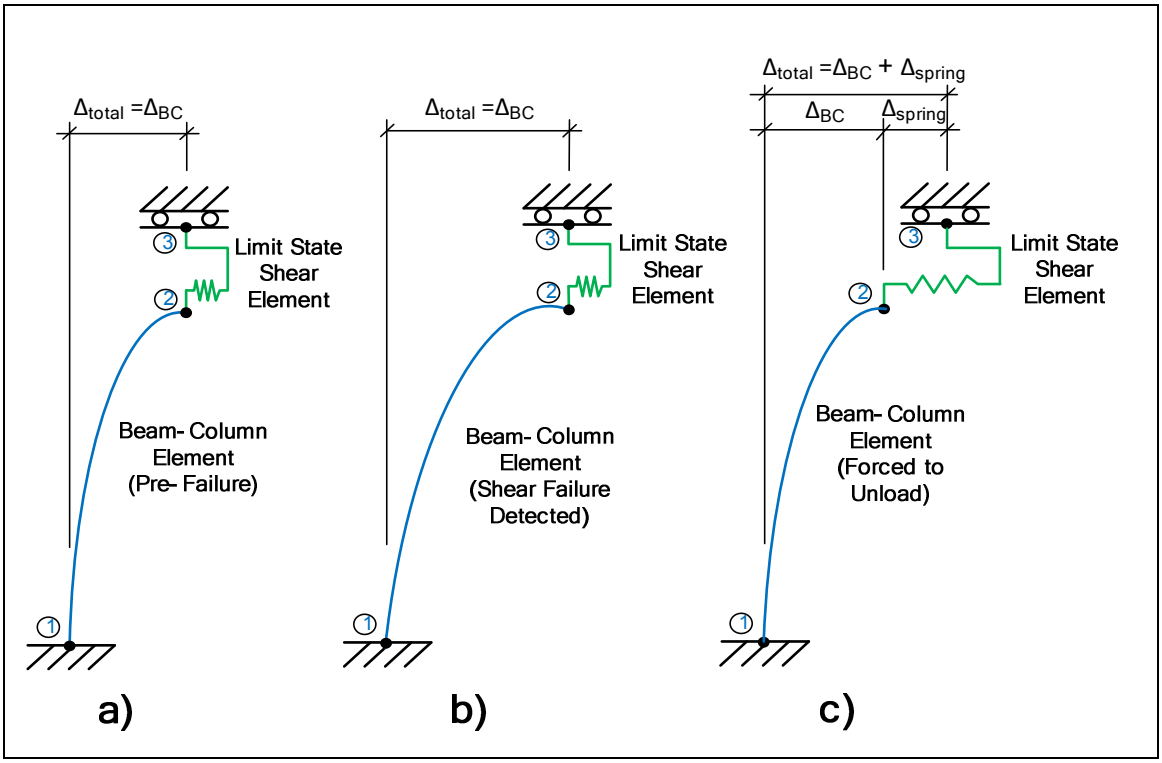

Figure 4.4 Limit State Column Response

State determination of the pre-failure response, and the response when shear spring stiffness degradation is initiated, uses the combined stiffness of the column element and the shear spring element.

$$\frac{1}{K_t} = \frac{1}{K_{BC}} + \frac{1}{K_{spring}} \tag{4.2}$$

If the spring is initially very stiff, the total stiffness is equal to K_{BC}.

$$\frac{1}{K_t} = \frac{1}{K_{BC}} + \frac{1}{\infty} = \frac{1}{K_{BC}} \tag{4.3}$$

$$K_t = K_{BC} \tag{4.4}$$

Once failure is initiated, the shear spring stiffness is degraded (K_{deg}) and the column begins to unload ($K_{BC} = K_{unload}$). The total degraded stiffness then becomes the combination of these (Figure 4.5).

$$\frac{1}{K_{deg}^t} = \frac{1}{K_{BC}} + \frac{1}{K_{spring}} = \frac{1}{K_{unload}} + \frac{1}{K_{deg}} \tag{4.5}$$

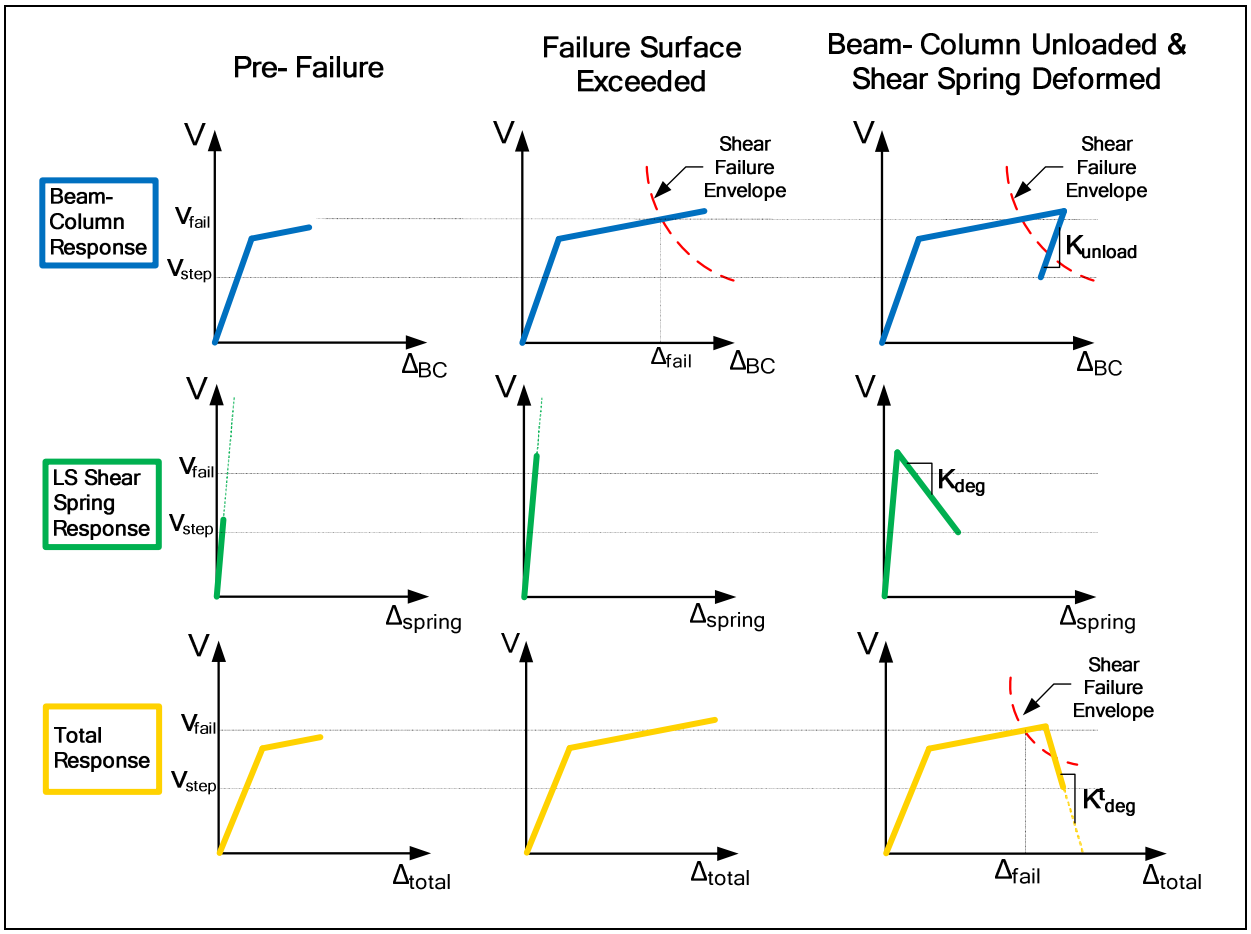

Figure 4.5 Limit State Individual Responses

4.3 Mercury Implementation

4.3.1 Limit State Input Variables

The developed model calls for inputs to define the drift capacity surface, the shear spring pre-failure stiffness, the shear spring degraded stiffness, and the residual strength factor. The input variables are summarized in Table 4-1.

Variable	Description
A_g	Gross Cross-Sectional Area
ρ"	Transverse Reinforcement Ratio
V_u	Column Shear Strength
f'_c	Concrete Compressive Strength
ele_{ID}	Associated Beam-Column Element ID
$node_{BC,i}$	Associated Beam-Column node "i" ID
$node_{BC,j}$	Associated Beam-Column node "j" ID
K_{stiff}	Initial Spring Stiffness
K_{deg}	Degrading Spring Stiffness
λ	Residual Strength Factor

Table 4-1 Limit State Input Variables

The cross-sectional area, transverse reinforcement ratio, shear strength, and concrete compressive strength are each used to define the failure surface (Equation (4.1)). In addition, at each loading increment, the column axial force is included for determination of the failure surface. The beam-column element and node ID's are used to monitor the column lateral drift for state determination. The residual strength factor (λ) is used to define the residual strength (V_r) as a factor of the peak shear force (V_{fail}) at which the failure surface is reached.

$$V_r = \lambda V_{fail} \tag{4.6}$$

Implementation of this model poses a particular challenge due to the exchange of information between the beam-column element and zero-length element. The Limit State constitutive model must receive the beam-column lateral deformation as an input. In the context of object oriented programming, this exchange of information may be problematic, requiring modification to the Mercury source code.

4.3.2 Validation

A lumped plasticity validation model is used with rotational springs at the top and bottom of the column, an elastic element, and a Limit State shear spring at the top node (Figure 4.6). For clarity, rigid axial springs at the top and bottom nodes and a rigid shear spring at the bottom node are not shown in the figure. The constitutive model used for the rotational springs is the *Hardening* model (Kang, 2010) given by the moment-rotation plot provided in the figure. *Hardening* material is chosen for validation to eliminate post-peak computational issues. As mentioned in Chapter 4.1, use of a calibrated hysteresis model (Chapter 3.4.2) is recommended for lumped plasticity modeling.

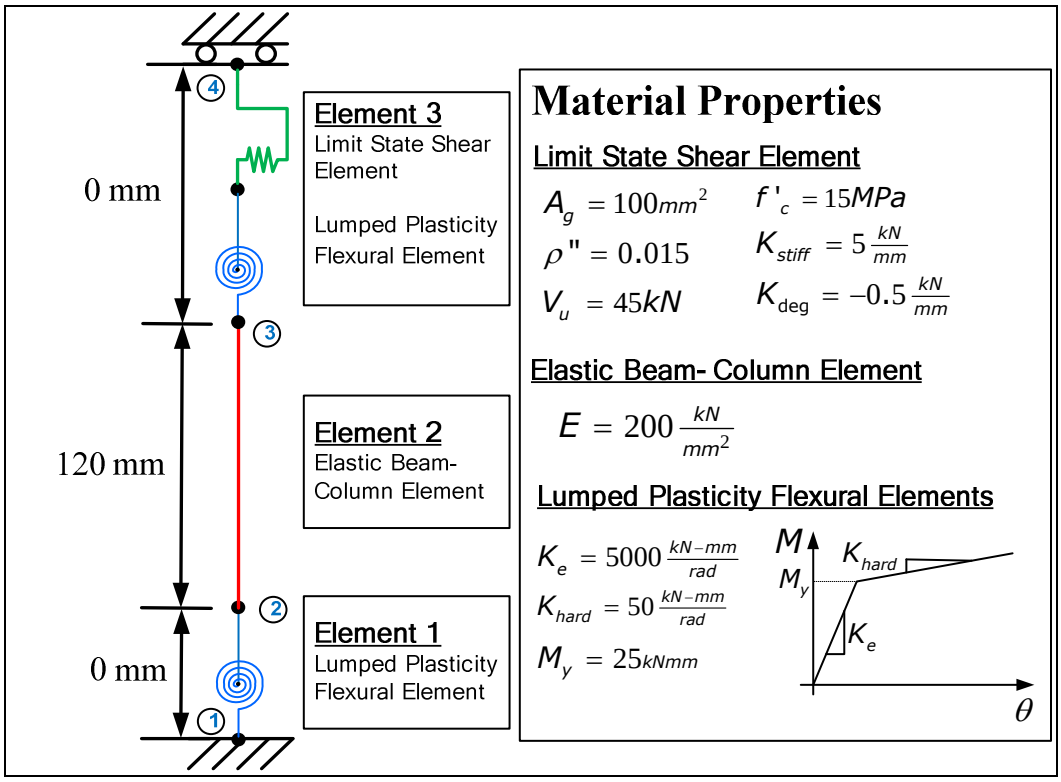

Figure 4.6 Lumped Plasticity Limit State Model

In order to capture the strength degradation of the shear spring element, monotonic lateral displacements are applied at node 4 following the pattern of Figure 4.7.

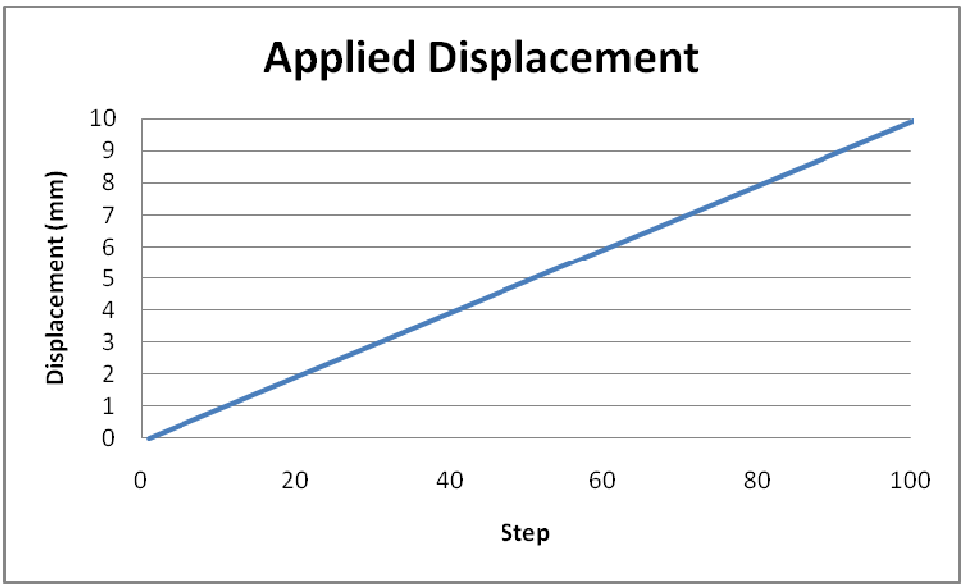

Figure 4.7 Displacement Control Prescribed at Node 4

In order to develop Mercury as a tool for lumped plasticity modeling, a few necessary changes were made to the source code. First, when a user defines a zero-length element in Mercury, the element is a series of 3 springs: an axial spring, a shear spring, and a rotational spring. Previously, a user was not capable of defining different material properties for each of the springs. Instead, users had the choice of either assigning the same material property to each of the three springs, or assigning the material property to a particular spring and allowing free deformation in the other springs (Kang, 2010). For this reason, changes were made to allow a user to assign the Limit State model for the shear spring while also defining a stiff axial spring and a nonlinear model for the rotational spring. These changes are vital for lumped plasticity modeling, but the most difficult component of the implementation of the Limit State model concerns the state determination process in Mercury.

In order to capture the degrading behavior of the Limit State shear spring, it is necessary for flexural unloading to occur in the column (Figure 4.5), while the shear spring response follows the degrading branch. Unfortunately, in the currently developed model, the opposite is occurring. While the shear spring stiffness degrades, the spring is forced to unload and the rotational springs and elastic element

continue to load. Figure 4.8 and Figure 4.9 detail the rotational spring response at the base of the column (element 1 M-θ) and the Limit State shear spring response at the top of the column (element 3 V-Δ). When the failure surface is met, the shear spring stiffness becomes the degraded stiffness ( ) while the column element continues to load. As a result, the spring unloads with the degraded stiffness.

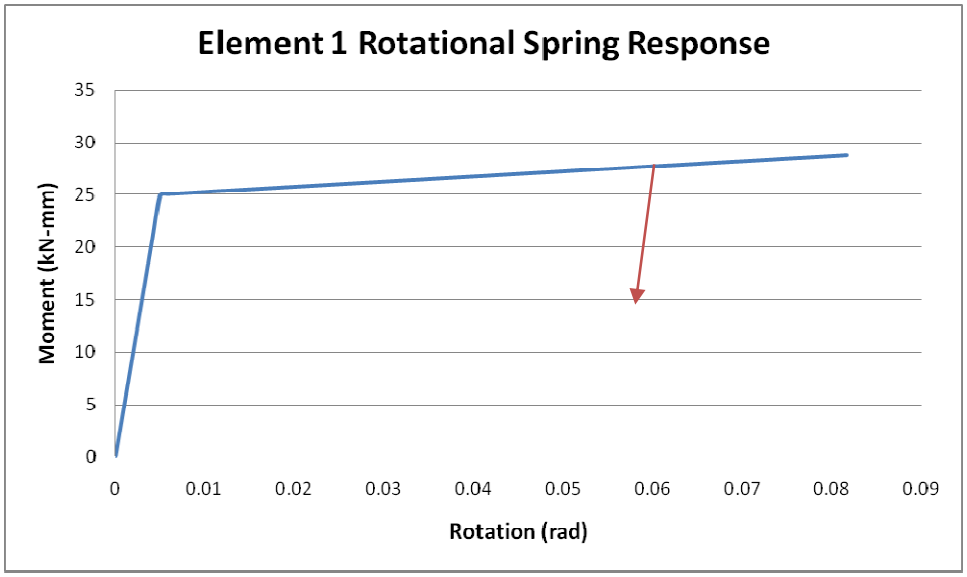

Figure 4.8 Limit State Flexural Response

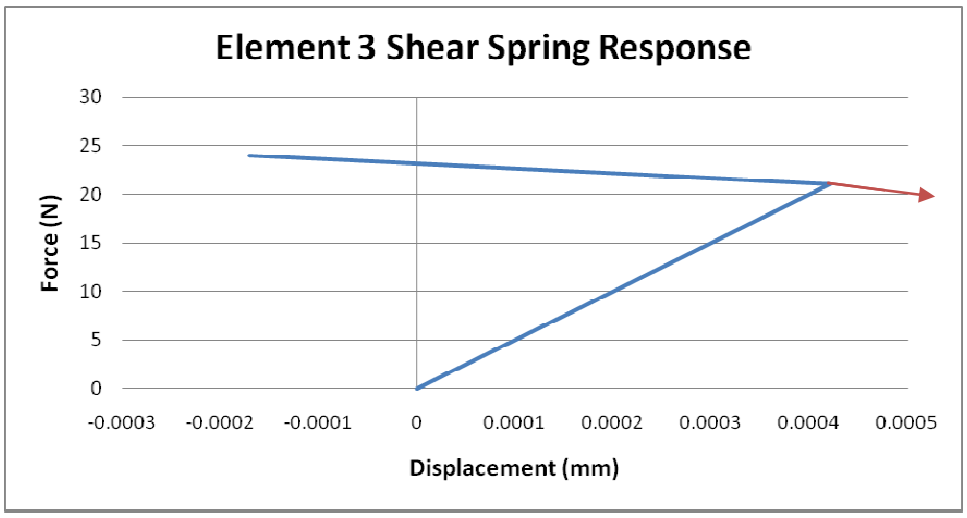

Figure 4.9 Limit State Shear Response

```
%% Basic In formation
unit    = {'N', 'mm', 'sec'};
strmode = {2, 3};
%% Node Information
nodcoord   = {1, 0,    0;
              2, 0,    0;
              3, 0,    120;
              4, 0,     120;
             };
%% Boundary Information
constraint = {1, 1, 1, 1
              };
%% Material Information
% {mattag, 'LimitState', Ag, trr, V_u, fc, eletag, sn, en, K_stiff, K_deg, lamba}
materials = { {1, 'Hardening', 20000000, 0.01, 200000, 0, 0};
              {2, 'LimitState', 100, 0.015 , 45000, 15, 2, 2, 3, 5000, -500, 0.2};
              {3, 'Elastic', 200, 0, 0};
              {4, 'Elastic', 100000, 0, 0};
              {5, 'Hardening', 5000, 25, 50, 0, 0};
            };
%% Section Information
sections =  {1, 'General2D', {3, 20, 0, 1400};
             };
%% Element Information
elements = { {1, 'zerolength2d', 1, 2, 4, 4, 5, pi/2};
             {2, 'elasticbeamcolumn', 2, 3, 1, 0};
             {3, 'zerolength2d', 3, 4, 4, 2, 5, pi/2};
           };
%% Force Information
forces = { 1, 'DispCtrl', { 4, 1, linearforceinput(0,0.1,10);
                          };
         };
%% Iteration Information
iterations = { 1, {'NewtonRaphson', 100, 1e-8, 'dispNorm'} };
total = 99;
```

Figure 4.10 Limit State Mercury Input File

Modifications will need to be made to address the state determination process in order to ensure that the rotational elements are unloaded while the Limit state shear spring element is subjected to lateral deformations. A primary concern is that the total degraded stiffness uses the strain hardening slope (K_s) of the beam-column rather than the unloading stiffness (K_{unload}) which would be closer to the initial elastic stiffness.

$$\frac{1}{K_{deg}^{t}} = \frac{1}{K_{BC}} + \frac{1}{K_{spring}} = \frac{1}{K_s} + \frac{1}{K_{deg}} \tag{4.7}$$

4.4 Chapter Summary

This chapter detailed the Mercury implementation of a column shear failure constitutive model based on the research of Elwood and Moehle (Elwood & Moehle, 2003). The model initiates stiffness degradation of a zero-length uniaxial shear spring once the empirical drift capacity model has been exceeded. The drift capacity is predicted by Equation(4.1) (shown below), representing the column interstory drift at which shear failure is expected, following flexural yielding.

$$\frac{\Delta_s}{L} = \frac{3}{100} + 4\rho'' - \frac{1}{40}\frac{\nu}{f'_c} - \frac{1}{40}\frac{P}{A_g f'_c} \geq \frac{1}{100} \qquad (S.I.Units)$$

$$\frac{\Delta_s}{L} = \frac{3}{100} + 4\rho'' - \frac{1}{500}\frac{\nu}{f'_c} - \frac{1}{40}\frac{P}{A_g f'_c} \geq \frac{1}{100} \qquad (English Units)$$

Validation results indicated that stiffness degradation initiated unloading of the zero-length shear spring rather than unloading of the beam-column element. Development of the models discussed in this thesis were intended to take place in an object oriented format with no maintenance to the Mercury source code; however, failure to capture the intended response indicates a need to modify the state determination process. Implementation will depend on the assistance of the original developer of Mercury (Kang) to address implementation problems.

5 Constitutive Driver

5.1 Introduction

Constitutive models play a critical role in nonlinear analysis, yet their implementation is challenging. Furthermore, it is often desirable to compare material responses to a strain history, either for separate constitutive models or for a single model with varying parameters. Such a task may be achieved through a constitutive driver which enables visualization of the material response at a single Gauss point when subjected to a strain history. This section describes the development of such a driver.

The Constitutive Driver has been developed in MATLAB using the GUIDE tool for developing Graphical User Interfaces (GUI) (a detailed explanation about building a GUI in MATLAB is provided in Chapter 6.3). The application provides access to each of the constitutive models implemented in Mercury, including the *Hysteresis* and *HysteresisPinch* models outlined in Chapter 3.4.2.

1. Elastic Model
2. Kinematic Hardening Model
3. Simplified Bilinear Model
4. Modified Giuffre-Menegotto-Pinto Model
5. Modified Kent and Park Model
6. Anisotropic Damage Model
7. Anisotropic Damage Model with Permanent Strains
8. Hysteresis Model
9. Hysteresis Model with Pinching

For an in-depth description of the models, the reader is referred to the text authored by Saouma (Saouma, 2011). The layout of the GUI is explained first, followed by a description of running the driver and accessing example constitutive model properties.

5.2 Environment

The Constitutive Driver is a single interface composed of a user input region, an output plot region, two dropdown menus, and a toolbar with plot and viewing tools.

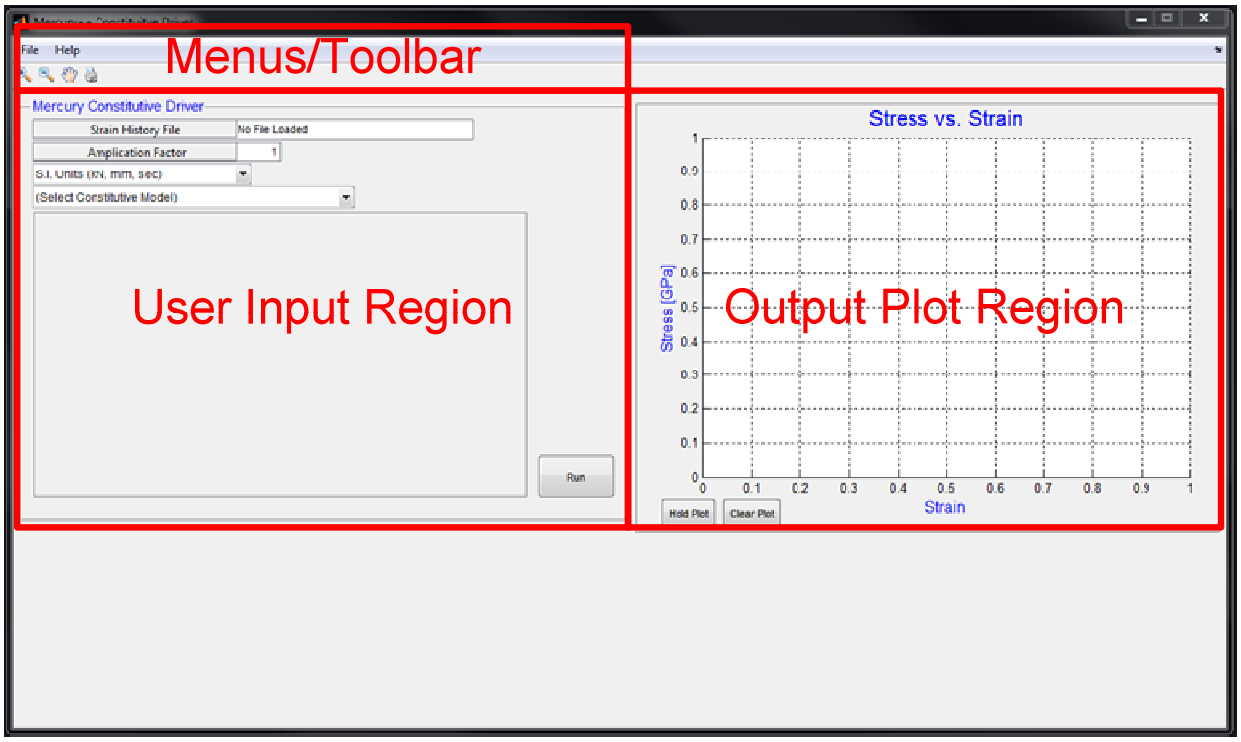

Figure 5.1 Constitutive Driver Environment

5.2.1 Menus and Toolbar

In the upper left corner, two dropdown menus can be accessed. The *Help* menu provides a link to the Mercury User's Manual with detailed information about each of the constitutive models. The *File* menu is used to load user defined strain histories (driver files) and to load files containing example constitutive model input values.

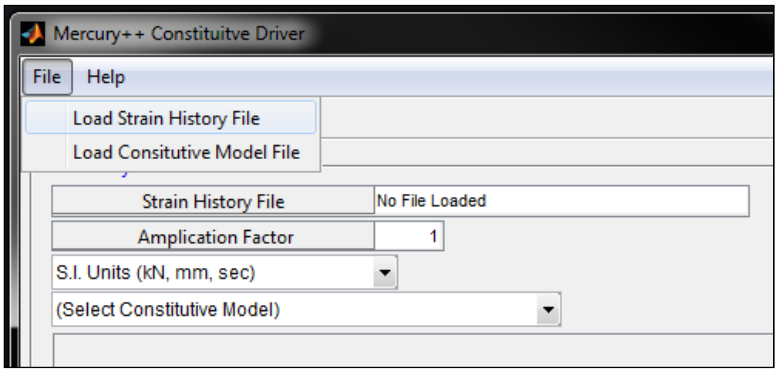

Figure 5.2 Constitutive Driver File Menu

A toolbar is located beneath the dropdown menus with tools to print the contents of the plot output region and to zoom and pan the output region plot.

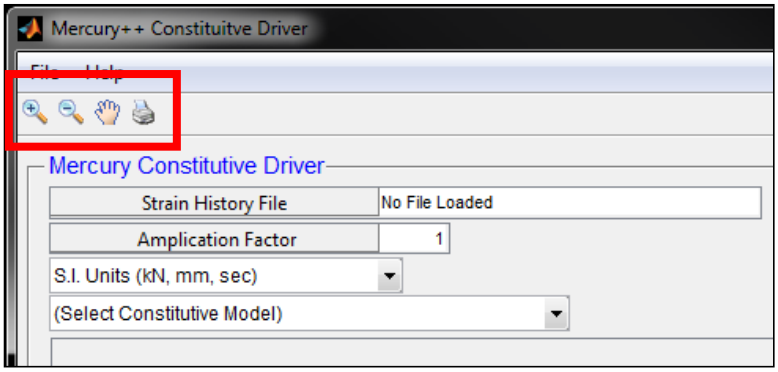

Figure 5.3 Constitutive Driver Toolbar

5.2.2 User Input

The user input region contains a box with the filename of the user defined strain history (driver) file. Another box labeled “Amplification Factor” receives a user input value to increase or reduce the amplitude of the strain history. Two popup menus are used to define the model units and select the constitutive model to be used, and a pushbutton is used to run the Constitutive Driver when the user has input all necessary information.

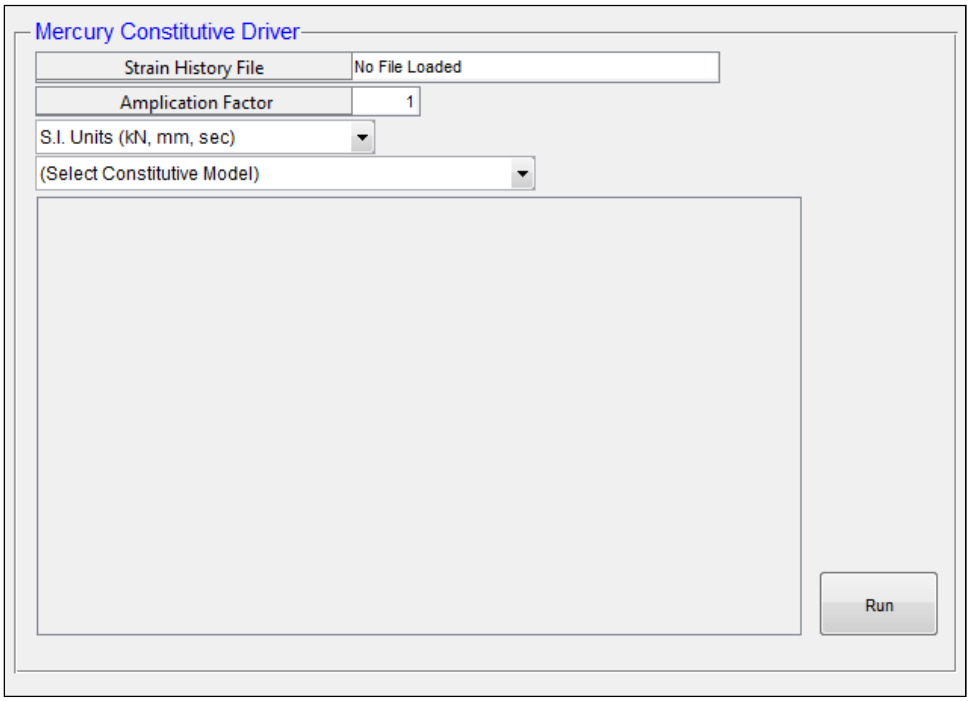

Figure 5.4 User Input Region

When a constitutive model is selected from the popup menu labeled "(Select Constitutive Model)", a table appears to receive relevant input parameters for the corresponding model.

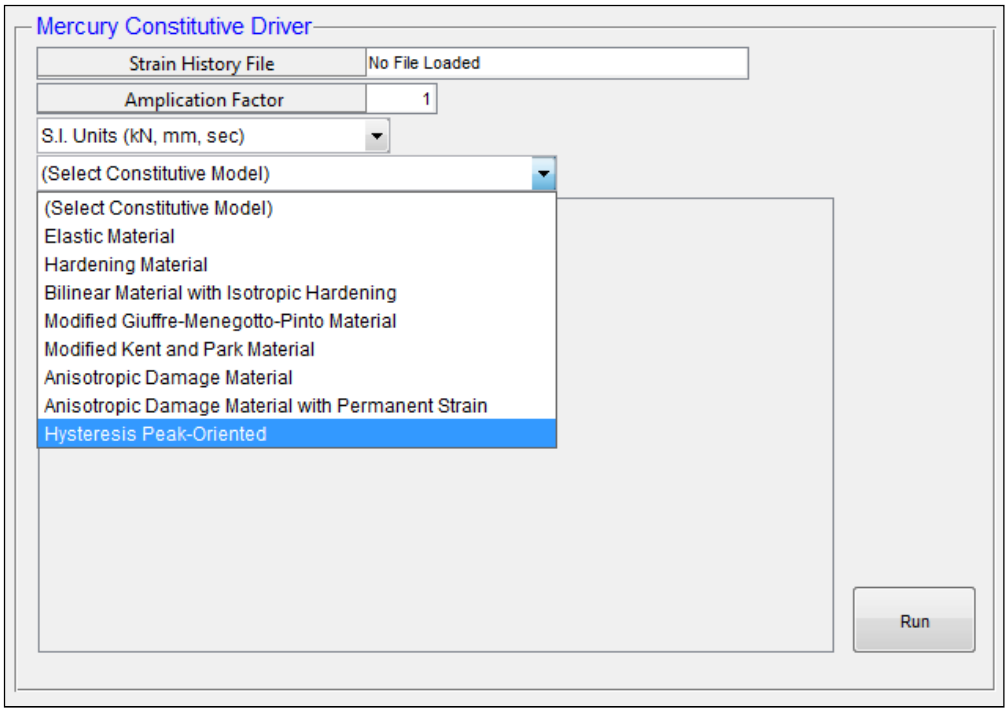

Figure 5.5 Selecting Constitutive Model

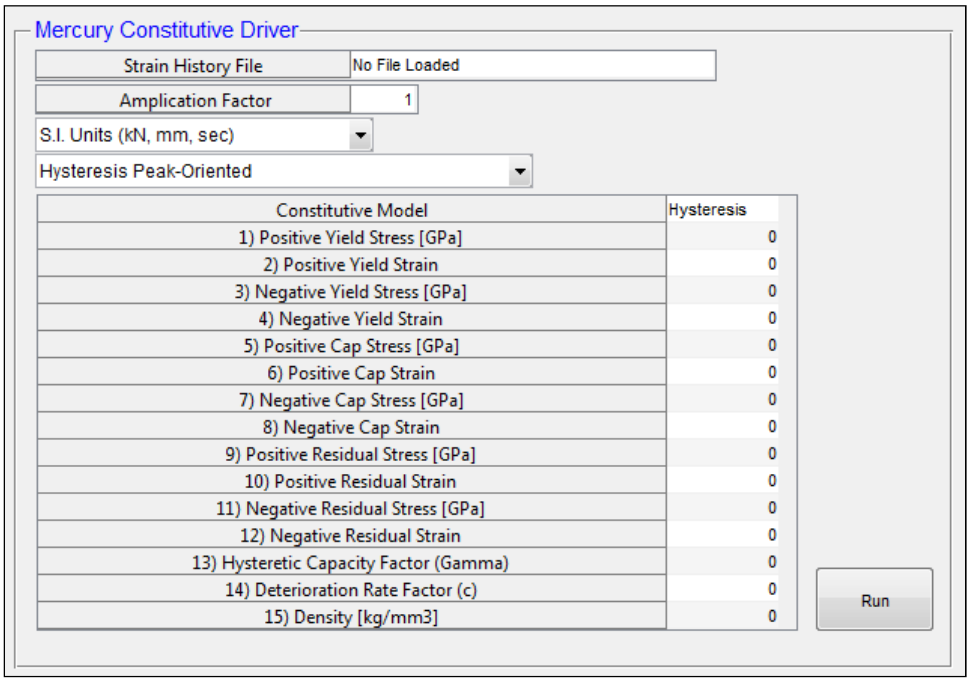

Figure 5.6 Model User Input

5.2.3 Driver Output

The output region handles all plotting results. When a driver file is loaded from the *File* dropdown menu, the strain history is plotted automatically.

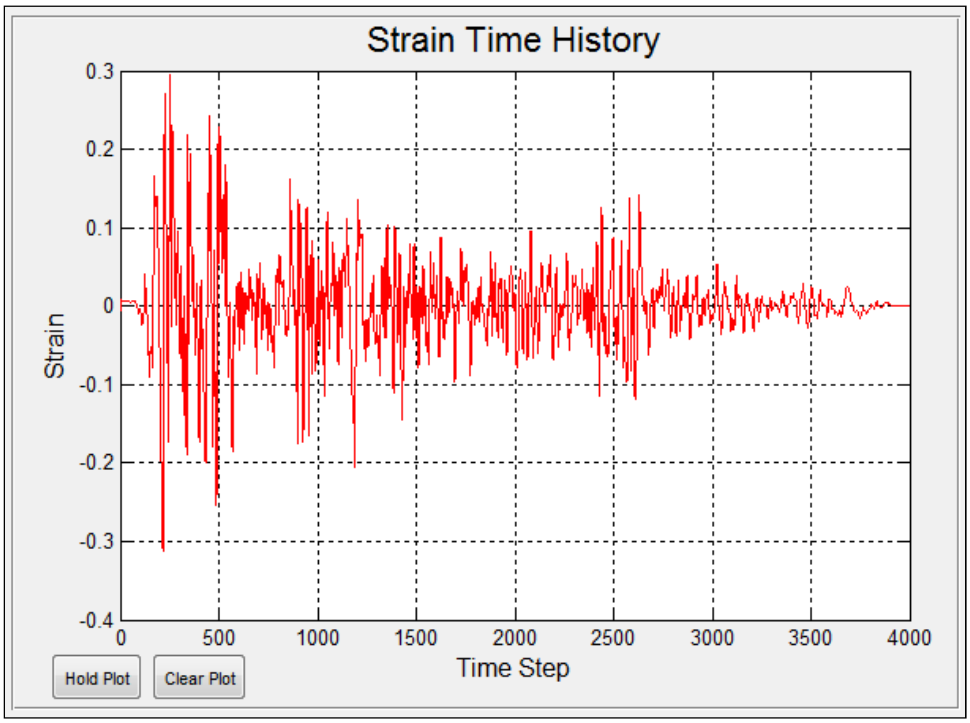

Figure 5.7 Constitutive Driver Output Region

Similarly, when the driver is run, the output plot region displays the stress-strain results, at a single Gauss point, in the units selected from the user input region.

The output region contains two pushbuttons:

1. Hold Plot - to hold a plot if the user desires to plot multiple outputs for comparison
2. Clear Plot – to clear the output plot region

The *Hold Plot* pushbutton should be used to compare models or for an understanding of the effects of changing model variables.

5.3 Running the Constitutive Driver

5.3.1 Loading a Strain History File

The application is used by loading a driver ASCII (.txt) file. The driver file represents the strain prescribed on an element, with the material properties assigned by the selected constitutive model, at a single Gauss point.

To load a driver file, the menu item *Load Strain History File* is selected from the *File* dropdown menu, prompting the user to select a .txt strain history input file. A few examples are provided in the "Examples" folder (*cyclicwave.txt* & *ElCentro_g_0_01.txt*). When a file is selected, the filename is displayed in the user input region and plotted in the plot region.

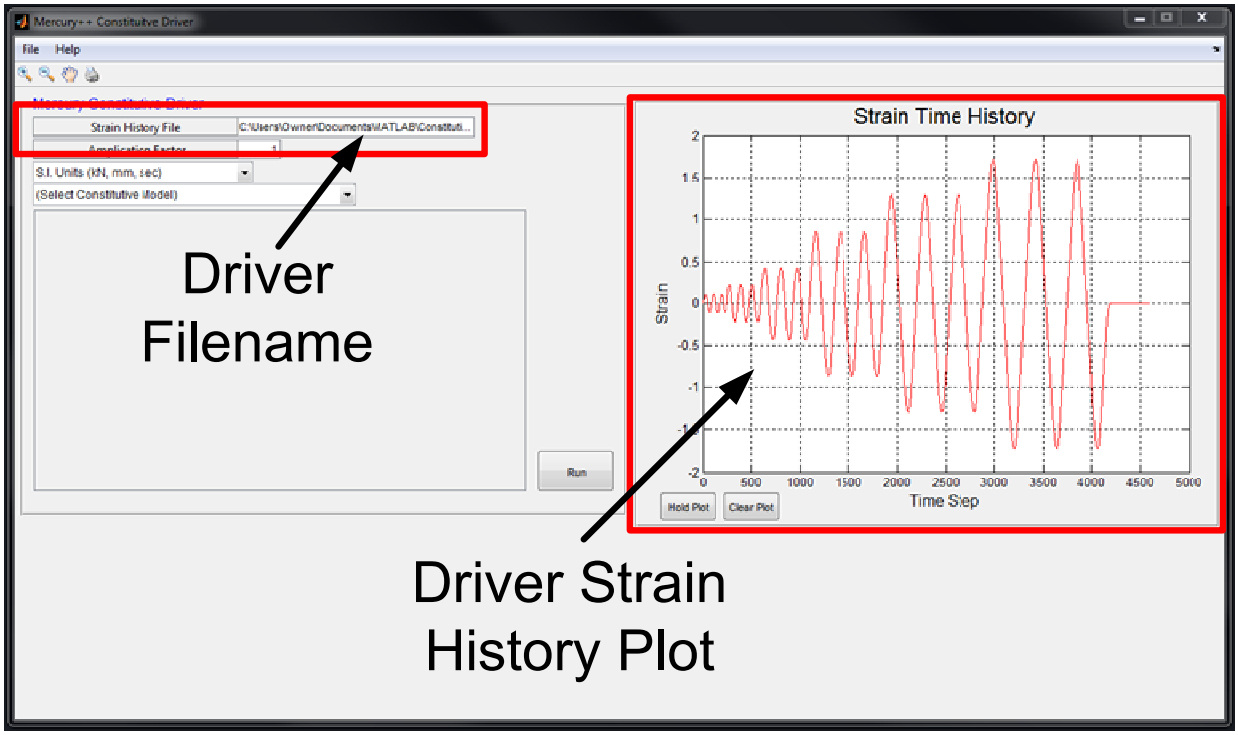

Figure 5.8 Strain History Loaded

Altering the value in the "Amplification Factor" input region changes the amplitude of the loaded strain history file. Notice in Figure 5.8 the strain history maximum amplitude is between 1.5 and 2. If an

amplitude factor of 20 is input, the maximum amplitude is factored 20 times (Figure 5.9) to a maximum value between 30 and 40.

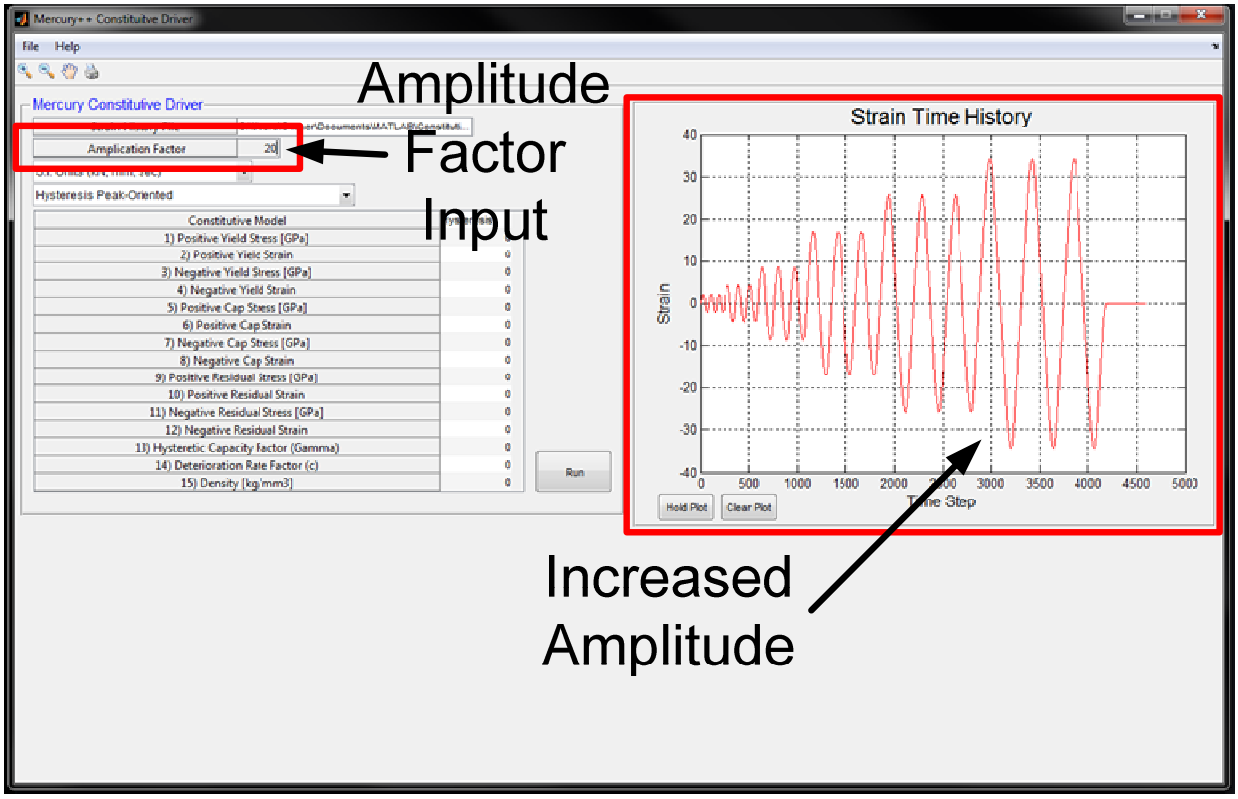

Figure 5.9 Adjusting Driver Amplitude

5.3.2 Obtaining Example Constitutive Model Properties

When the menu item *Load Constitutive Model File* is selected from the *File* dropdown menu, the user is prompted to select a .mat file with sample constitutive model values. An example file is included in the "Examples" folder (*Mercury_Materials_Input.mat*). When a file is selected, a region appears below the user input region, displaying all loaded example model values. A popup menu appears in the user input region with the text "(Select Existing Model Data)", allowing the user to select one of the example models to run or to modify.

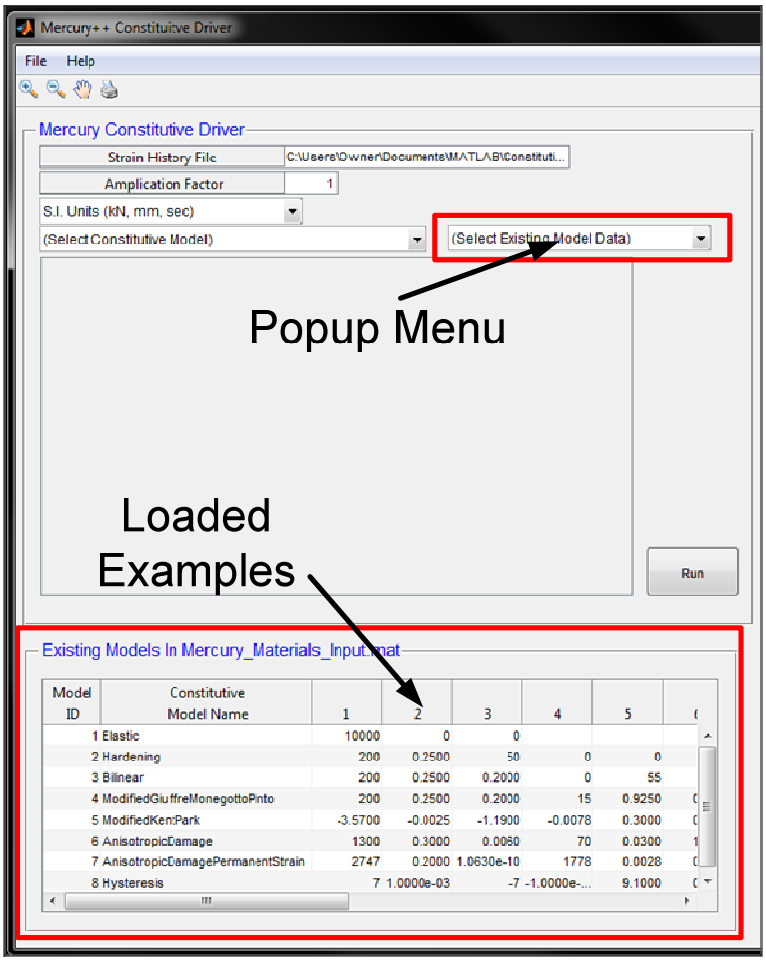

Figure 5.10 Loading Existing Model Data

When a selection is made from the new popup menu, the example data is loaded in the user input region table.

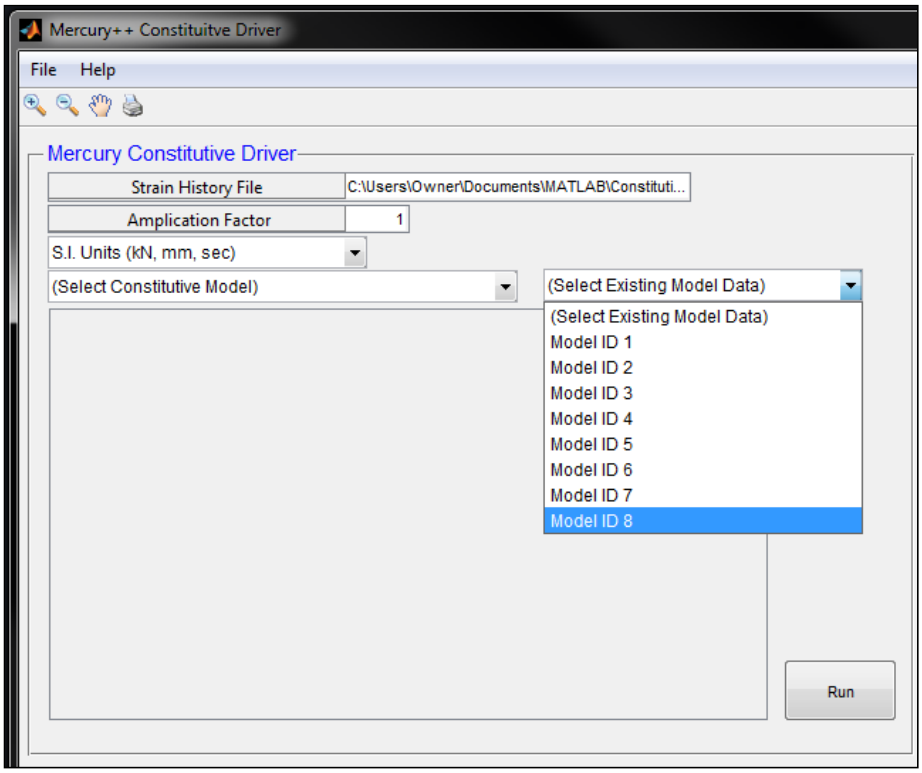

Figure 5.11 Using Existing Model Data

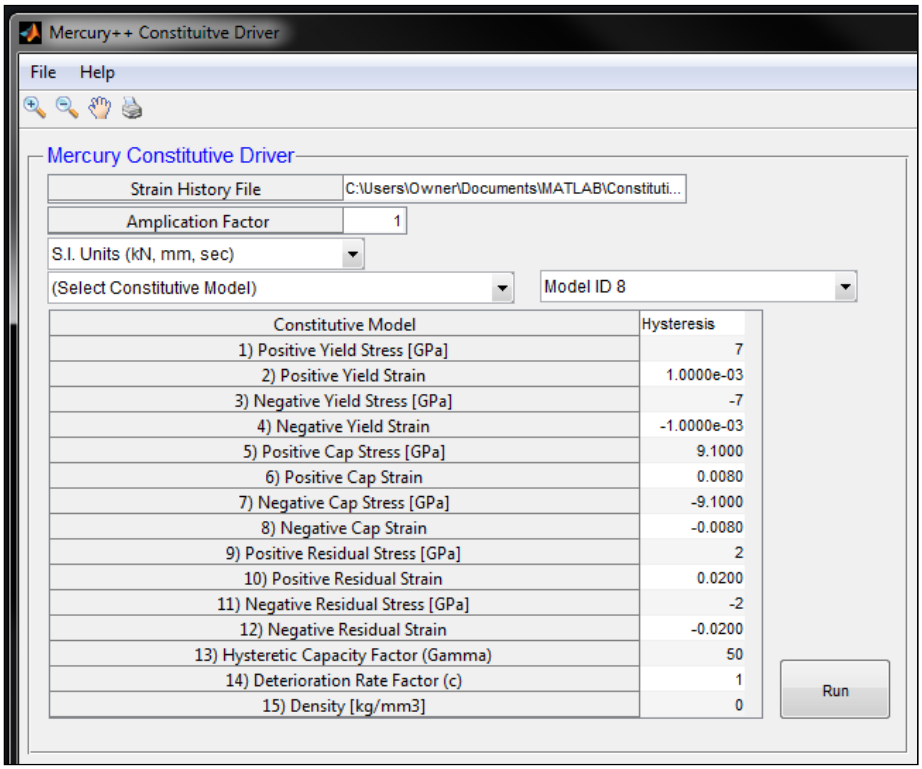

Figure 5.12 Existing Model Data

5.3.3 Constitutive Driver Results

After a strain history file has been loaded, and material parameters have been entered for a constitutive model, the program may be run by pressing the “Run” pushbutton in the user input region. The user is prompted to select a file save path to store the output and the stress-strain results are plotted in the output region.

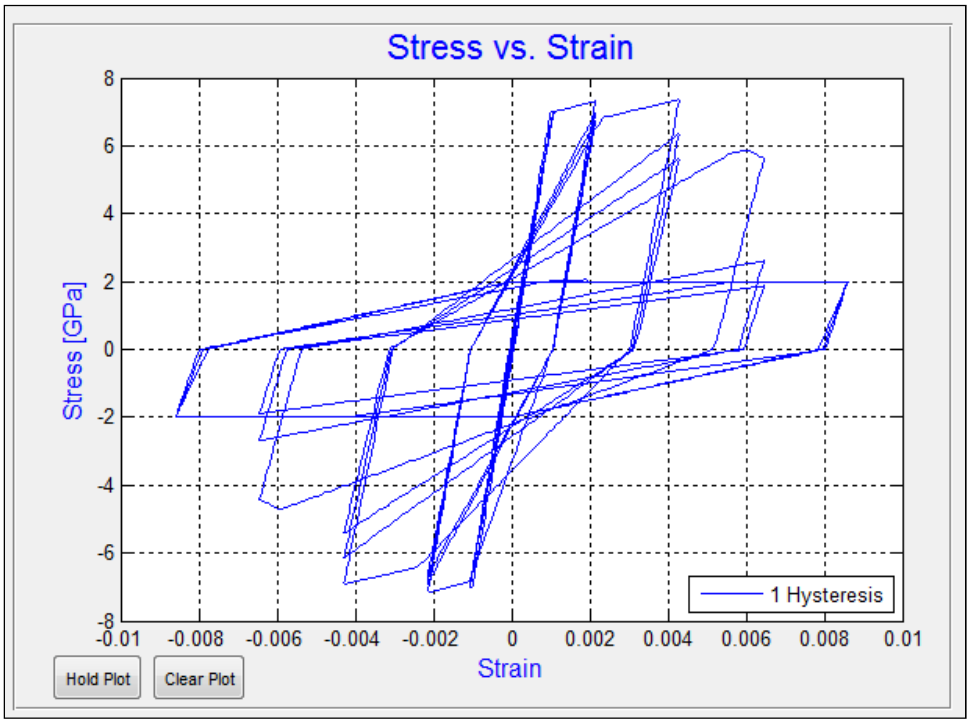

Figure 5.13 Model Run Output

To compare the output to another model, press the "Hold Plot" pushbutton and input the information to compare. When the model is run the output is displayed over the previous plot. For example, if a Bilinear model with similar material parameters is compared, the following output is displayed.

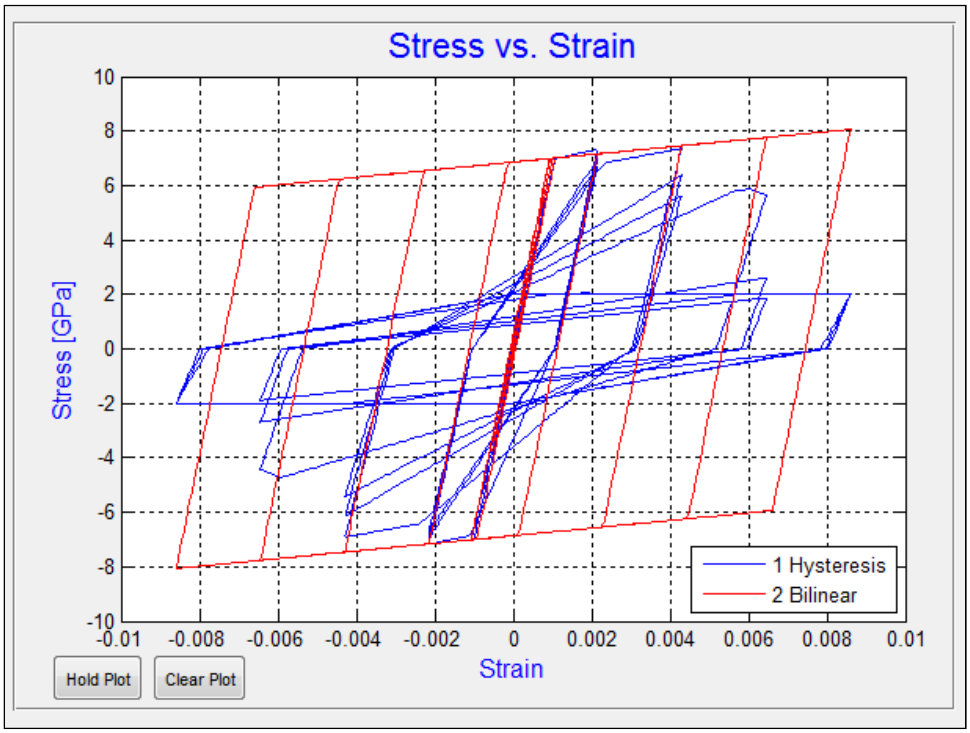

Figure 5.14 Comparing Driver Models

The Constitutive Driver provides special treatment of the Modified Kent & Park model for reinforced concrete. If the Modified Kent & Park model is selected from the constitutive model selection popup menu, a region appears in the bottom right-hand corner to receive cross-section information to populate model data in the user input region. Individual model parameters are determined for concrete confined by transverse stirrups and unconfined cover concrete depending on whether or not the check box labeled “Confined Concrete Properties” is selected.

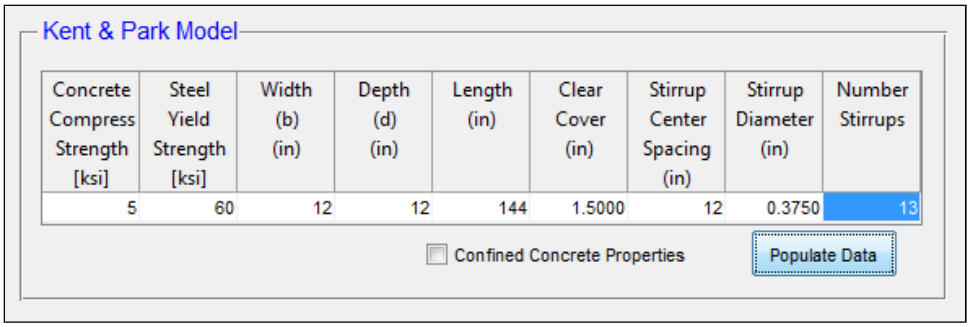

Figure 5.15 Modified Kent & Park Model Section Input

Once relevant information is input, the user may populate the data in the user input table by pressing the “Populate Data” pushbutton.

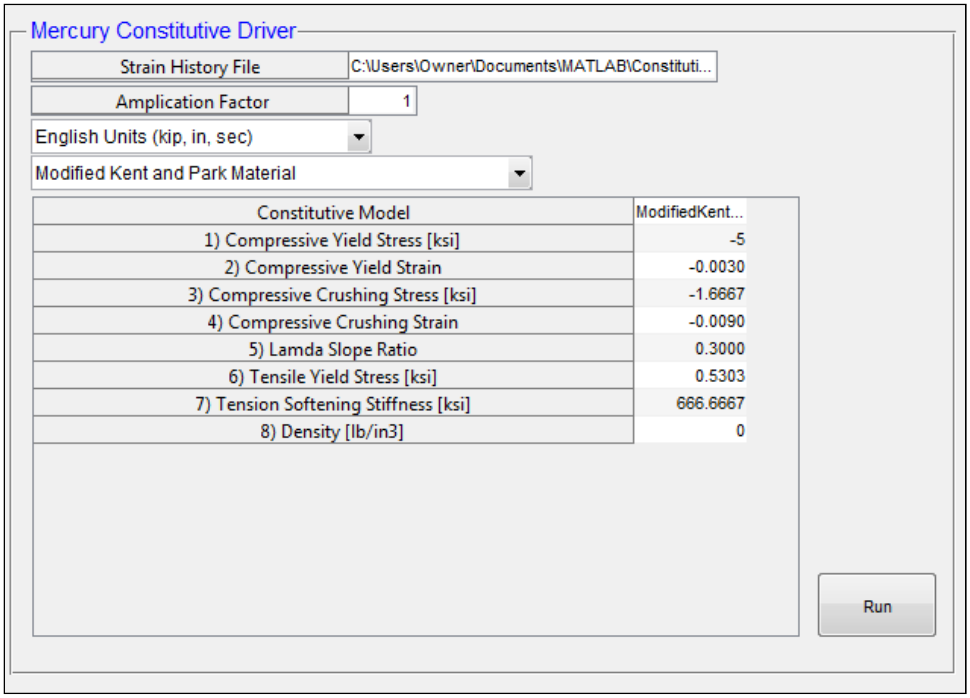

Figure 5.16 Modified Kent & Park Automated Values

5.4 Chapter Summary

This section details the development of a constitutive driver tool used for constitutive model development in Mercury and as a learning tool. This constitutive driver has been included in a Mercury pre and post-processor application described in Chapter 6.

6 Mercury++

6.1 Introduction

The second visualization tool developed as part of this thesis is a pre and post-processor tool for inputting model information and receiving analysis output from Mercury. Similar to the Constitutive Driver, Mercury++ has been developed using MATLAB's GUIDE tool for developing GUIs and the Constitutive Driver has been included in Mercury++. This section first presents the motivation for developing the Mercury++ pre and post-processor applications. Next, information is provided for designing and coding a GUI using MATLAB's GUIDE tool. The Mercury++ environment is then explained using a simple lumped plasticity model. The pre-processor is presented first to explain the process of building a Mercury input file in Mercury++. A Mercury analysis is performed for the lumped plasticity model and the post-processor is explained using the results of the analysis. Finally, menu functionality (i.e. printing, saving) is discussed.

6.2 Motivation for Development of Mercury Pre and Post-Processors

The MATLAB version of Mercury requires users to have a good understanding of MATLAB formatting and programming. Furthermore, an understanding of the exact formatting used for Mercury is necessary to build a model input file. Although learning basic MATLAB programming techniques is rather straight forward, it may deter some users from the benefits of using Mercury as a computational tool when formatting issues result in excessive pre-processing time to build model input files. Even more, basic formatting mistakes may cause an input file to relay unintended information to the Mercury platform. Users are easily frustrated when simple formatting issues cause model errors and little or no visual validation is available prior to running an analysis. A user friendly environment which provides a

step-by-step input model development process can promote the use of Mercury and allow users to quickly become familiar with its capabilities.

Acquiring all relevant results of a numerical simulation is vital to validating a model. Obtaining the results of a Mercury analysis requires users to call out the desired results in the Mercury input file. Mercury analysis output is stored either as a data array (in a .mat format) or as an ASCII file. In order to view model output, it is necessary for users to load the output in a spreadsheet or assign the output as a variable in MATLAB, and combine information to plot. When plotting multiple outputs or when plotting information pertaining to separate nodes or elements (i.e. base shear vs. column lateral drift), this process may require a good amount of post-process time. In addition, a user may run a model without calling for all necessary output information, requiring analysis to be performed again. A program capable of obtaining all possible analysis output and providing users with instant access to results may provide substantial time savings, provide a better understanding (to Mercury users) of the program's capabilities, and promote use of the program.

6.3 Building a GUI in MATLAB

Mercury++ and the Constitutive Driver were each developed with the assistance of MATLAB's GUIDE program for creating a GUI skeleton. The GUIDE Quick Start menu may be accessed in MATLAB from the File>New>GUI dropdown menu or by typing "guide" in the MATLAB command prompt. By selecting the option to design a blank GUI, the user is provided a blank, untitled figure with several control options to select from on the left panel of the figure.

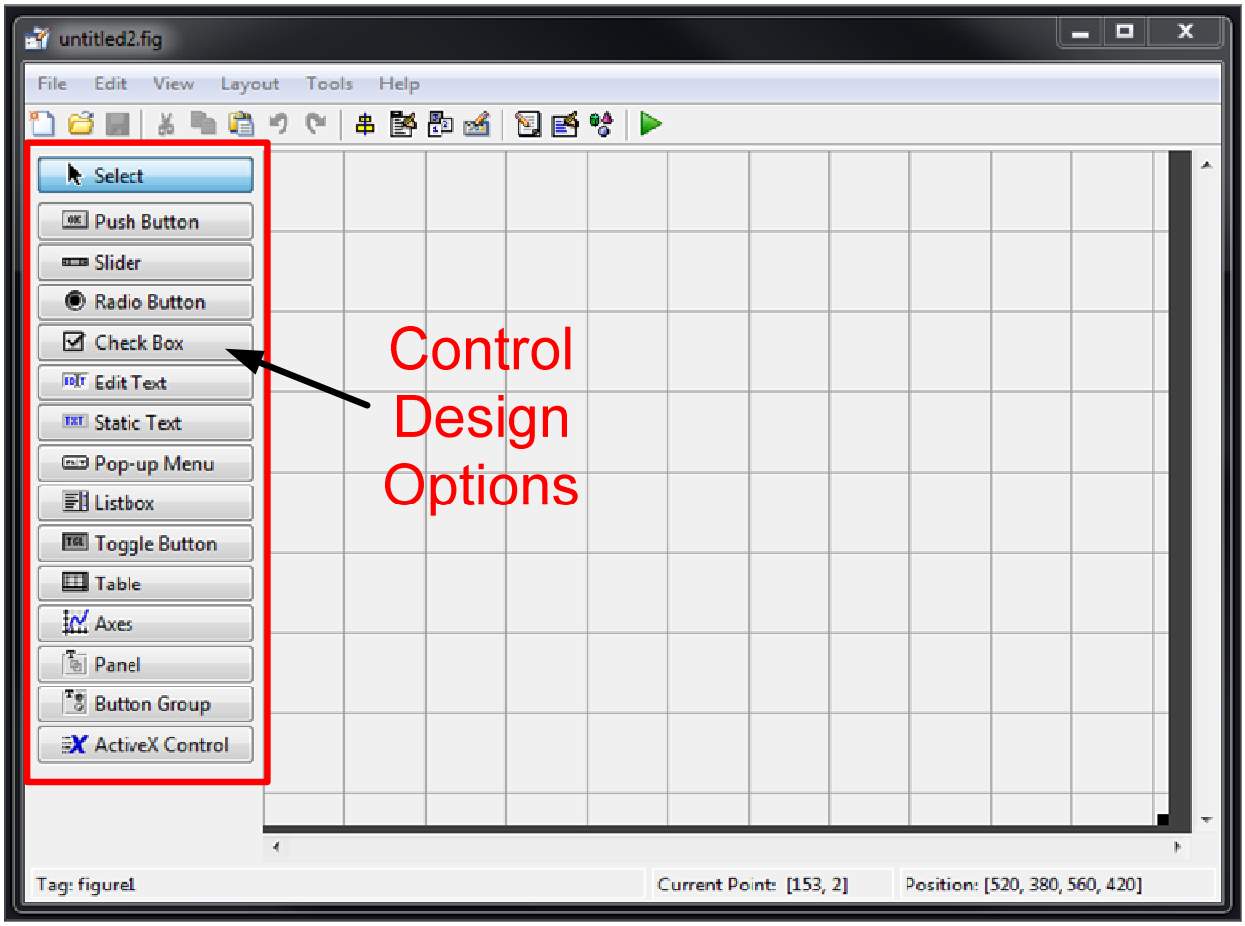

Figure 6.1 MATLAB Blank GUI

When the figure is saved for the first time, MATLAB assembles the framework coding for the GUI in object oriented format and saves the GUI figure (.fig file). MATLAB's object oriented format relies on a series of MATLAB functions that operate using the *handles* structure for passing information, and the *varargin* and *varargout* structures for receiving and returning input and output variables respectively. The framework assembled by MATLAB includes an initialization function, an opening function, and an output function. For a new GUI figure saved under the filename "NewGUI", the following functions are assembled and the figure is saves as NewGUI.fig.

No.	Function Name	Description
1	function varargout = NewGUI(varargin)	Initializes GUI State Identification to Signal Response Upon Opening of GUI, Callback of a Control, & Closing of the GUI
2	function NewGUI_OpeningFcn	Sets Control Properties When GUI is Opened
3	function varargout = NewGUI_OutputFcn	Handles Data Acquired From GUI Process

Table 6-1 MATLAB GUI Framework Functions

```
function varargout = NewGUI(varargin)
 gui_Singleton = 1;
 gui_State = struct('gui_Name',       mfilename, ...
                    'gui_Singleton',  gui_Singleton, ...
                    'gui_OpeningFcn', @NewGUI_OpeningFcn, ...
                    'gui_OutputFcn',  @NewGUI_OutputFcn, ...
                    'gui_LayoutFcn',  [] , ...
                    'gui_Callback',   []);
      if nargin && ischar(varargin{1})
            gui_State.gui_Callback = str2func(varargin{1});
      end
      if nargout
            [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
      else
            gui_mainfcn(gui_State, varargin{:});
      end

function NewGUI_OpeningFcn(hObject, eventdata, handles, varargin)

      handles.output = hObject;

guidata(hObject, handles);

function varargout = NewGUI_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;
```

Figure 6.2 NewGUI Framework Code

Controls placed on the figure are assigned properties and callbacks by the GUI developer to produce a functional interface. In order to be used as an object within the GUI figure, a control must be assigned a function to create and initialize properties and a function to handle callback processes. If, for instance, a popup menu and a pushbutton are placed on NewGUI.fig, the user must assign functions to create the controls and handle callbacks.

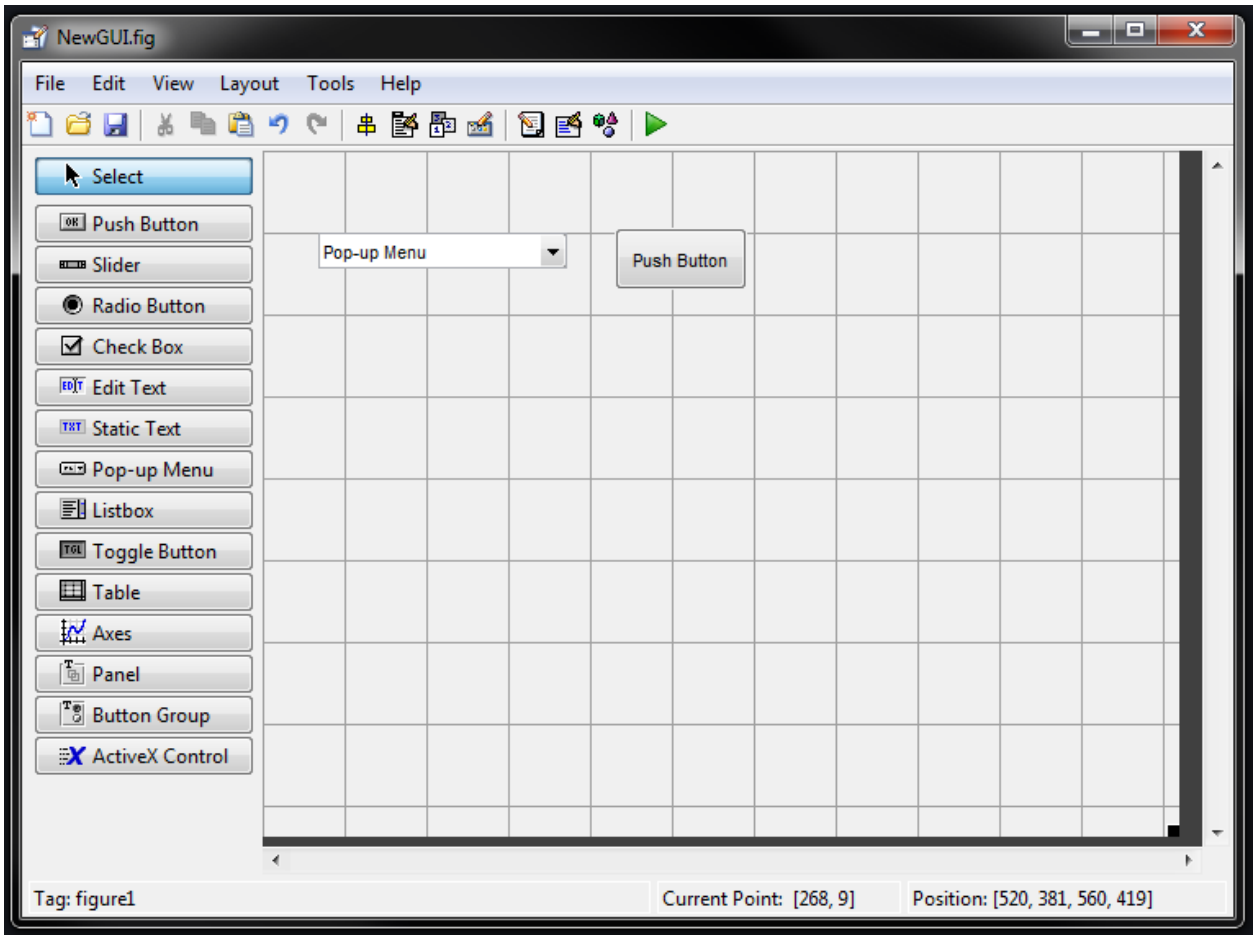

Figure 6.3 Adding Controls to GUI Figure

No.	Function Name	Description
1	function pushbutton1_Callback	Handles User Defined Events When pushbutton1 Callback is Signaled
2	function popupmenu1_Callback	Handles User Defined Events When popupmenu1 Callback is Signaled
3	function popupmenu1_CreateFcn	After Properties Have Been Set, Handles Creation of popupmenu1
4	function pushbutton1_CreateFcn	After Properties Have Been Set, Handles Creation of pushbutton1

Figure 6.4 Create and Callback Functions

```
function pushbutton1_Callback(hObject, eventdata, handles)

guidata(hObject, handles);

function popupmenu1_Callback(hObject, eventdata, handles)

guidata(hObject, handles);

function popupmenu1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pushbutton1_CreateFcn(hObject, eventdata, handles)

guidata(hObject, handles);
```

Figure 6.5 Control Create and Callback Function Code

Development of the GUI consists of setting control properties in the GUI opening function and control callback functions, and coding event handling in the control callback functions.

A simple GUI could involve displaying a message box with a phrase selected from the popup menu when the pushbutton is pressed. For example, the GUI user may have the option of displaying either of the following phrases:

1. "hello world"
2. "Go Buffs"
3. "Happy New Year"

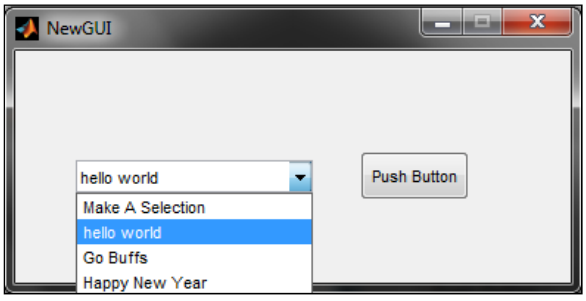

Figure 6.6 Example GUI Popup Menu Choices

To do this, the developer sets the *String* property of the popup menu in the GUI opening function to provide the three phrase choices (Figure 6.7). Then, event handling is coded in the pushbutton callback to obtain the selected phrase from the popup menu and to display the phrase in a message box. For this particular example, when the pushbutton is pressed, event handling is coded to:

1. Obtain the popup menu *Value* property corresponding to the item number selected
2. Do nothing if a selection is not made from the popup menu
3. If a selection is made in the popup menu, display the phrase associated with the selection


```
function varargout = NewGUI(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @NewGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @NewGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function NewGUI_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
set(handles.popupmenu1, 'String', [{'Make A Selection'} {'hello world'} ...
    {'Go Buffs'} {'Happy New Year'}])
handles.phrase = get(handles.popupmenu1, 'String');
guidata(hObject, handles);

function varargout = NewGUI_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)
switch get(handles.popupmenu1, 'Value');
    case 1
        % do nothing
    otherwise
        msgbox(handles.phrase{get(handles.popupmenu1, 'Value')})
end
guidata(hObject, handles);

function popupmenu1_Callback(hObject, eventdata, handles)

guidata(hObject, handles);

function popupmenu1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Figure 6.7 GUI Development Example Code

Figure 6.8 Functioning Example GUI

6.4 Mercury++ Environment

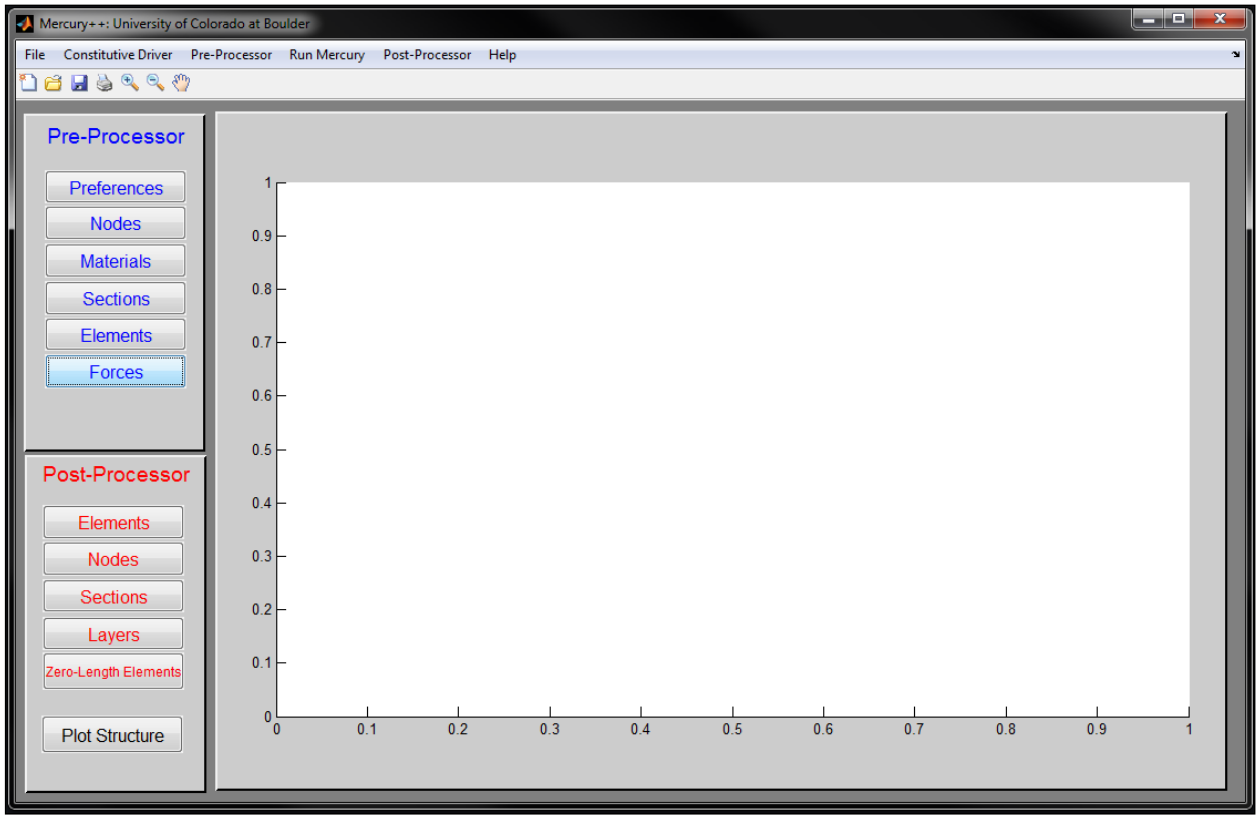

Figure 6.9 Mercury++ Front Panel

The Mercury++ environment is a collection of 17 GUIs separated between pre-process model development, Mercury analysis, and post-process data acquisition. The pre-processor, Mercury analysis application, and post-processor are assigned individual dropdown menus. In addition, there is a File dropdown menu with the functionality of most software applications, a dropdown menu to access the

Constitutive Driver, and a Help dropdown menu to access the Mercury User's Manual. The Mercury++ Menu architecture is summarized in Figure 6.10.

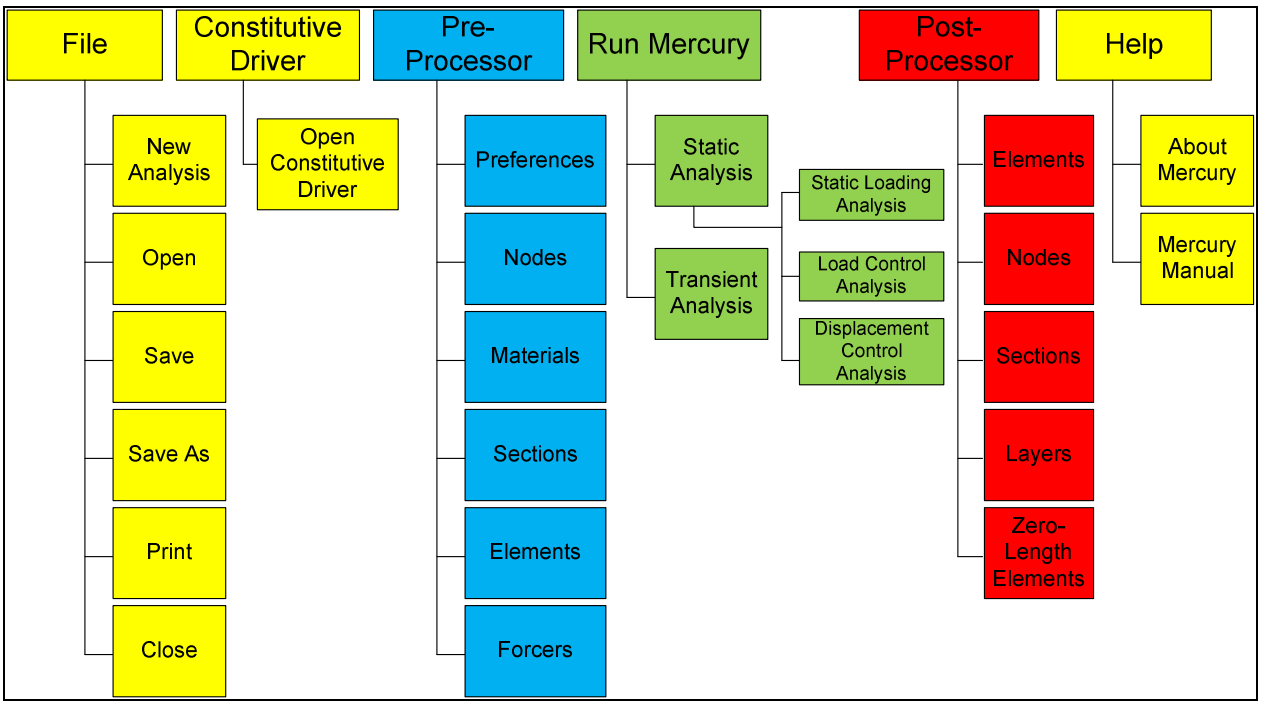

Figure 6.10 Mercury++ Menu Architecture

The Mercury++ front panel is primarily composed of an axis for plotting the model and results. Below the dropdown menus there is a toolbar with shortcuts for starting a new analysis, opening a file, saving, and printing. There are also zoom and pan tools for viewing results. Two panels are located next to the plot axis with links to the pre-process and post-process menus. Mercury++ pre-process applications are signified by blue coloring in the Mercury++ environment (see Figure 6.11) and post-process applications are signified by red coloring.

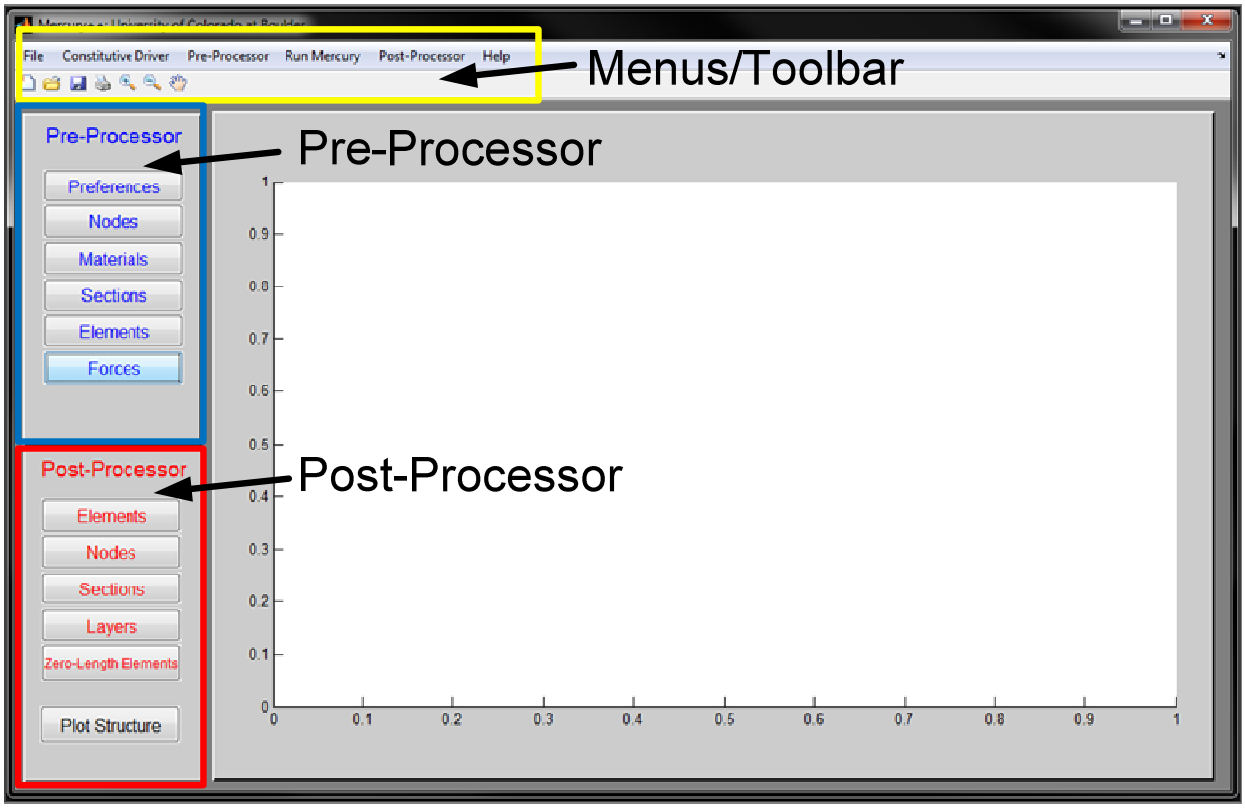

Figure 6.11 Front Panel Regions

6.4.1 Pre-Processor

For illustration of the pre-process capabilities of Mercury++, a cantilever column, with a zero-length rotational spring at its base, is modeled. The material and cross-sectional properties are summarized in Figure 6.12.

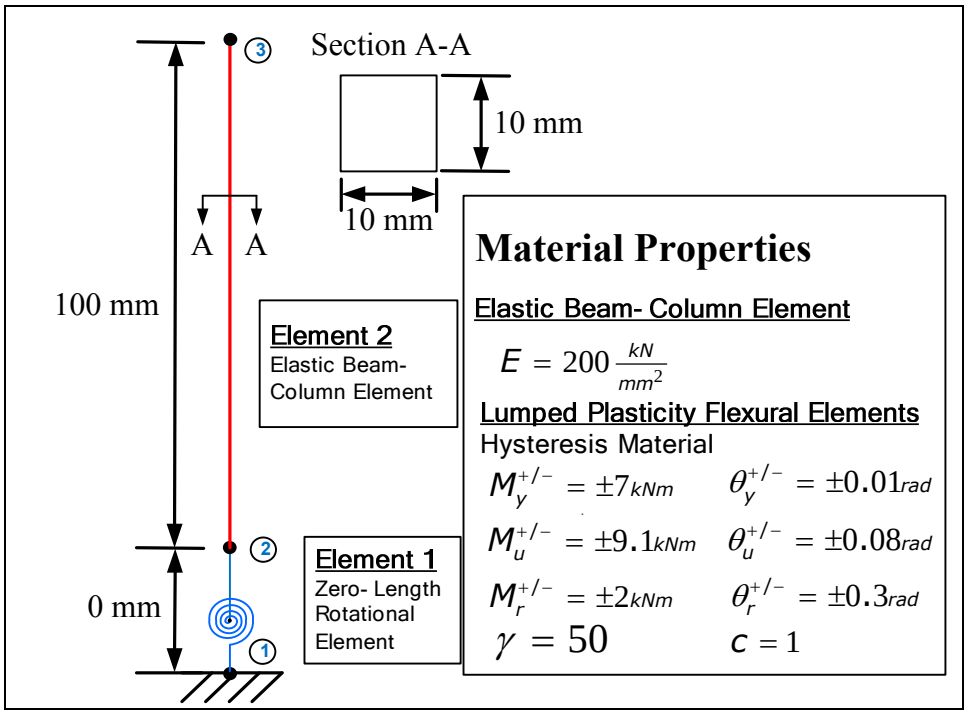

Figure 6.12 Mercury++ Example Model

Mercury++ input file assembly uses the Pre-Processor menu options which can be accessed from the dropdown menu and from the blue Pre-Processor panel.

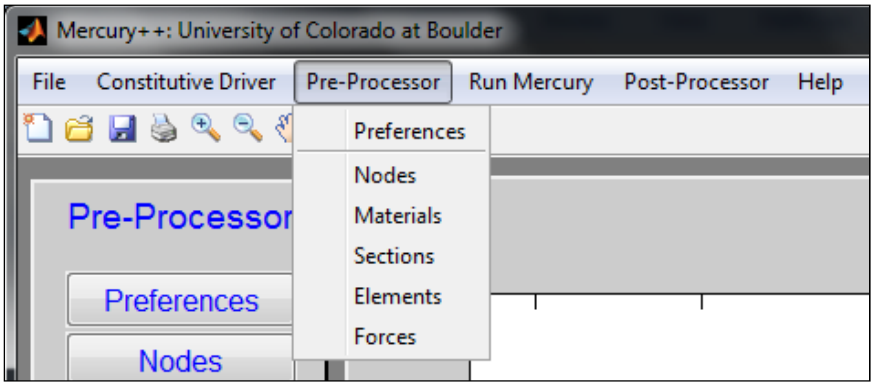

Figure 6.13 Pre-Processor Dropdown Menu

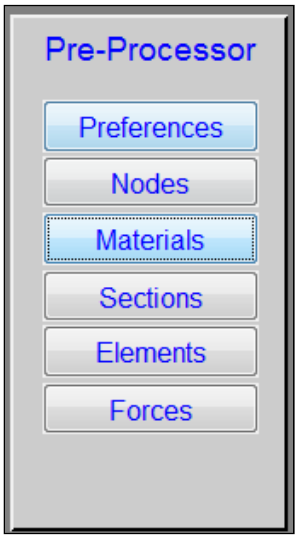

Figure 6.14 Pre-Processor Panel

When assembling a Mercury input file using the Mercury++ Pre-Processor, the model design must follow a certain process. Information entered in each of the Pre-Processor menu items is used for reference and model design in each of the subsequent Pre-Processor menu items.

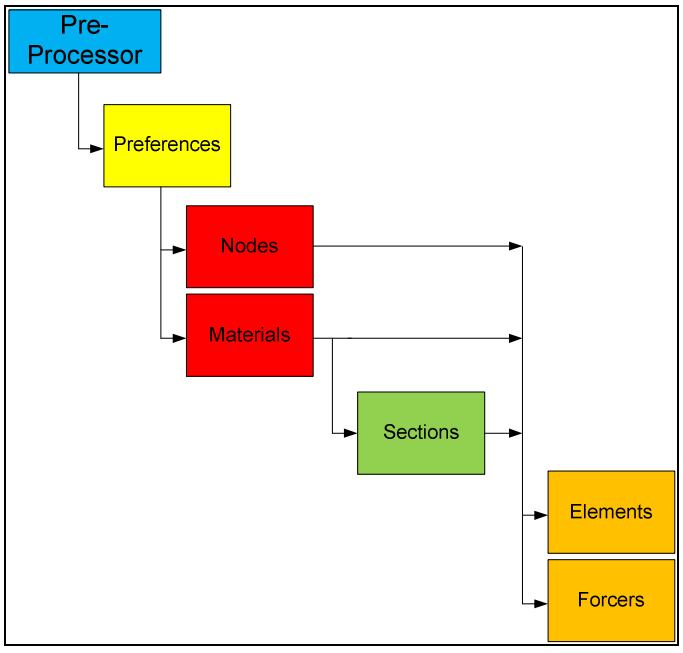

Figure 6.15 Mecury++ Pre-Processor Flow Chart

This process is organized with the intention that Mercury++ users are easily guided through the pre-process. Messages appear when users attempt to access Pre-Processor menu items out of order.

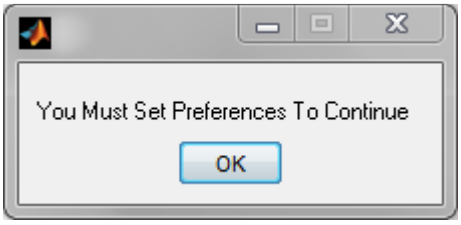

Figure 6.16 Pre-Processor Flow Message

6.4.1.1 *Preferences Editor*

To begin building a Mercury input file, preferences must be set in the Preferences Editor GUI (Figure 6.17). Preferences include units, structure dimension, and structure type (number of degrees of freedom per node). The Preferences Editor GUI assembles the Mercury input variables *unit* and *strmode*.

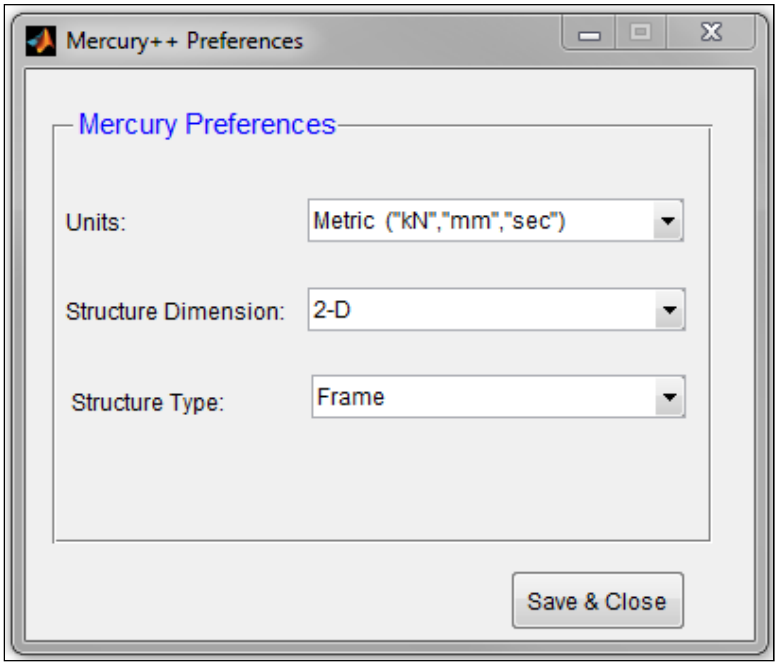

Figure 6.17 Preferences GUI

6.4.1.2 Node Editor

Nodal coordinates may be defined, and constraints applied, in the Node Editor GUI. Nodal coordinates and constraints may be modified at any time during the development process by making changes to the editor table. The Node Editor GUI assembles the Mercury input variables *nodcoord* and *constraint*.

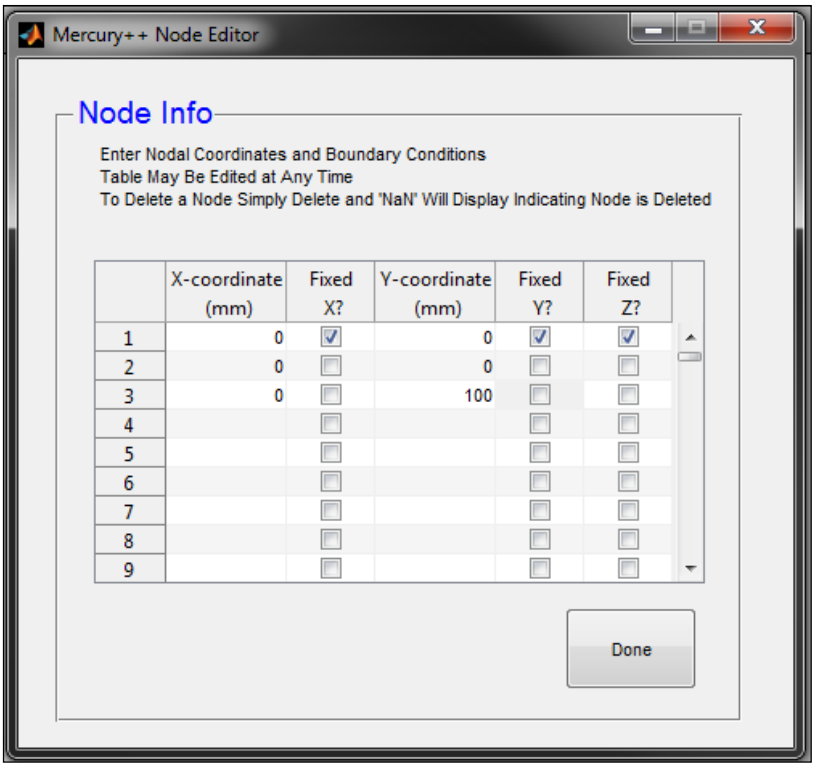

Figure 6.18 Nodes Editor GUI

6.4.1.3 Material Editor

Material properties are assigned using the Material Editor GUI. New materials are created in the "Create New Materials" panel and added to the "Project Materials" panel to display active project materials. The "Project Materials" table is editable and materials can be deleted by selecting the desired material from the "Delete Materials" panel. The Material Editor assembles the Mercury input variable *materials*. The Constitutive Driver may be accessed from the Materials Editor GUI by using the "Constitutive Driver" pushbutton at the bottom.

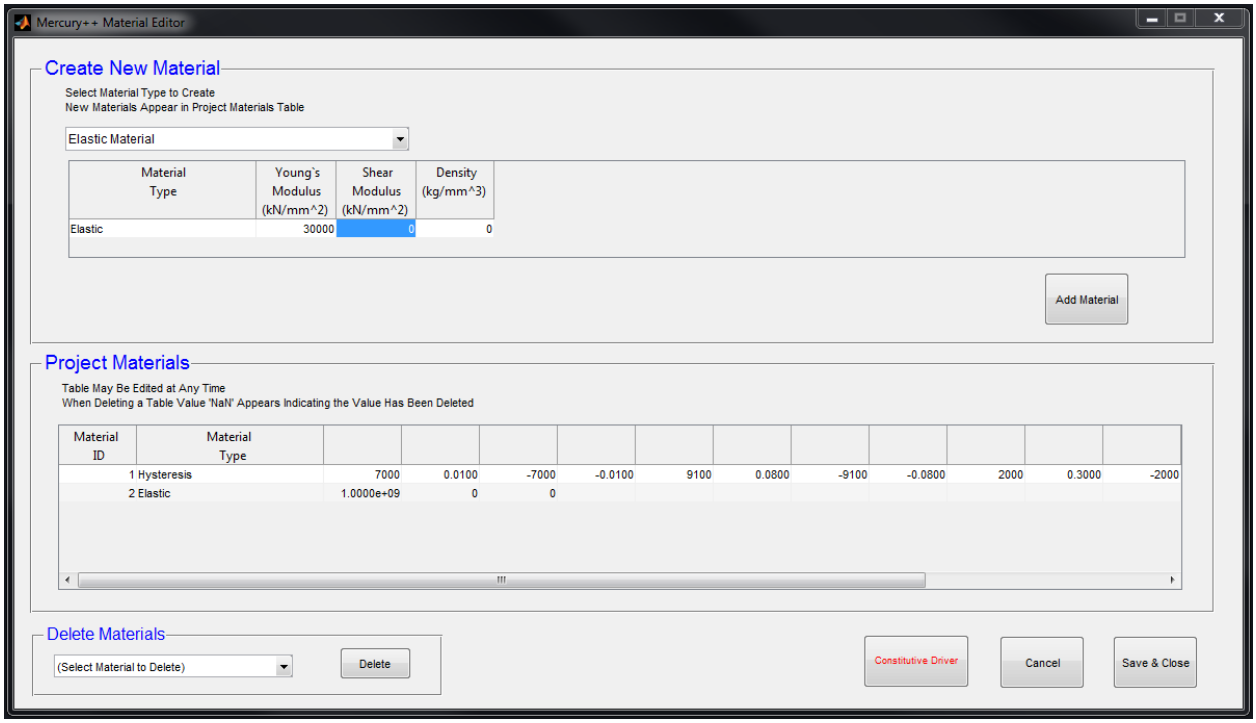

Figure 6.19 Materials Editor GUI

6.4.1.4 *Sections Editor*

Section properties are defined in the Sections Editor GUI. Mercury has the capability of assigning a "General2D" section, for elastic elements, or "Layer" and "Fiber" sections for distributed plasticity elements. The user selects the desired section type from the "Create New Section" panel. New sections are added to the "Projects Sections" panel containing all current project sections. Material properties assigned in the Material Editor GUI are loaded in the "Project Materials" panel for user reference. The Section Editor GUI assembles the Mercury input variable *sections*.

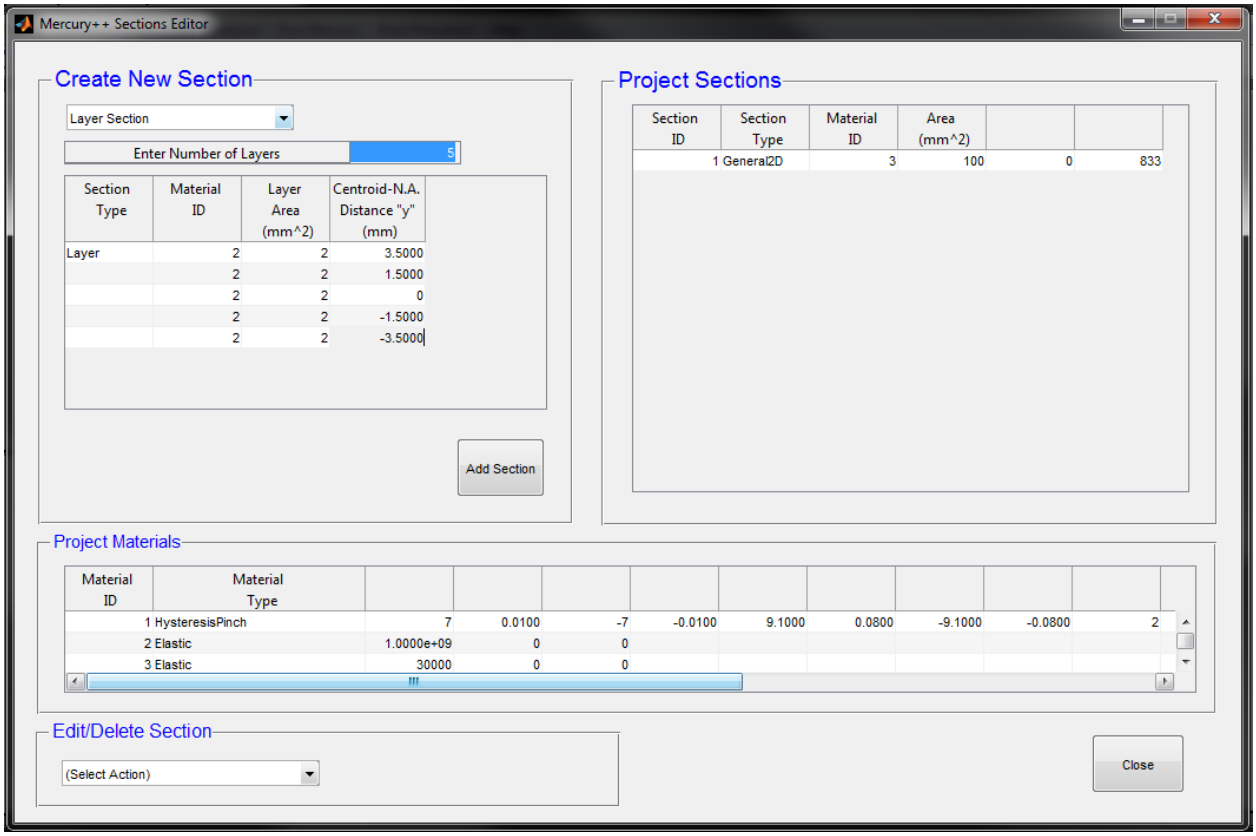

Figure 6.20 Sections Editor GUI

Sections may be edited or deleted using the menus in the "Edit/Delete Section" panel. If section editing is desired, the user is directed to a separate GUI to complete editing. Information is displayed for the section for which editing is desired only.

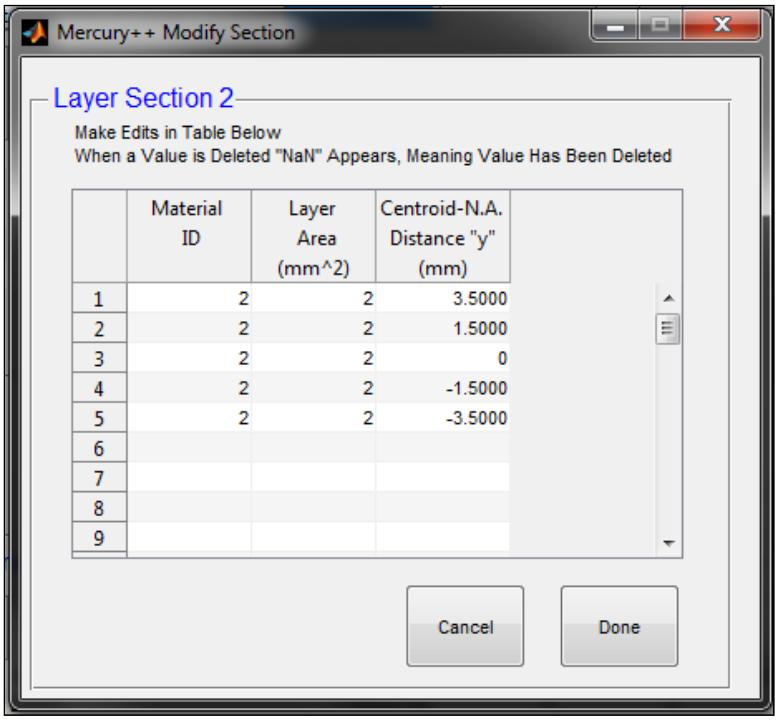

Figure 6.21 Section Editing GUI

6.4.1.5 Element Editor

Once nodes, materials and sections have been defined, elements may be assembled in the Element Editor GUI. For user reference, nodal coordinates, nodal constraints, project materials, and projects sections are displayed in separate panels. New elements are added using the "Create New Element" panel and displayed in the "Elements" panel, containing all project elements, when the element is created. The Element Editor GUI assembles the Mercury input variable *elements*.

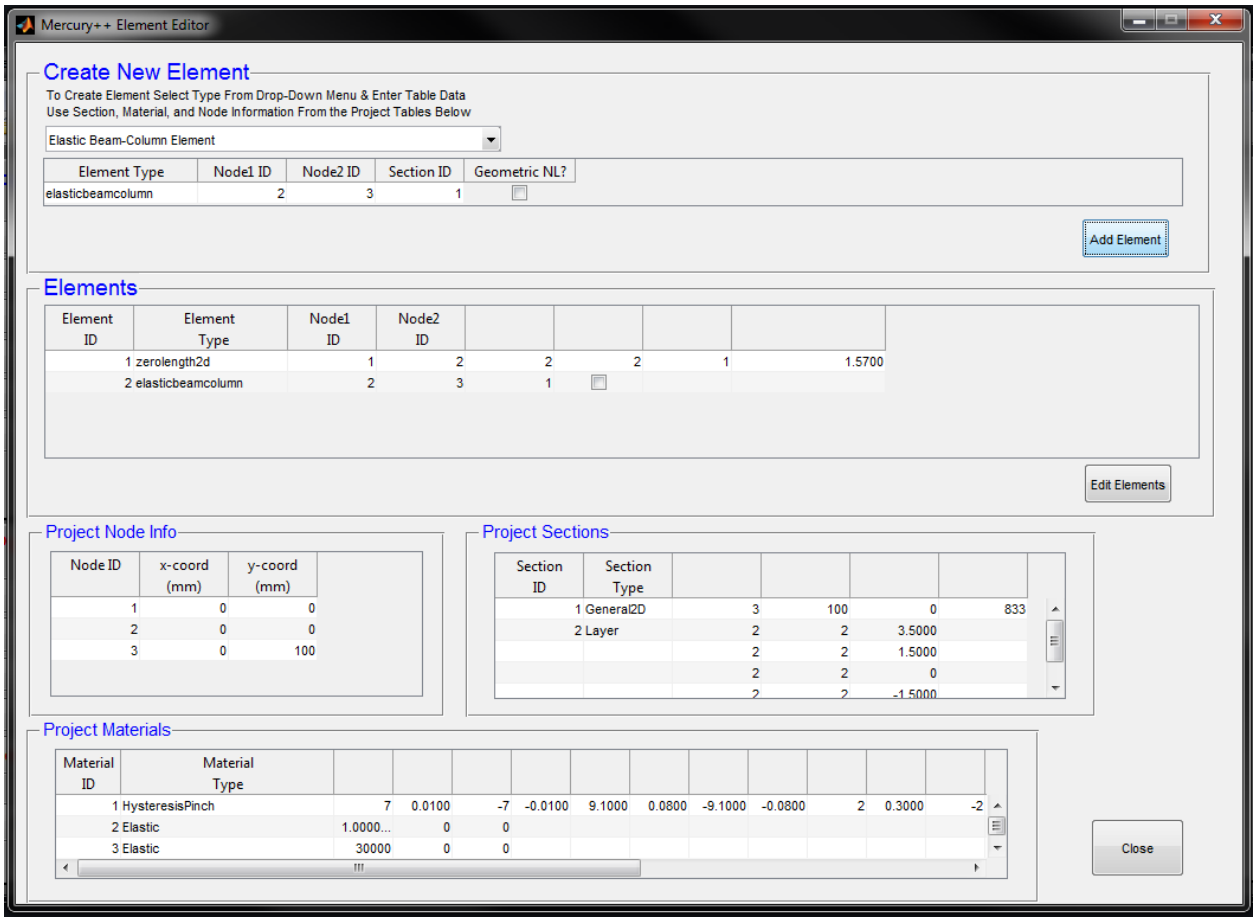

Figure 6.22 Elements GUI

Elements are edited by pressing the “Edit Elements” pushbutton to access a separate GUI fro editing and deleting.

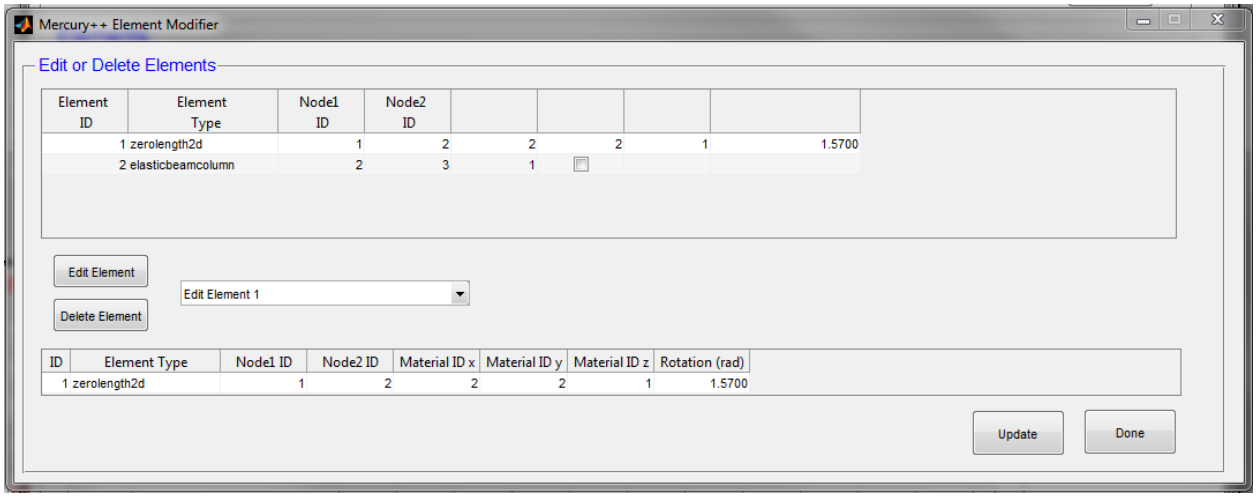

Figure 6.23 Element Editing GUI

6.4.1.6 *Forces Editor*

Forces are applied to the structure using the Forces Editor GUI. Mercury is capable of performing analyses for single static forces, incremental static forces, incremental static displacements, and dynamic forces. Mercury++ users select static or dynamic force type from the "Force Type" panel to enter force values in the "Apply New Force" panel. Nodal coordinates and element information is provided for user reference. Applied forces appear in the "Forces" panel and separate force types may be viewed individually by selecting the corresponding panel identifier. The Forces Editor GUI assembles the Mercury input variable *forces*.

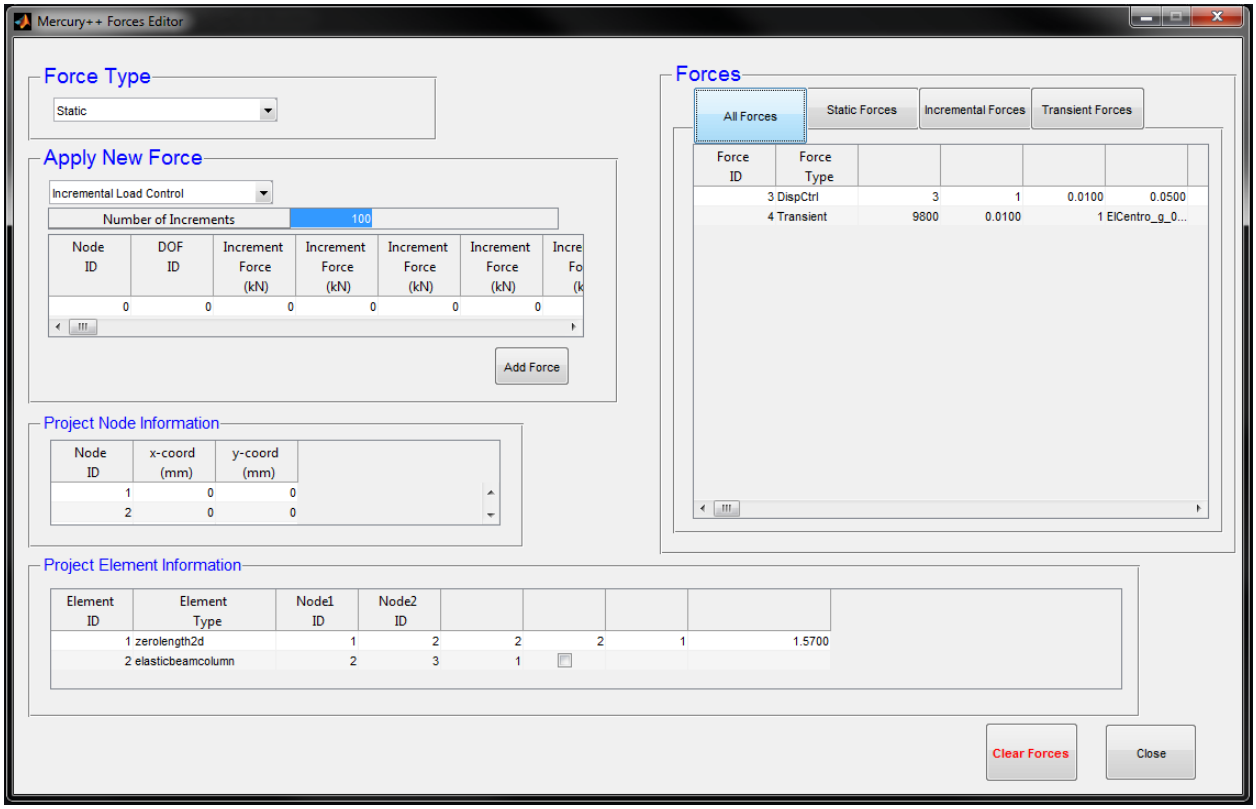

Figure 6.24 Forces Editor GUI

6.4.1.7 *Model Plotting in Mercury++*

As the Mercury input file is assembled, updates automatically appear visually on the Mercury++ front panel. Nodes are numbered according to the Node Editor GUI inputs, constraints are labeled, and elements are numbered according the Element Editor GUI. Zero-length elements appear as large asterisk symbols. At any time during the pre and post processes, the model may be plotted on the Mercury++front panel by pressing the "Plot Structure" pushbutton located beneath the red Post-Processor panel menu items.

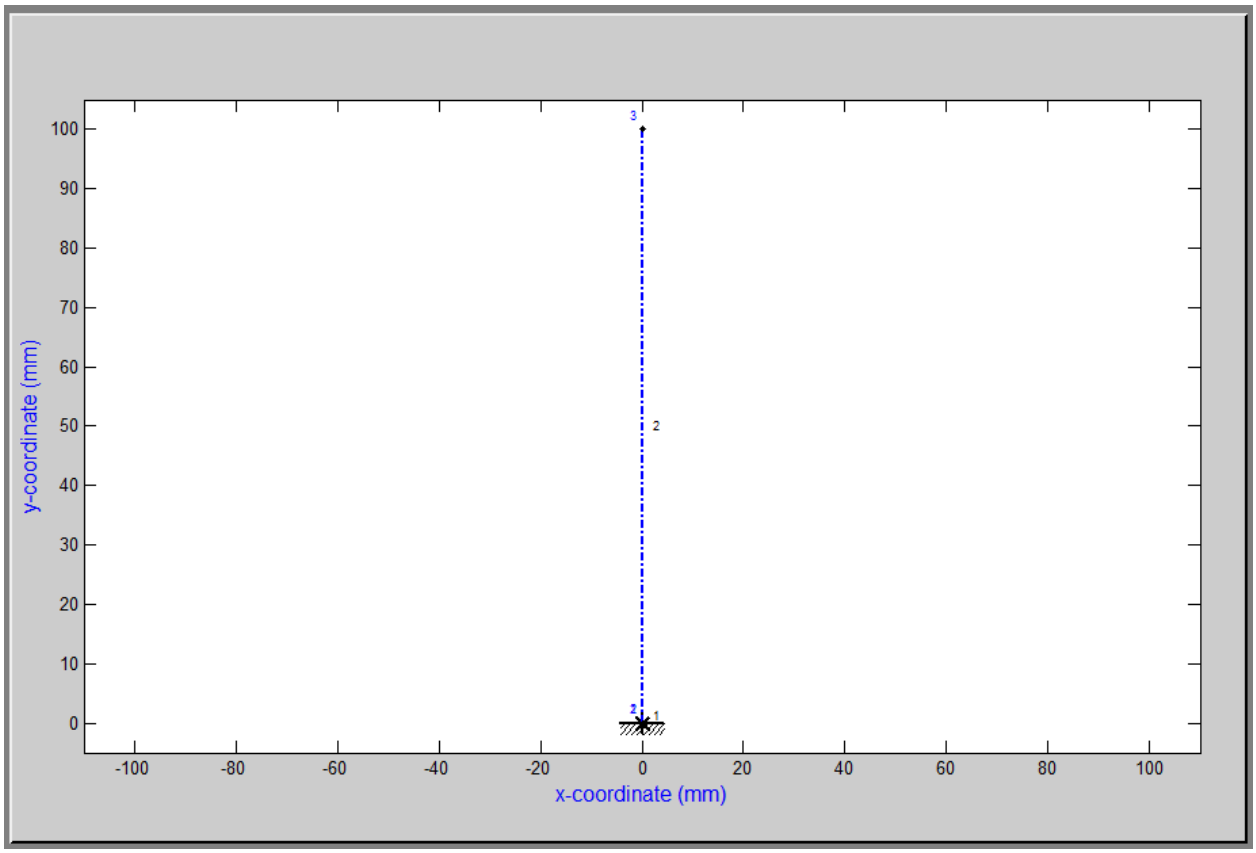

Figure 6.25 Mercury++ Model Verification Plot

6.4.2 Complex Modeling Advantages Using Mercury++

Mercury++ has been carefully coded to allow users to work with a spreadsheet format to assemble complex models and input lengthy force information. Users have the opportunity to prepare input data in an Microsoft Excel spreadsheet and copy the information into Mercury++. The Mercury input file is accessible within MATLAB because Mercury++ stores input file variables in the MATLAB workspace. Once Mercury++ has been opened, each of the Mercury input variables identified in Chapter 6.4.1 is loaded and formatted in the MATLAB workspace.

As an example, if a user wishes to apply the incremental static displacements shown in Figure 6.26, to the example cantilever column, inputting the information in the Forces Editor GUI may be inefficient if the displacement history is lengthy.

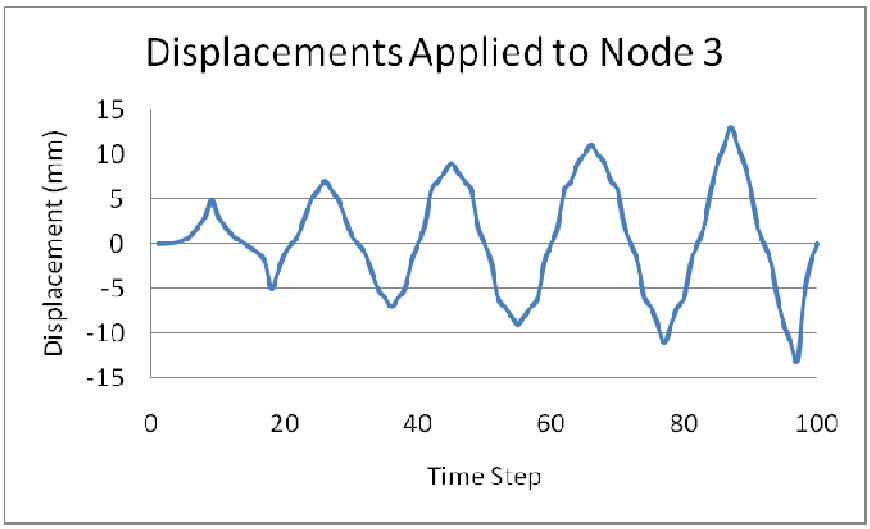

Figure 6.26 Incremental Displacements Example Model Node 3

Instead, if the displacements are created in a spreadsheet, they can be loaded in the MATLAB workspace and are available in Mercury++. The user simply creates the force type in the Forces Editor GUI without inputting all of the displacements. In this case, there are a total of 100 increments, applied as static incremental displacements at node 3 in the horizontal direction corresponding to nodal degree of freedom 1.

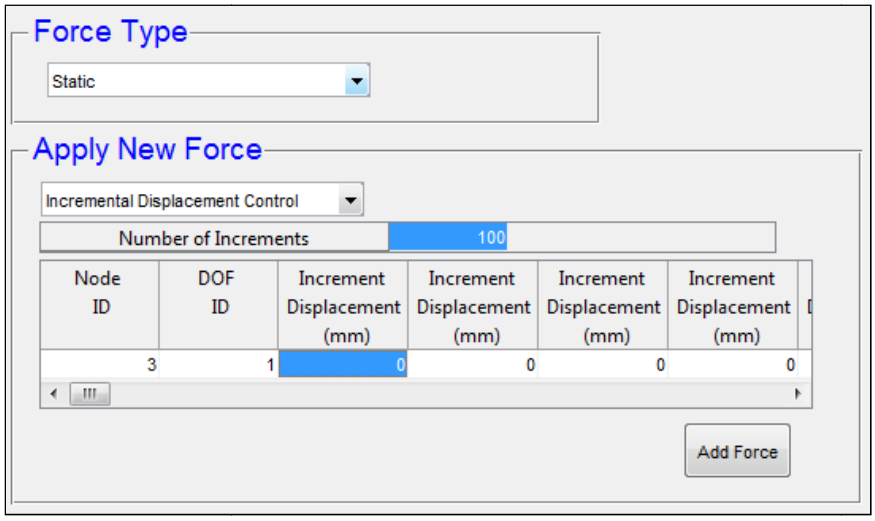

Figure 6.27 Allocating Space for Spreadsheet Values

Notice that the displacements are automatically assigned as zeros. The user only inputs the node ID and degree of freedom ID and adds the force. The zero displacement history can then be accessed in the MATLAB workspace by opening the variable *forces*.

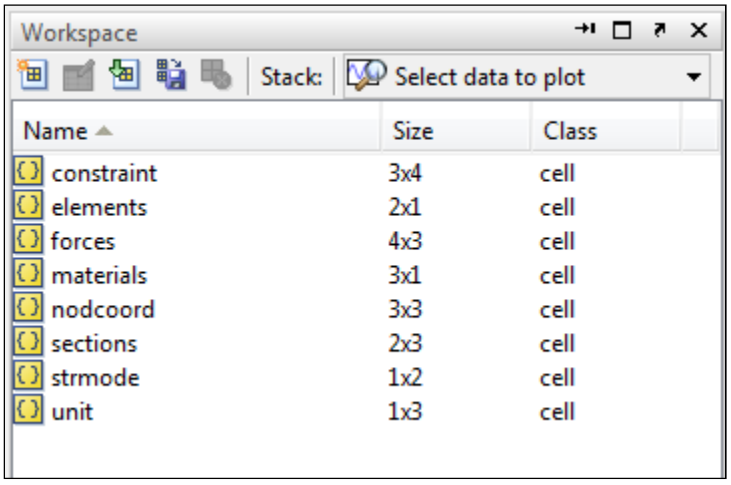

Figure 6.28 MATLAB Workspace Variables

When *forces* is accessed in the MATLAB workspace, the user is presented with all possible force types.

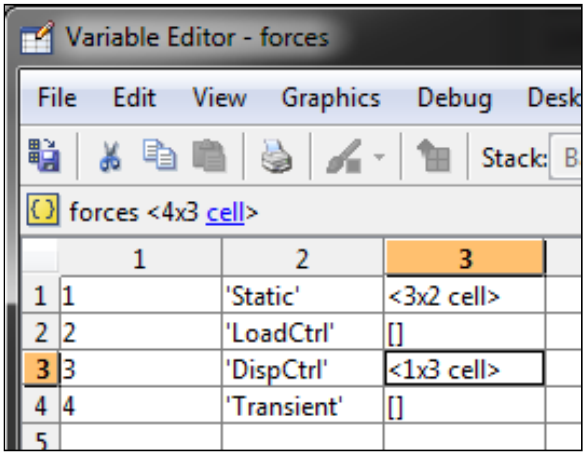

Figure 6.29 Opening MATLAB Spreadsheet to Load Displacements

Opening the displacement control portion of the *forces* variable (cell {3,3}), and opening the allocated 1x100 array (Figure 6.30), the allocated zero displacements are seen (Figure 6.31).

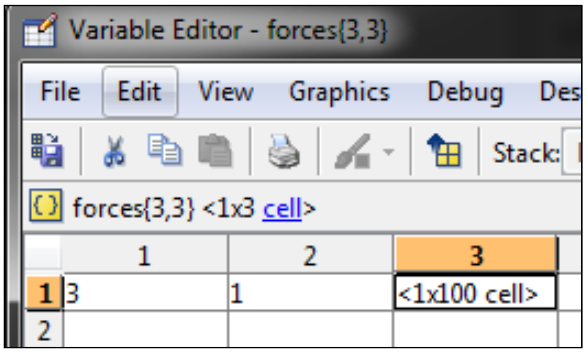

Figure 6.30 Allocated Array

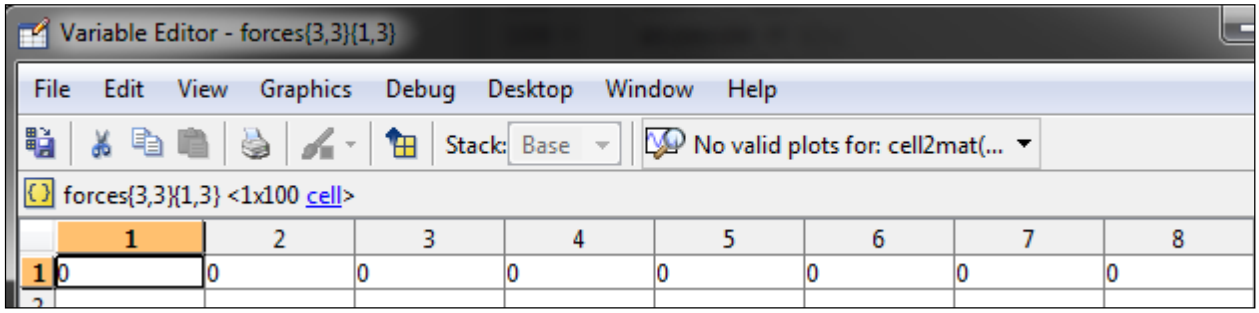

Figure 6.31 Allocated Cells

By pasting displacement values in the spreadsheet, the displacements are loaded, and the displacement controlled analysis may be run.

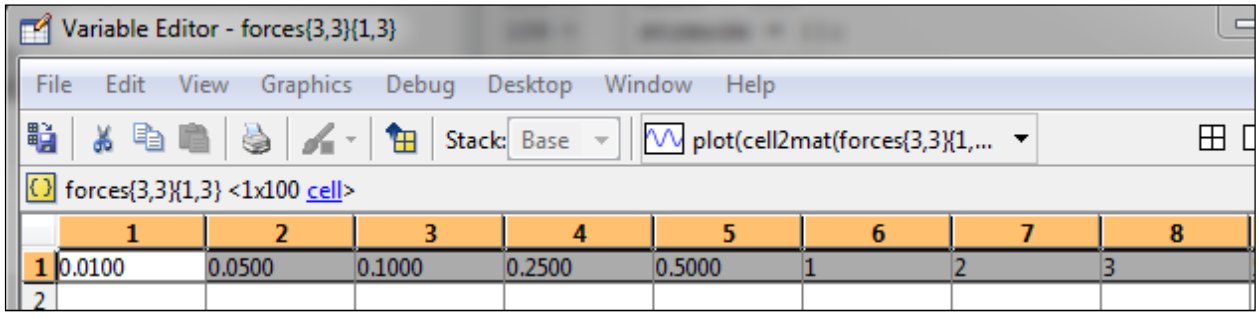

Figure 6.32 Transferring Spreadsheet Data

The Mercury++ pre-processor provides a series of individual, editable menus which eliminate the need for users to understand Mercury input formatting. Just as important, the input model is plotted for user visualization, creating a much simpler input verification process and easier error identification. Mercury++ provides additional benefits when users desire to prepare model data in a spreadsheet (i.e.

Excel format). Because the Mercury++ Pre-Processor assembles data arrays for each of the Mercury input variables, transferring data from a spreadsheet is relatively straight forward.

6.4.3 Running Mercury

Mercury analyses are run by selecting the desired analysis type from the "Run Mercury" dropdown menu.

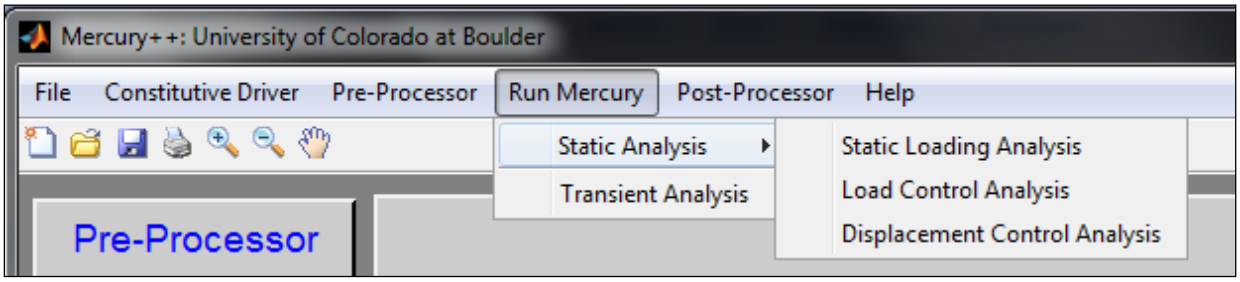

Figure 6.33 Mercury Dropdown Menu

Analyses are divided between static and transient. Static loading and incremental static loading follow the same process, in Mercury++, while transient loading directs the user to input the analysis integration scheme, iterative methods, damping, etc.

6.4.3.1 Static Analyses

When Mercury++ users select a static analysis (static load analysis, load control analysis, or displacement control analysis), the program prompts the user with the size of the analysis. For the displacement controlled example, when the user runs the analysis a message box appears displaying 100 analysis steps. This value is editable if the user wishes to specify a shorter analysis. The value entered is used to assemble the Mercury variable *total*.

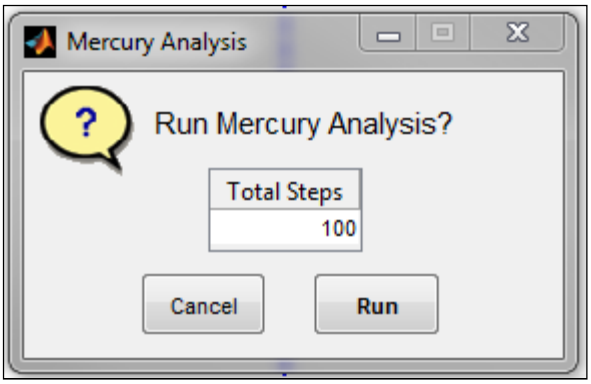

Figure 6.34 Static Analysis Mercury++ Prompt

When the analysis is run, a message box appears, informing the user that the analysis is in progress, and another message appears once the analysis is completed.

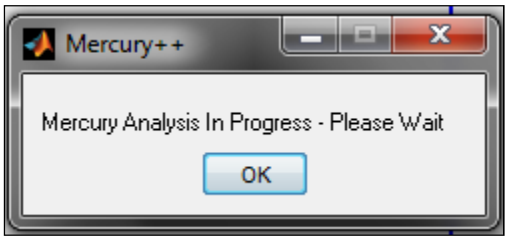

Figure 6.35 Mercury Analysis In Progress User Message

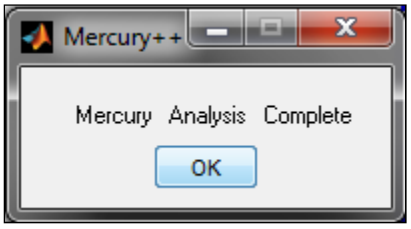

Figure 6.36 Mercury Analysis Complete User Message

6.4.3.2 *Transient Analyses*

When a transient analysis is chosen, Mercury++ users are directed to the Transient Analysis GUI to enter analysis information. The GUI is composed of four panels to enter iterative methods, add lumped mass, specify the integration scheme, specify damping, and specify the number of analysis steps.

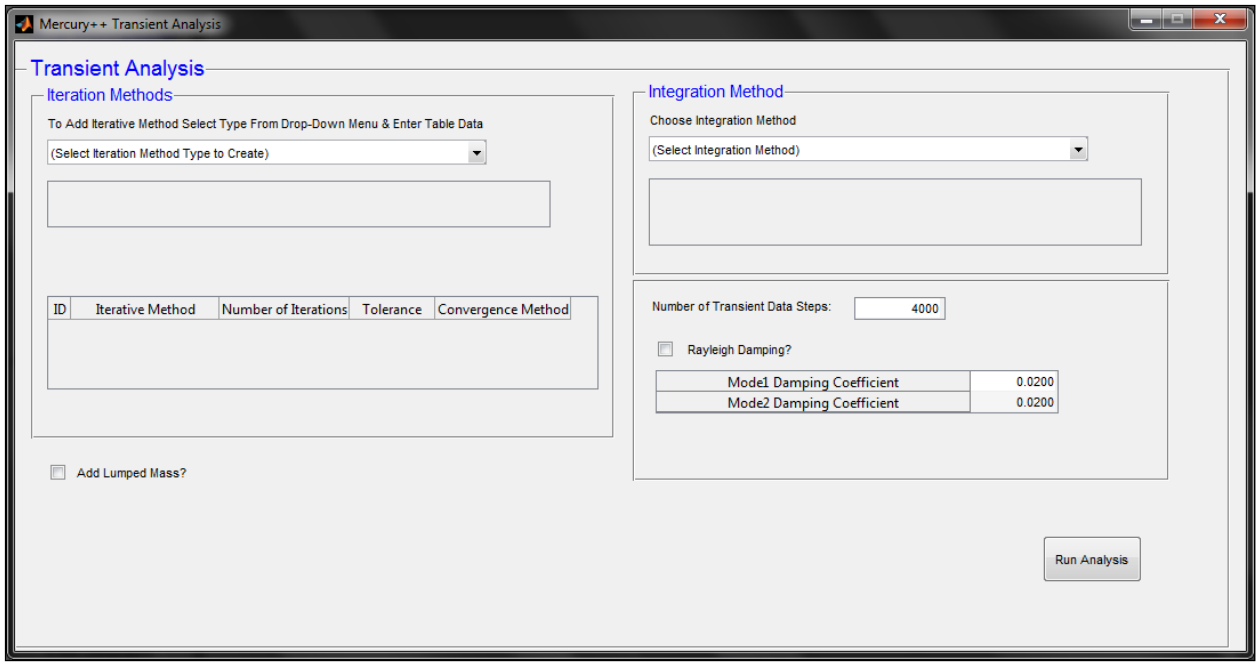

Figure 6.37 Transient Analysis GUI

For a transient analysis, in the Forces Editor GUI, the user selects an input .txt ground acceleration file, specifies a time interval, and assigns an acceleration factor. The total number of input file analysis steps is displayed in the Transient Analysis GUI. Like the static analysis, a shorter analysis may be run by reducing this value. Damping information is entered on the same panel. This panel assembles the Mercury input variables *total* and *dampingfactor*.

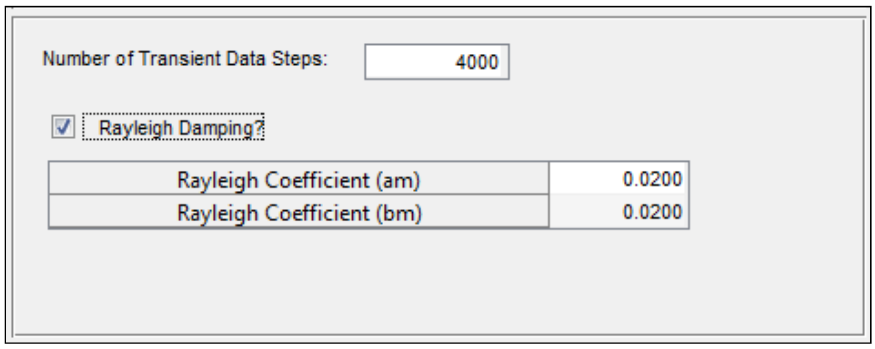

Figure 6.38 Damping and Analysis Size

Iterative methods are entered on the “Iteration Methods” panel and used to assemble the Mercury input variable *iterations*.

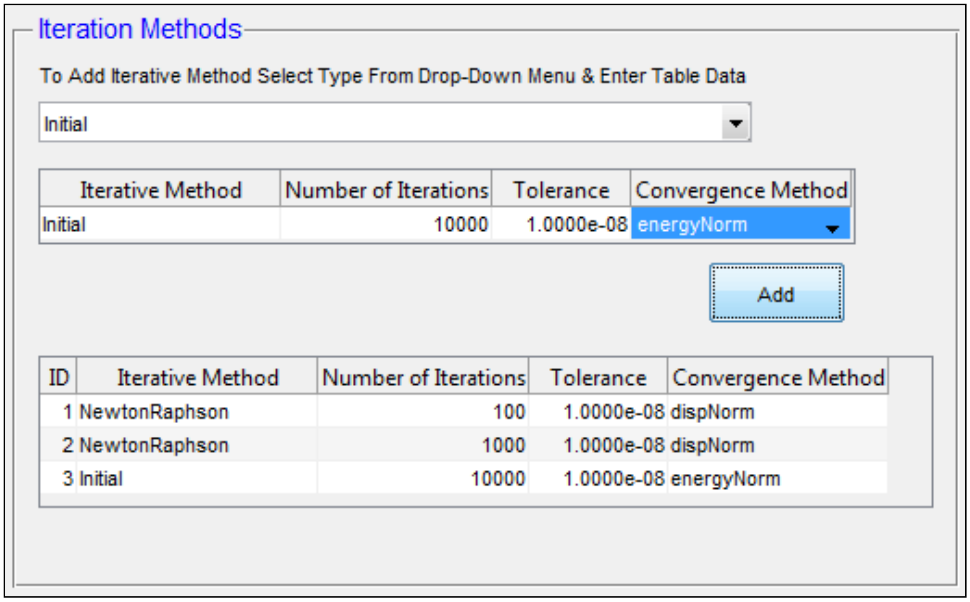

Figure 6.39 Iteration Methods Panel

Mercury++ users have the options of adding lumped mass at nodes by selecting the “Add Lumped Mass?” checkbox and entering mass information in the table. This panel assembles the Mercury input variable *addMass*.

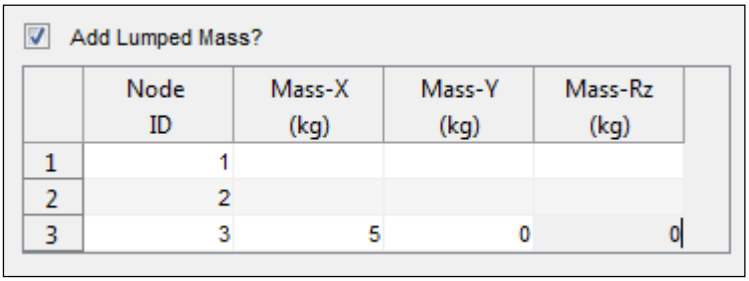

Figure 6.40 Lumped Mass Panel

The analysis integration scheme is entered using the “Integration Method” panel. This panel assembles the Mercury input variable *Integration*.

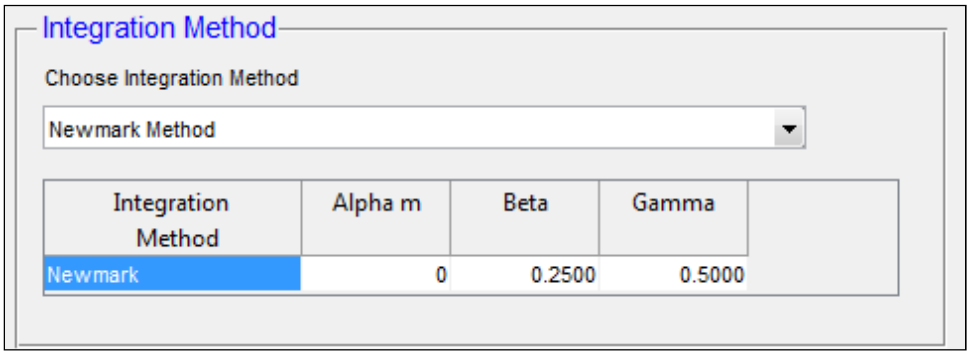

Figure 6.41 Integration Method Panel

When information has been entered in each of the panels, the analysis may be run. When the user begins the analysis, Mercury++ checks the Mercury input file and displays a message is information is missing. Messages appear to inform the user when the analysis begins and when the analysis is complete.

6.4.4 Post-Processor

Analysis data acquisition uses the Mercury++ Post-Processor, accessed from the Post-Processor dropdown menu and from the red Post-Processor panel items.

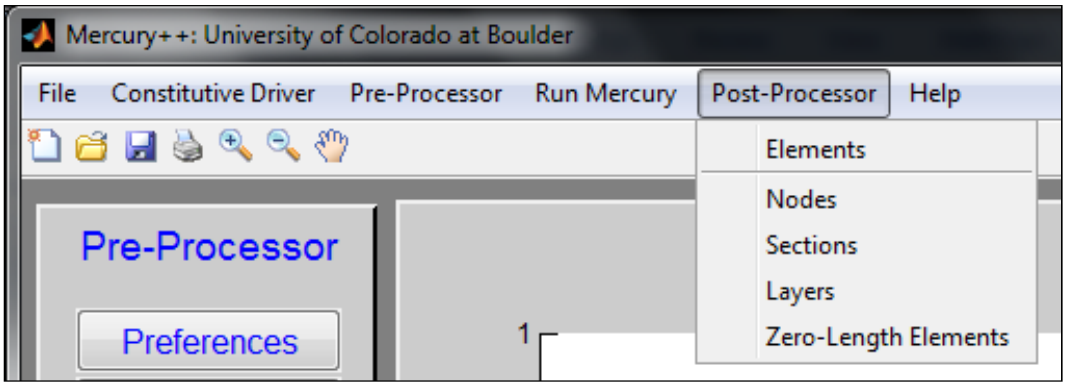

Figure 6.42 Post-Processor Dropdown Menu

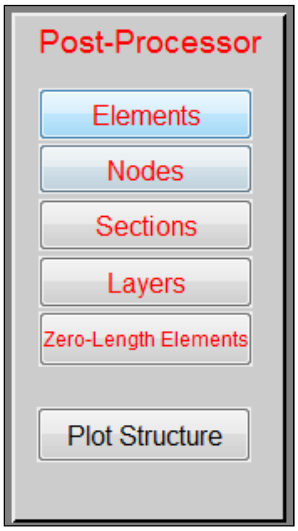

Figure 6.43 Post-Processor Panel

6.4.4.1 *Element Results*

Element shear distribution, moment distribution, and deformed shape results may be obtained, for a user specified time step, using the Element Results GUI. Results may be obtained for all non-zero-length elements by selecting the desired element, the time step at which the information is desired, and the desired information.

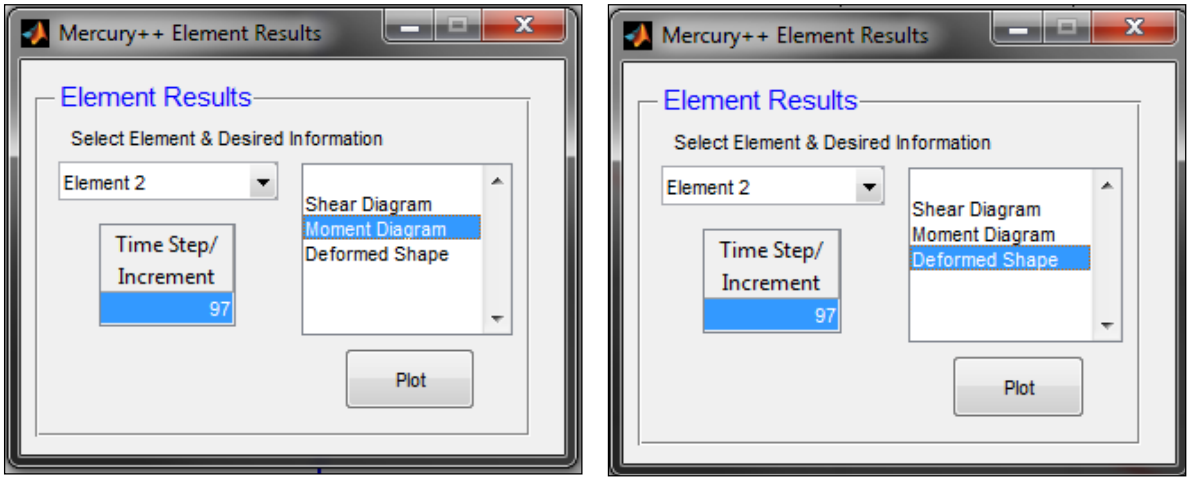

Figure 6.44 Element Results GUI

When a selection is made, the result is plotted on the Mercury++ front panel.

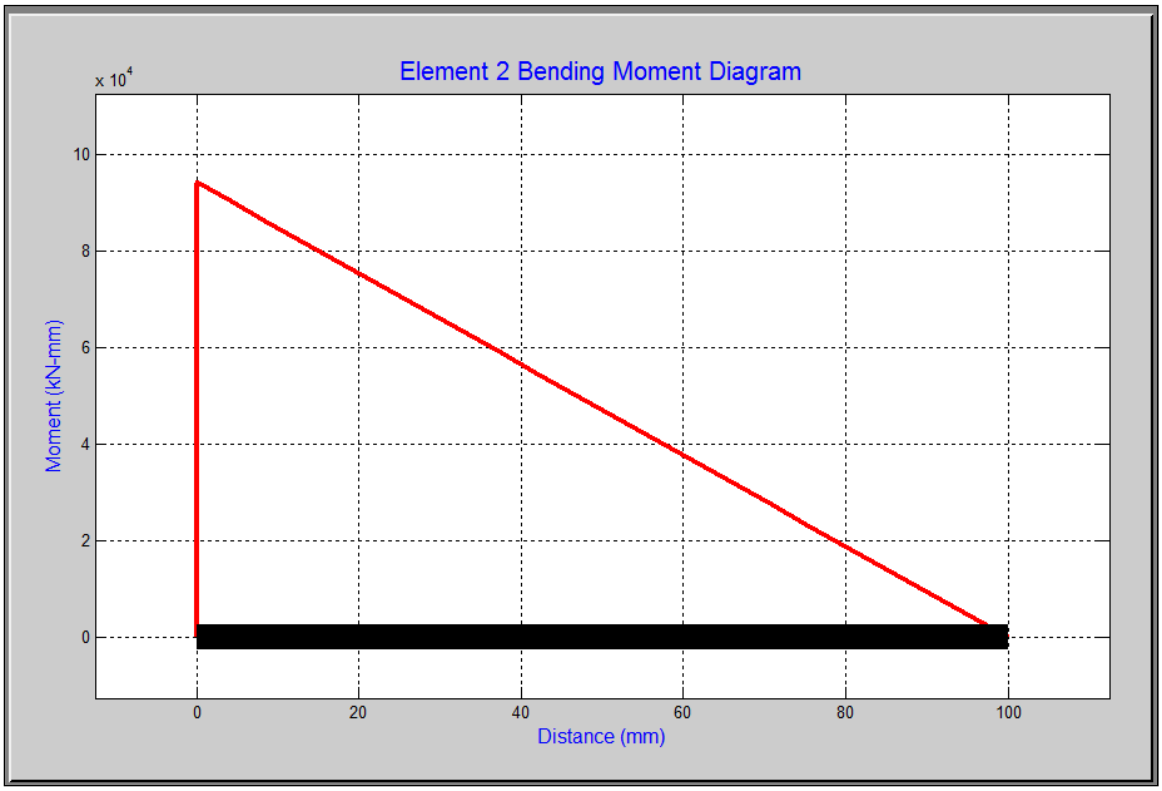

Figure 6.45 Element Moment Diagram

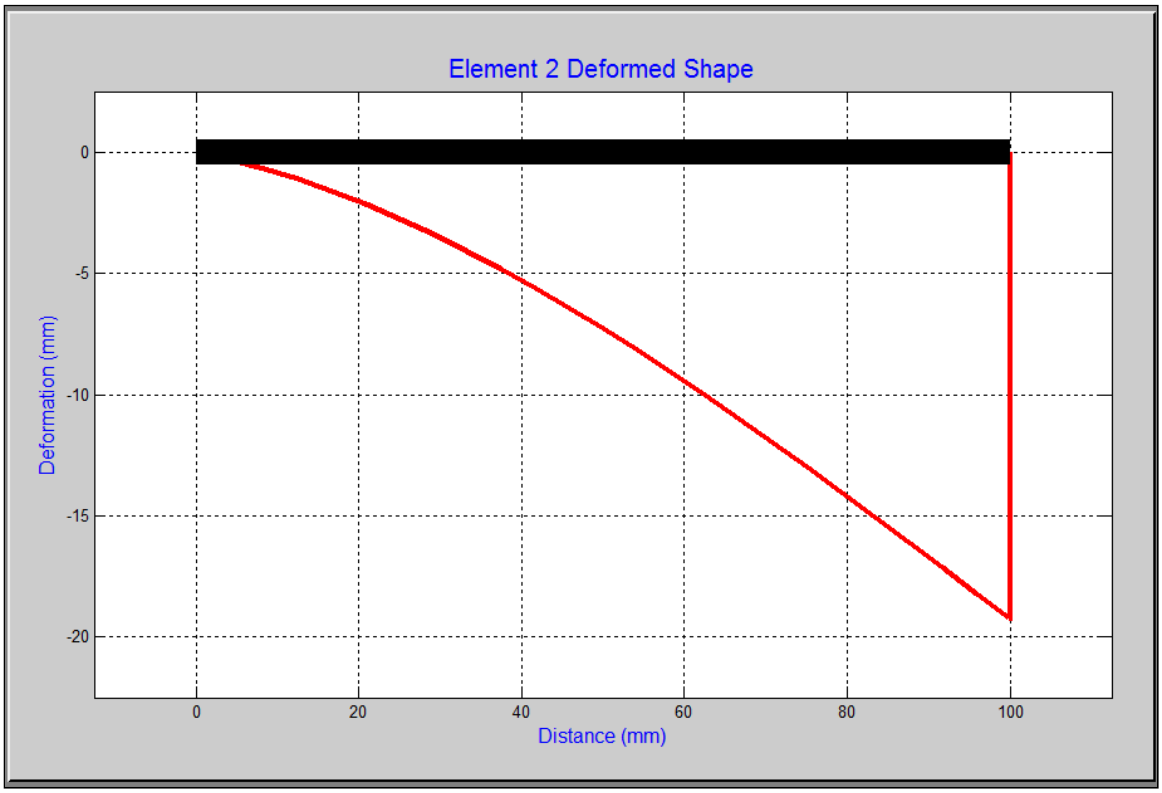

Figure 6.46 Element Deformed Shape

6.4.4.2 Nodal Results

Nodal forces and displacements may be obtained in the Nodal Results GUI. Information is available for all nodes. Basic nodal results (results for a single node for every analysis step) are plotted by selecting the desired node, selecting the desired result, and pressing the "Plot Basic Info" pushbutton.

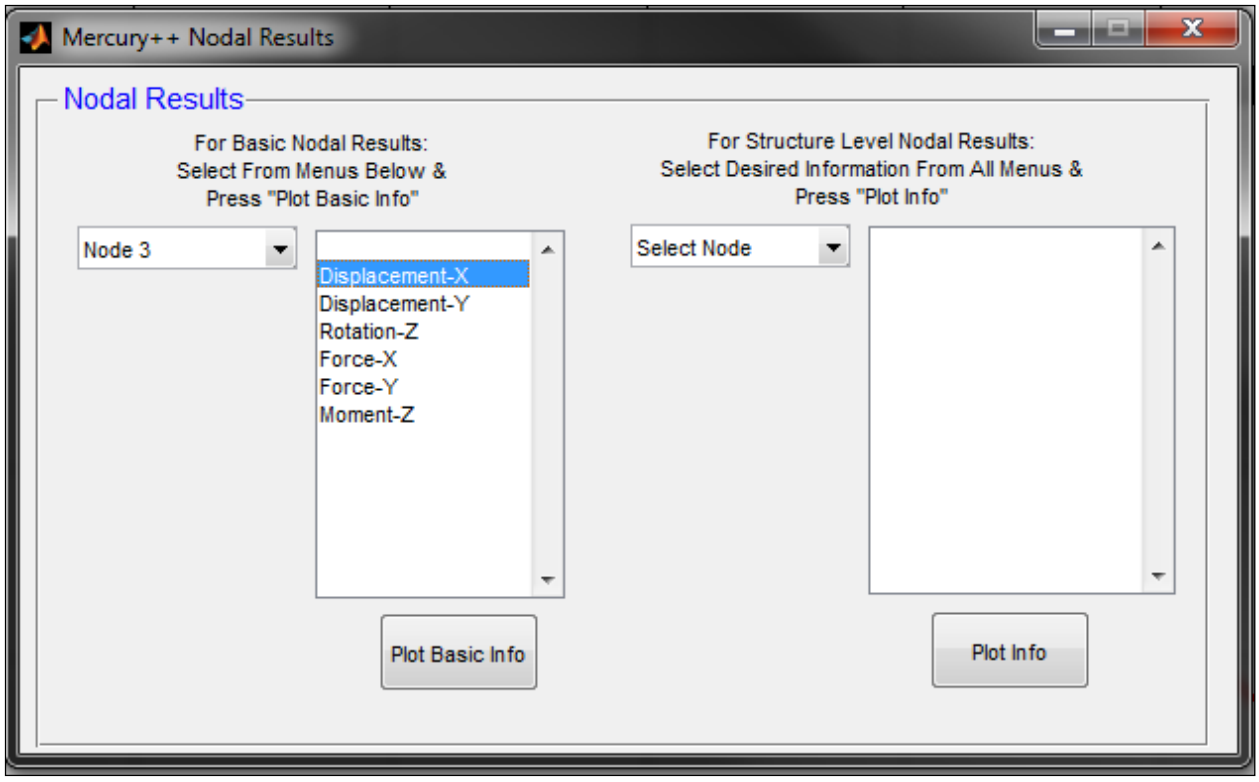

Figure 6.47 Nodal Results Basic Info

Results are plotted on the Mercury++ front panel.

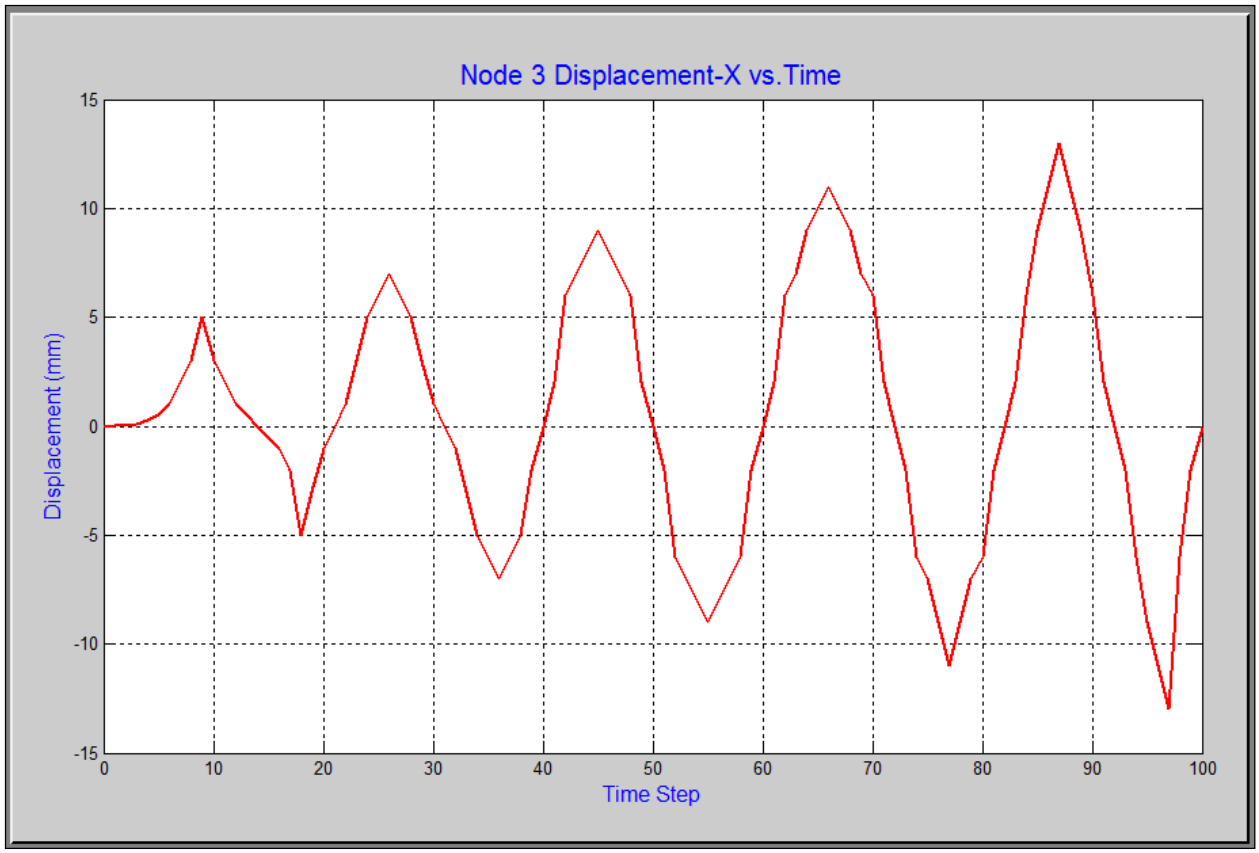

Figure 6.48 Plotting Nodal Results Basic Info

Structure level nodal results may be plotted to compare a value at one node to a value at another node (i.e. node 3 displacement vs. node 1 moment), for every analysis step. The result is obtained by selecting the desired nodes, selecting the desired information, and pressing the "Plot Info" pushbutton. The structure level results table also allows users to plot the interstory drift for all analysis steps.

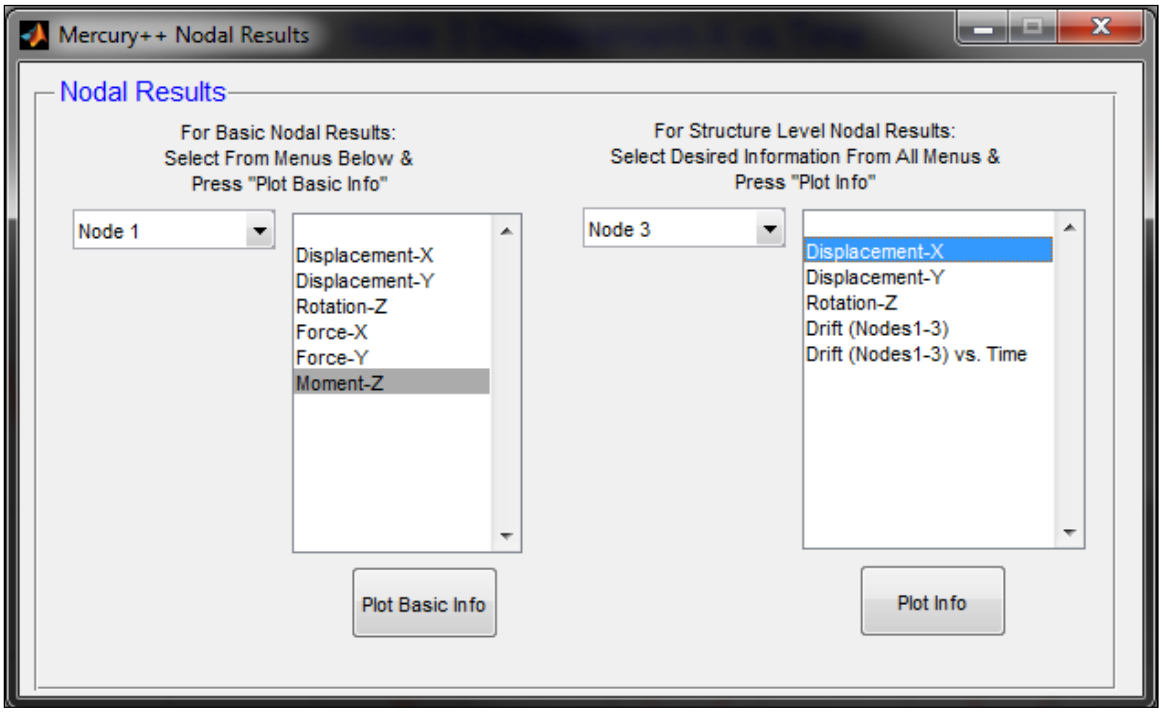

Figure 6.49 Structure Level Nodal Results

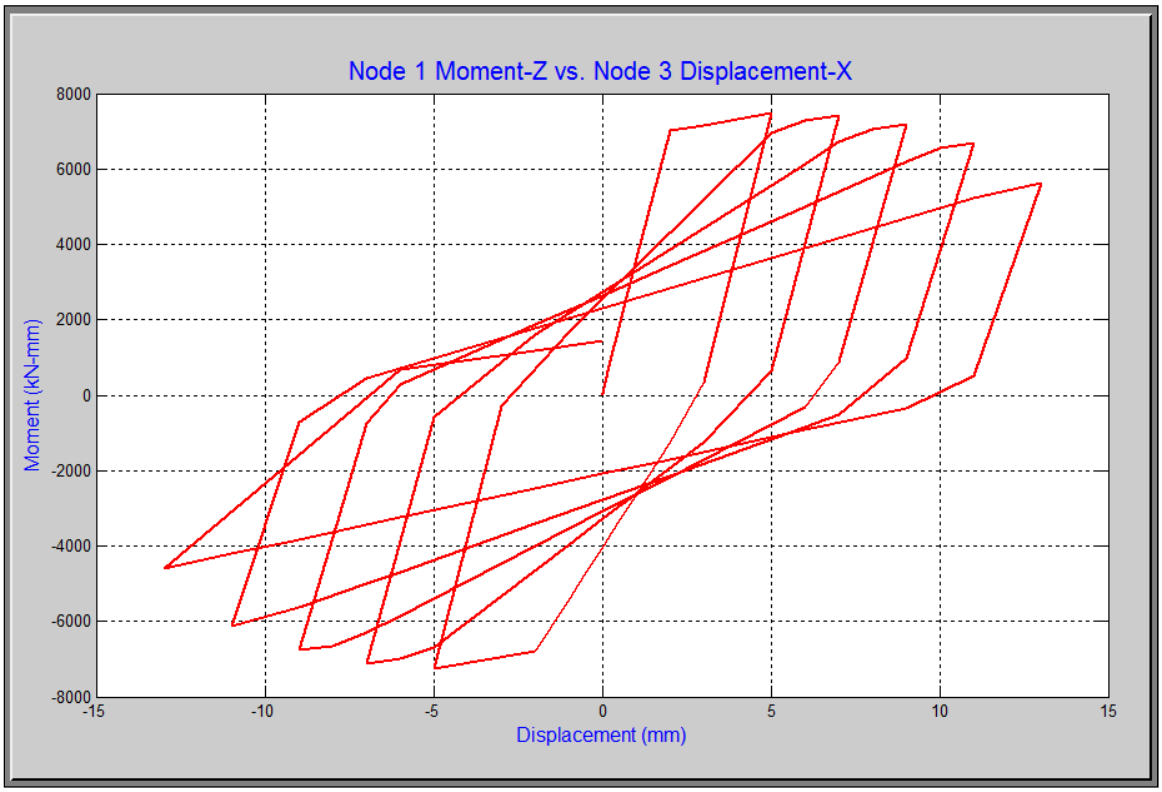

Figure 6.50 Plotting Structure Level Nodal Results

6.4.4.3 Section Results

Sectional results are available in the Section Results GUI for all non-zero-length elements at each Gauss integration point (IP).

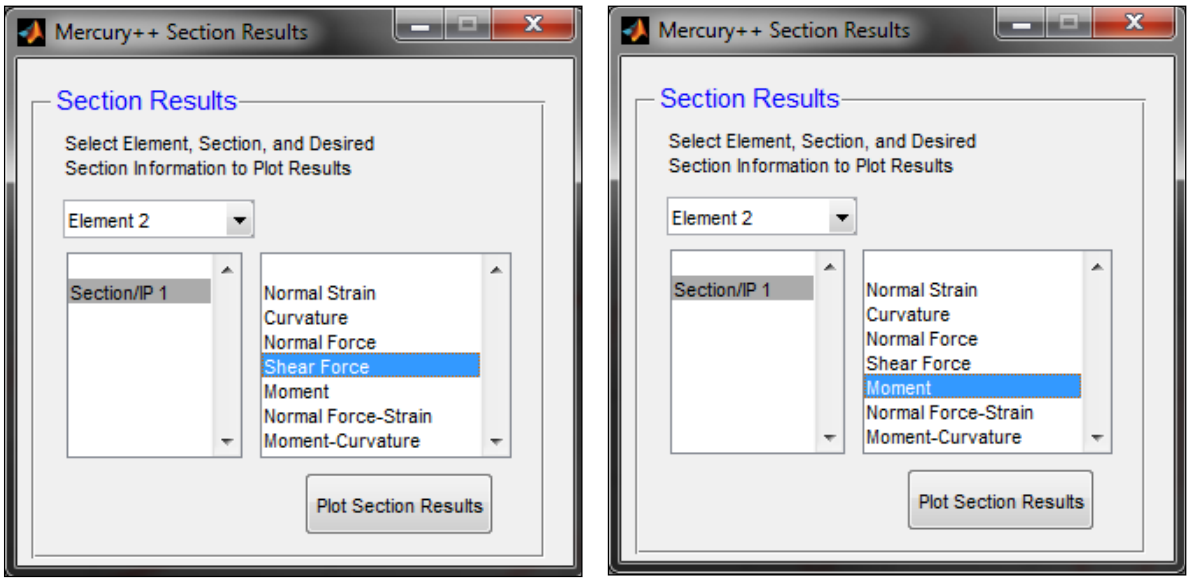

Figure 6.51 Section Results GUI

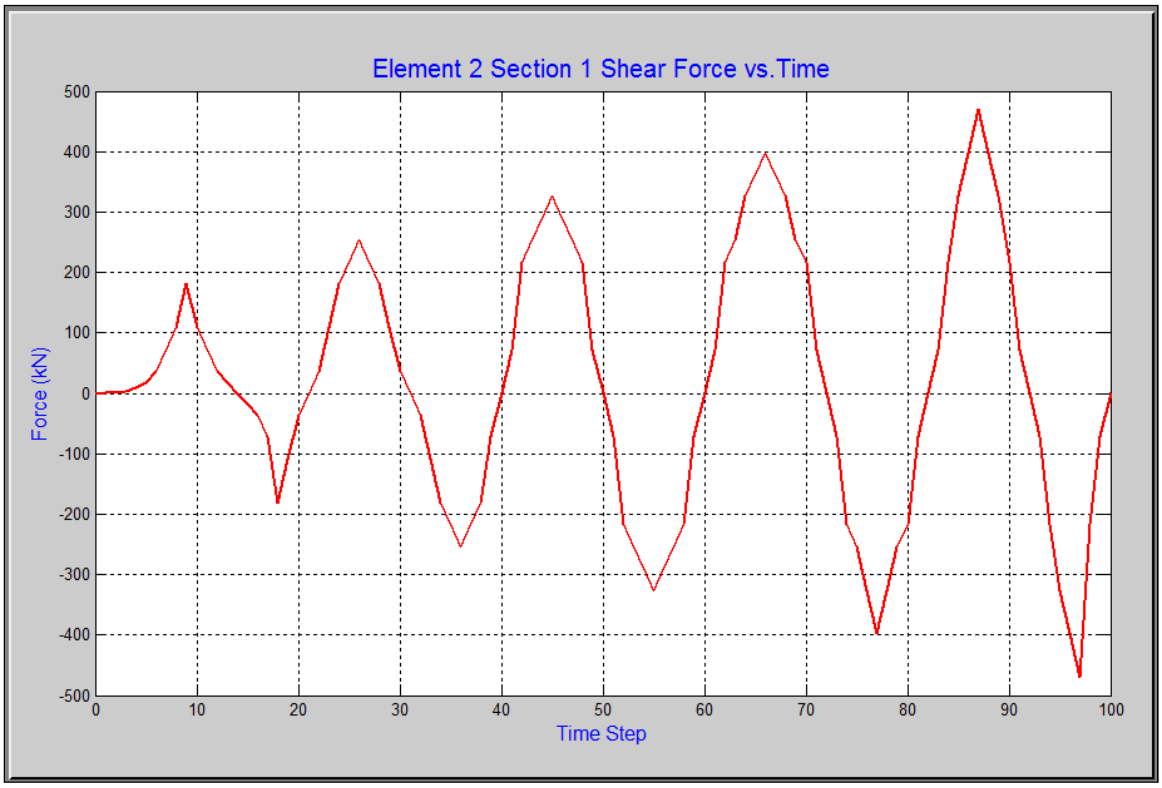

Figure 6.52 Section Results Shear Force

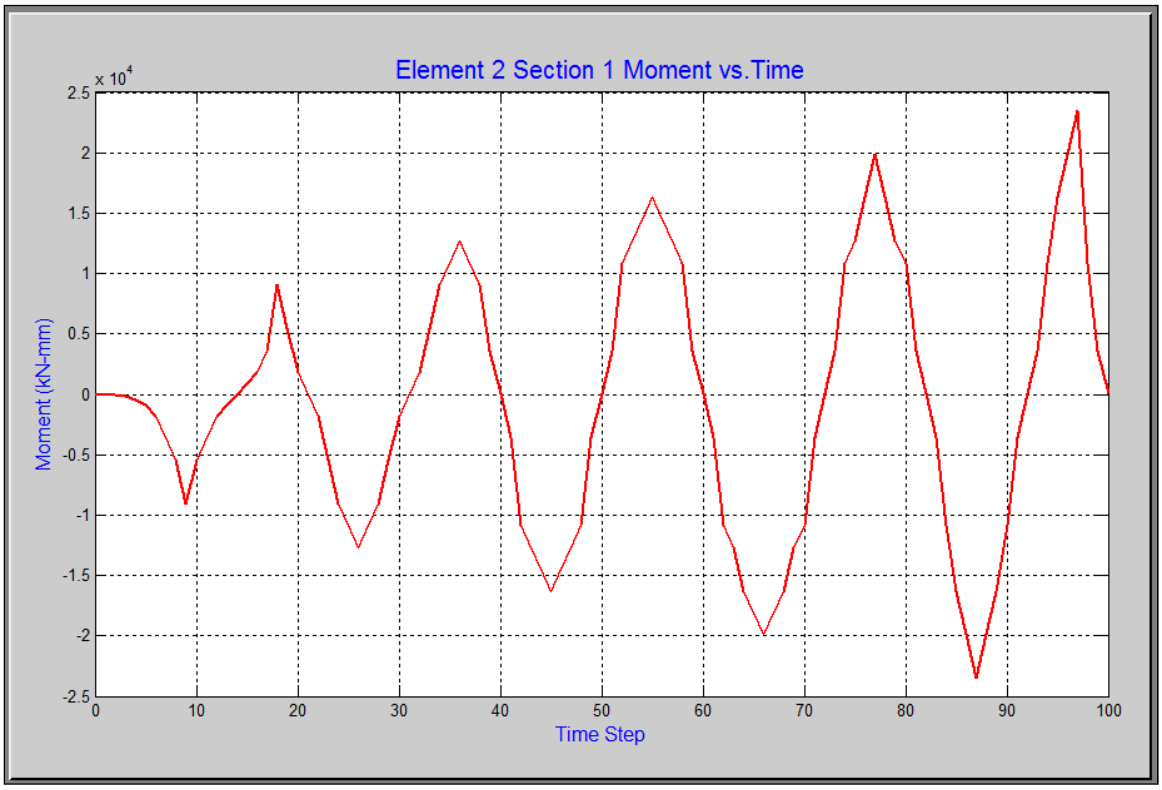

Figure 6.53 Section Results Moment Plot

6.4.4.4 Layer/Fiber Material Results

For layer and fiber section elements, material stress-strain results are obtained using the Material Results GUI. Analysis output is available for all distributed plasticity elements. Elastic beam-column elements do not output layer material results and material results for zero-length elements are available in a separate GUI. Results are available for every layer/fiber at each integration point.

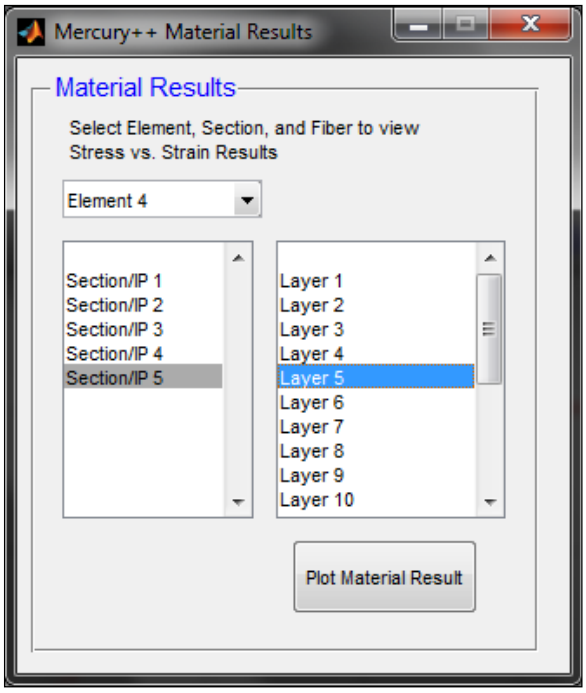

Figure 6.54 Layer Results GUI

Figure 6.55 Layer Stress-Strain Plot

6.4.4.5 *Zero-Length Results*

Element results, for all zero-length members, are accessed using the Zero-Length Results GUI. For zerolength2d uniaxial spring elements, force-deformation or moment-rotation results are available for the axial, shear, and rotation springs.

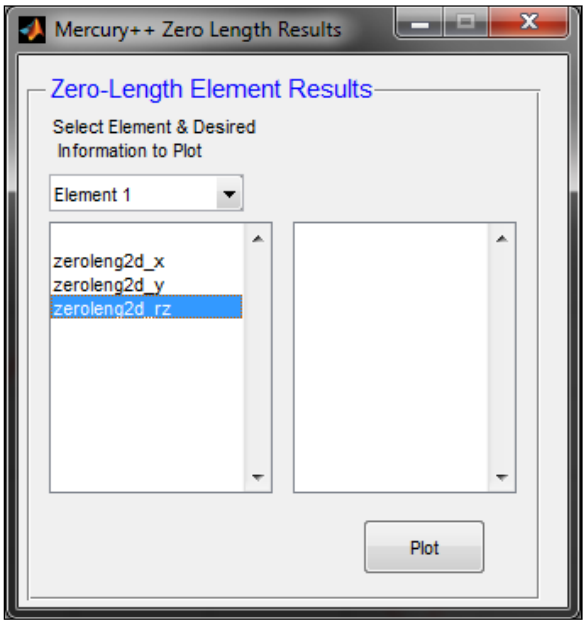

Figure 6.56 Uniaxial zerolength2d Element Results

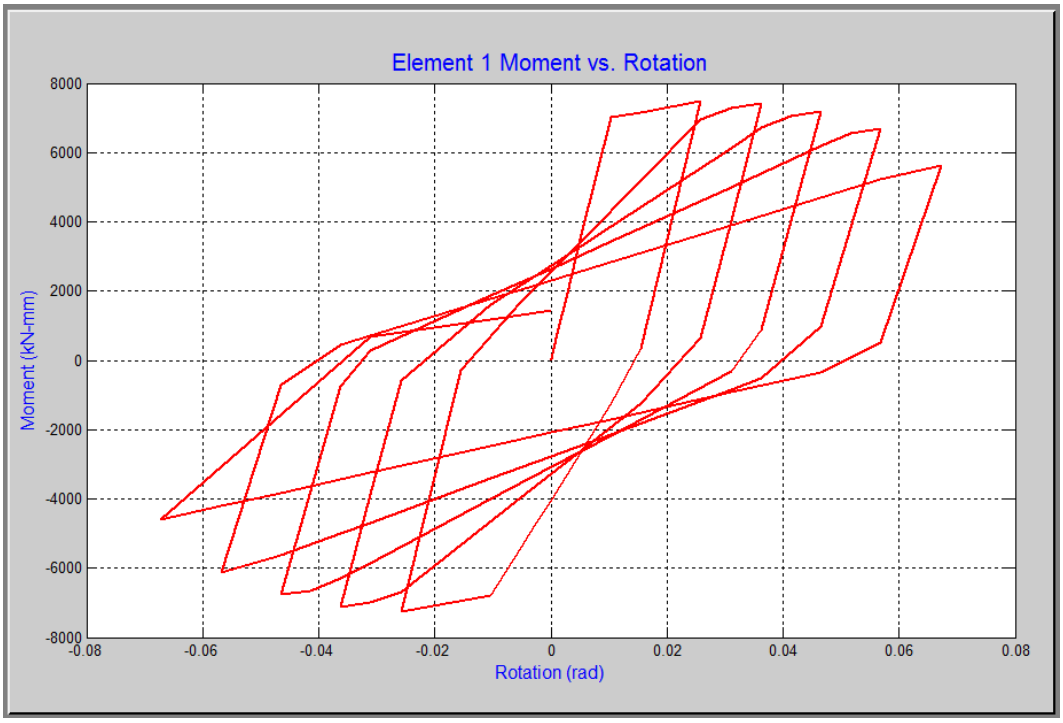

Figure 6.57 Uniaxial Rotation Spring Results

For zerolength2dsection elements, stress-strain results are available for each layer/fiber and force-deformation results are available for all analysis steps.

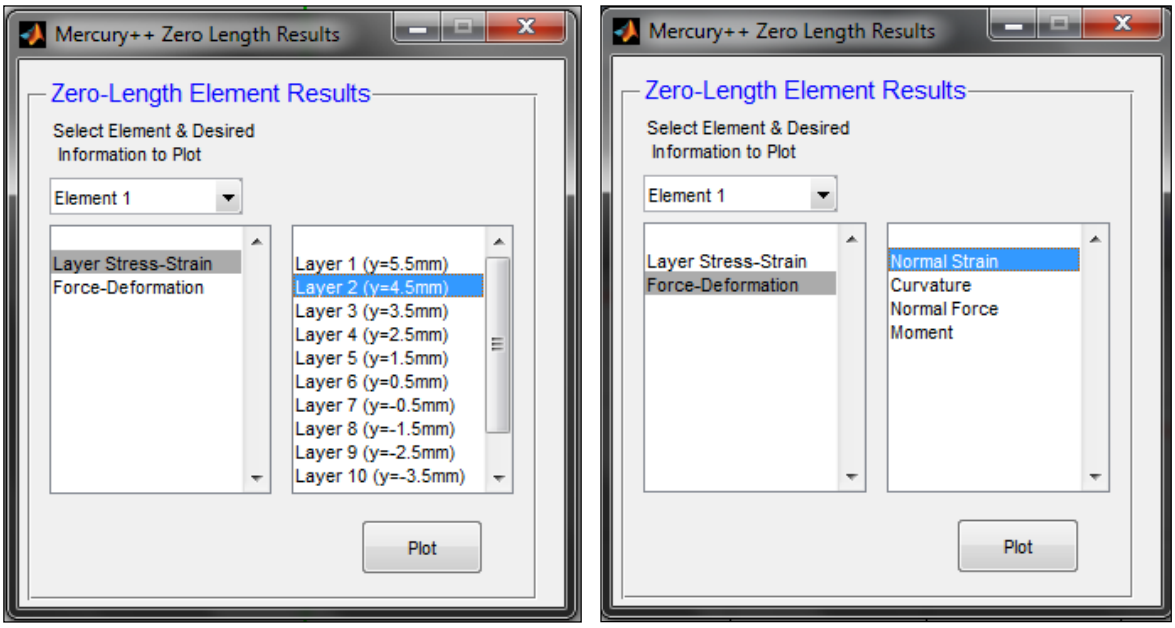

Figure 6.58 Zerolength2dsection Stress-Strain Results

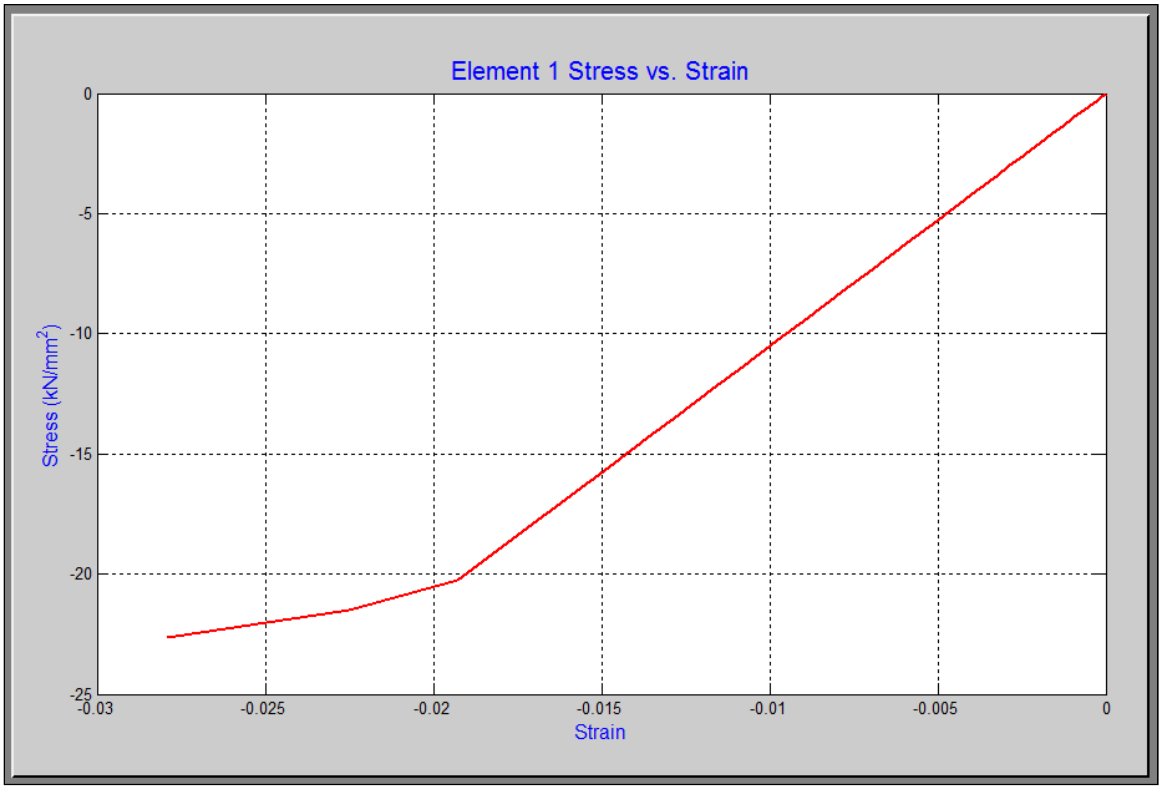

Figure 6.59 Zerolength2dsection Fiber Stress-Strain Results

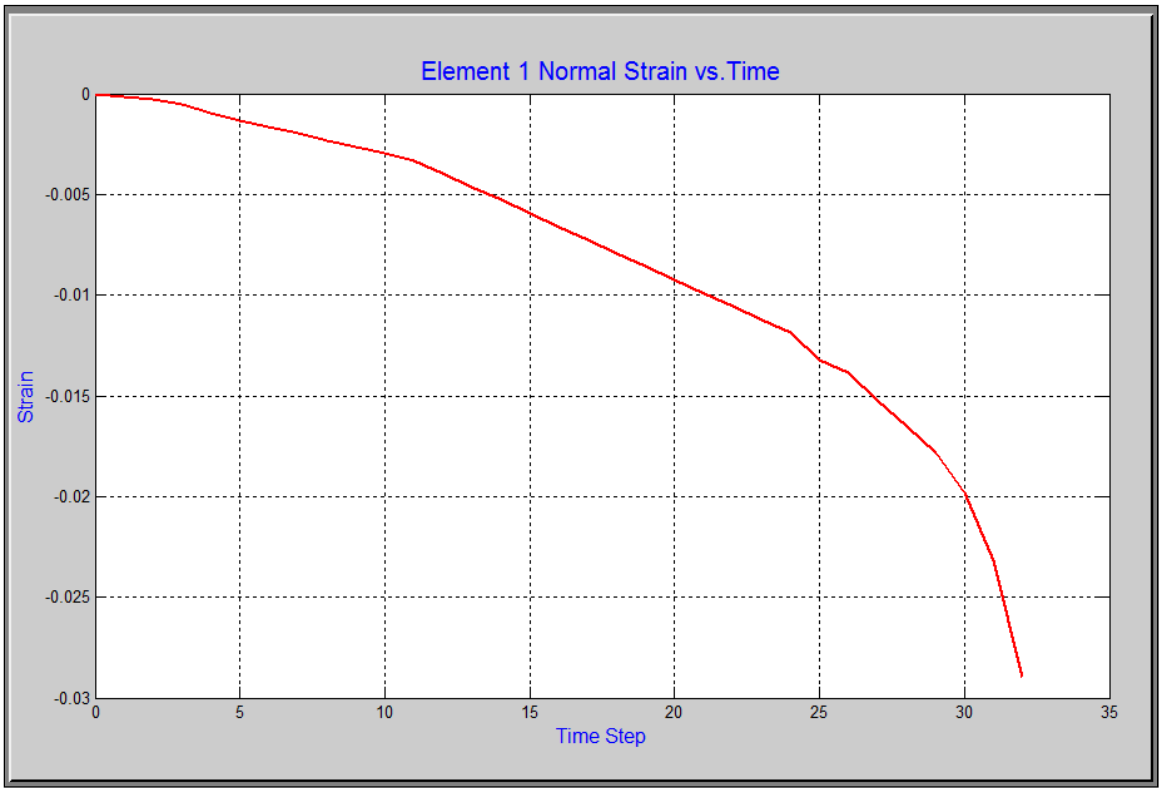

Figure 6.60 Zerolength2dsection Force-Deformation Results

Mercury++ reduces the post-process time by obtaining all output information and providing a series of menus for users to plot any relevant nodal, element, and sectional information. This is a two-folded benefit because users need not transfer information for visualization and all output is obtained without the need for the user to specify the desired information.

6.4.5 Additional Menu Functionality

Mercury++ has the ability to open and save an input file in progress and to print all axes plots including the Mercury input model and analysis results. Each of these options is accessed from the File dropdown menu and shortcuts are provided in the Mercury++ toolbar. When a file is saved, the user is prompted to name the file and select the file path and the Mercury input variables are stored as a structure array (.mat). They are accessible within Mercury++ or separately within MATLAB. The printing tool prints the contents of the axes only.

6.5 Chapter Summary

Chapter 6 presented a detailed description of the pre and post-processor tools developed for Mercury. The tools are a single application from which Mercury users can prepare a Mercury input file, visually verify the input model, access the Constitutive Driver (Chapter 5), run a Mercury analysis, and obtain all analysis output information. Some of the many benefits of these tools, known as Mercury++, are: simple Mercury input file verification and error identification, easy access to analysis results, and easy use of Mercury for those with limited MATLAB programming knowledge.

7 Conclusions and Recommendations for Future Work

7.1 Summary and Conclusions

In order to develop the MATLAB based Mercury platform for lumped plasticity modeling capabilities, two hysteretic constitutive models with stiffness and strength degradation were implemented. Lumped plasticity modeling of a transversely loaded column was explained to describe the process of creating a lumped plasticity model to simulate the flexural response expected from a much more computationally exhaustive distributed plasticity model.

Implementation of a shear failure model was attempted, although the intended response was not achieved. Failure to implement the Limit State model is likely attributed to the element state determination flow in Mercury. The model effectively prompts element unloading; however, the beam-column element should unload instead of the shear spring element. This problem may be addressed by something as simple as assigning an element state determination hierarchy to determine the order in which elements (i.e. zerolength2d) proceed with state determination. Element state determination, however, may not be the source of the problem and material state determination debugging may lead to the intended model results.

7.2 Future Developments

Correction of the Limit State material model is suggested. Further development will depend on the assistance of Mercury's original developer (Kang).

The lumped plasticity calibration process discussed in Chapter 3.5.2 can be a lengthy process when complex models are analyzed. To further develop lumped plasticity modeling in Mercury, the

development of a self-calibrating two-dimensional lumped plasticity element is suggested. The lumped plasticity element should be a combination of:

- (4) nodes
- (1) linear elastic beam-column element (*elasticbeamcolumn*)
- (2) zero-length (*zerolength2d*) elements assigned rigid axial and shear springs

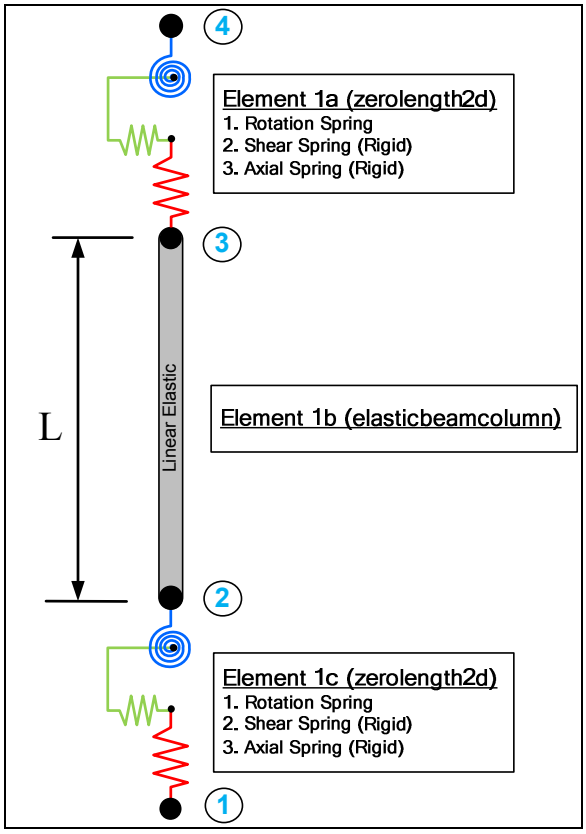

Figure 7.1 Recommended Mercury Element

The element input variables for such an element should be:

- Element tag
- Element name (*lumpedplastbeamcolumn*)
- Node i tag

- Node j tag
- Section tag

The section tag should refer to a *General2D* section. This section type provides:

- Material tag
- Cross-sectional area
- Moments of inertia (I_x and I_z)

A nonlinear constitutive model should be assigned to the *General2D* section; however, it is important that the *lumpedplastbeamcolumn* element is developed to modify the *General2D* section properties for a Mercury analysis because a linear elastic constitutive model must be assigned for the *elasticbeamcolumn* element (Element 1b). The nonlinear model assigned to the section will be used to calibrate the stiffness of the two rotational springs and to calibrate the elastic modulus of the linear beam-column element.

The *lumpedplastbeamcolumn* element should receive material properties for the entire 3 element assembly. The element can then calibrate the properties for the individual components (elements 1a, 1b, and 1c). Calibration should follow the process presented in Chapter 3.5.2, modified to include a second zero-length lumped plasticity element at the top of the column. When either of the hysteresis constitutive models with stiffness and strength degradation (*Hysteresis* or *HysteresisPinch*) is assigned for the *lumpedplastbeamcolumn* element, the desired values of the hysteretic capacity factor (γ) and rate deterioration factor (c) should be assigned and the element should perform calibration of these variables. In other words, if the user wishes to assign values of 50 and 1 for γ and c, respectively, the lumped plasticity element should modify the values of γ and c for each of the rotational springs to reflect the desired results.

The results of the element initialization should produce the following element and material characteristics:

- Linear elastic beam-column element (*elasticbeamcolumn*)
    - General2D section
    - Linear elastic constitutive model with calibrated elastic modulus
- (2) Lumped plasticity elements (*zerolength2d*)
    - Rigid linear elastic axial spring
    - Rigid linear elastic shear spring
    - Nonlinear constitutive model assigned to rotational spring, calibrated to accurately model linear and nonlinear behavior of entire element assembly

Lumped plasticity modeling is driven by extensive research aimed at model calibration. The reader is referred to the report by Haselton et al. (Haselton et al., 2007) for details of calibration results for the peak-oriented hysteresis model (*Hysteresis*) for reinforced concrete frames. These calibration guidelines are suggested as the motivation for development of a self-calibrating lumped plasticity element. Development of such a model, however, does not provide additional benefits other than Mercury user ease.

The Limit State model is extendible to combined axial and shear failure modeling (Elwood & Moehle, 2003). The combined model uses an axial failure surface, similar to the shear failure surface. During the time step when the shear failure surface is reached, the combine shear-axial failure model determines the degraded shear spring stiffness using the two failure models. Similar to the shear failure surface, the axial failure model is a function of the column axial load. As such, the degraded stiffness is not linear under variable axial loading. Furthermore, the degraded stiffness is not taken as a user input, making

this an ideal choice for such an empirical model. As developments are made to the attempted Limit state implementation, it is recommended that this model extension be included.

8 References

Clough, R. W. (1966). Effect of Stiffness Degradation on Earthquake Ductility Requirement. *Report No. 66-16.* Structural Engineering Laboratory, University of California, Berkeley.

Elwood, K. J., & Moehle, J. P. (2003). Shake Table Tests and Analytical Studies on the Gravity Load Collapse of Reinforced Concrete Frames. *PEER Report 2003/01.* Berkeley, CA: Pacific Earthquake Engineering Research Center, University of California.

Haselton, C. B., Liel, A. B., Lange, S. T., & Deierlein, G. G. (2007). Beam-Column Element Model Calibrated for Predicting Flexural Response Leading to Global Collapse RC Frame Buildings. *PEER Report 2007/03.* Berkeley, CA: Pacific Earthquake Engineering Research Center, University of California.

Ibarra, L. F. (2003). Global Collapse of Frame Structures Under Seismic Excitations. *PhD Dissertation.* Department of Civil and Environmental Engineering, Stanford University.

Kang, D. H. (2010). An Optimized Computational Environment for Real Time Hybrid Simulation. *PhD Dissertation.* Department of Civil, Environmental, and Architectural Engineering, Univerisity of Colorado at Boulder.

Krawinkler, H. (1999). Challenges and Progress in Performance-Based Earthquake Engineering. *International Seminar on Seismic Engineering for Tomorrow - In Honor of Professor Hiroshi Akiyama*, (pp. 1-10). Tokyo, Japan.

OpenSees. (2005). *Open System for Earthquake Engineering*. Retrieved June 10, 2011, from Pacific Earthquake Engineering Reseach Center (PEER), University of California, Berkeley: http://opensees.berkeley.edu/

Rahnama, M., & Krawinkler, H. (1993). Effects of Soft Soil and Hysteresis Model on Seismic Demands. *PhD Dissertation.* Department of Civil and Environmental Engineering, Stanford University.

Saouma, V. E. (2011). *Structural Analysis.* Manuscript under preparation.

Sivaselvan, M. V., & Reinhorn, A. M. (2000). Hysteretic Models for Deteriorating Inelastic Structures. *Journal of Engineering Mechanics , 126* (6), 633-640.

Takeda, T., Sozen, M., & Nielsen, N. (1970). Reinforced Conrete Response to Simulated Earthquakes. *Journal of the Structural Division , 96* (12), 2557-2573.

Wen, Y. K. (1976). Method For Random Vibration of Hysteretic Systems. *Journal of Engineering Mechanics Division, ASCE , 102* (2), 249-263.