MERCURY

Optimized Software for (Single Site) Hybrid Simulation
From Pseudo Dynamics to Real Time Hybrid Simulation

User's Manual

Ver. 1

Dae-Hung Kang
Gary Haussmann
Victor E. Saouma

Fast Hybrid Testing Laboratory

http://fht.colorado.edu

Department of Civil Environmental and Architectural Engineering
University of Colorado, Boulder
Boulder, Colorado 80309-0428

Contents

List of Figures

List of Tables

Chapter 1

MATLAB Version

Mercury's input data for the Matlab and the c++ version are nearly identical. This document describes the input for Matlab. Since the c++ version uses the Lua scripting language (analogous to TCL in OpenSees), a simple program (to be embedded in the c++ version) will translate the Matlab format into Lua scripts.

Note: In this preliminary version of Mercury, no attempt has been made to simplify (generate/automate) data entry, and there is not (yet) a mesh generator for the program. Those are simple future developments.

```
%%------------------------------------------------------------------------
% Unit = {'force unit','displacement unit'};
%%------------------------------------------------------------------------
% StrMode = {ndim, ndofpn};
%%------------------------------------------------------------------------
% Iteration = { 'static', { {'Linear'};
%                           {'NewtonRaphson', number of iterations, tolerance, 'DisplNorm'};
%                           {'ModifiedNewtonRaphson', number of iterations, tolerance, 'EnrgyNorm'};
%                           {'InitialStiffness', number of iterations, tolerance, 'ForceNorm'};
%                         };
%               'transient', { {'Linear'};
%                              {'NewtonRaphson', number of iterations, tolerance 'DisplNorm'};
%                              {'ModifiedNewtonRaphson', number of iterations, tolerance, 'EnrgyNorm'};
%                              {'InitialStiffness', number of iterations, tolerance, 'ForceNorm'};
%                            };
%               'element', { {'Linear'};
%                            {'NewtonRaphson', number of iterations, tolerance, 'DispNorm'};
%                            {'ModifiedNewtonRaphson', number of iterations, tolerance, 'EnrgyNorm'};
%                            {'InitialStiffness', number of iterations, tolerance, 'ForceNorm'}
%                          }
%             };
%%------------------------------------------------------------------------
% Integration = {'Newmark', alphaformoment, beta, gamma, am, bk};
% Integration = {'HHT', alphaformoment, alpha, beta, gamma, am, bk};
% Integration = {'Shing', alphaformoment, alpha, beta, gamma, am, bk, number of iteration}
% eigens  = {zeta_1, zeta_2}
% addMass = {nodtag1, m1, m2, m3, m4, m5, m6;
%            nodtag2, m1, m2, m3, m4, m5, m6};
%%------------------------------------------------------------------------
% nodcoord = { nodtag1, x1, y1, z1}
% constraint = {nodtag1, id1, id2, id3}
%%------------------------------------------------------------------------
% elements = {{eletag, 'Simple2DTruss', snode, enode, sectag};
%             {eletag, 'StiffnessBased2DBeamColumn', snode, enode, nIp, sectag};
%             {eletag, 'FlexibilityBased2DBeamColumn1', snode, enode, nIp, sectag};
%             {eletag, 'FlexibilityBased2DBeamColumn2', snode, enode, nIp, sectag};
%             {eletag, 'ZeroLength2D', snode, enode, localx_mattag, localy_mattag, localz_mattag, angel(radian) from local to global}
%             {eletag, 'ZeroLength2DSection', snode, enode, angel from local to global(radian), sectag}
%             {eletag, 'Hybrid2DBeamColumnNumerical', snode, enode, sectag for general section, 'IP address', port number for sending
%                  {3, 'Hybrid2DBeamColumnNumerical', 3, 4, 1, '128.138.228.71', 3000}};
%%------------------------------------------------------------------------
% We need imaginary to make StiffnessBased2DBeamColumn with elastic material
% for Hybrid2DBeamColumnNumerical
% ---------- MercuryMatlab hybrid program ----------
% {eletag, 'Hybrid2DBeamColumnPhysical', 'IP address', port number for sending and receiving}
%%------------------------------------------------------------------------
% sections = {sectag, 'General', {mattag, A, Ix, Iy, Iz};
%             sectag, 'Layer', {mattag1, A1, y1;
%                               mattag2, A2, y2};
%             sectag, 'Fiber', {mattag1, A1, y1, z1;
%                               mattag2, A2, y2, z2}};
%%------------------------------------------------------------------------
% materials = { {1, 'Elastic', E, G, density};
%               {2, 'Hardening', E, sy, Hiso, Hkin, density};
%               {3, 'Bilinear', E, sy, b, density, a1, a2, a3, a4};
%               {4, 'ModGMP', E, sy, b, R0, cR1, cR2, density, a1, a2, a3, a4};
%               {5, 'ModKP', sc, ec, scu, ecu, lambda, st, Ets, density};
%               {6, 'AnisotropicDamage1D', E, nu, kappa0, A, a, Dc, density};
%               {7, 'AnisotropicDamage', E, nu, kappa0, A, kh, kd, Kh, Kd, Dc, density};}
%%------------------------------------------------------------------------
% forces = {forcetag, 'Static', {'NodalForces', {nodtag1, global axis1, magnitude1;
%                                                nodtag2, global axis2, magnitude2};
%                                'ElementDistributedForces', {eletag1, local axis1, magnitude1;
%                                                             eletag2, local axis2, magnitude2}};
%           forcetag, 'LoadCtrl', {nodtag1, global axis1, {m1, m2, m3};
%                                  nodtag2, global axis2, {m1, m2, m3}};
%           forcetag, 'DispCtrl', {nodtag1, global axis1, {m1, m2, m3}
%                                  nodtag2, global axis2, {m1, m2, m3}};
%           forcetag, 'Acceleration',{factor, {time1, m1 of globla x-axis, m1 of global y-axis;
%                                              time2, m2 of globla x-axis, m1 of global y-axis}}};
%%------------------------------------------------------------------------
```

1.1 Preface Block

This initial block declares units, spatial dimension of the structure, and the number of degrees of freedom per node.

1.1.1 Unit

The `Unit` declares selected units for analysis.

```
Unit = ['F, L']
```

`F`: force, and `L`: length units. For example **Unit = ['kN, mm']**

1.1.2 Structural mode

The `StrMode` declares dimension of structure and number of degrees of freedom per node.

```
StrMode = [ndim, ndofpn]
```

where `ndim` refers to the spatial dimension of the structure [2|3] and `ndofpn` to the number of degrees of freedom per node in global reference [2|3|6]

1.2 Control Block

The control block defines basic information about the structural analysis.

1.2.1 Analysis

The `Analysis` defines the analysis mode.

```
Analysis = 'AnalysisMode'
```

where `AnalysisMode`: [Static—Transient].

1.2.2 Iteration

The `Iteration` defines the iterative method adopted in the solution of the nonlinear system of equations.

```
Iteration = 'IterationMode'
```

where `IterationMode` can be

- `Linear`: Use only one single iteration to solve the nonlinear system of equations in a step.
- `NewtonRaphson`: Use the Newton-Raphson method in which the tangent stiffness matrix is updated at each iteration.
- `ModifiedNewtonRaphson`: Use the modified Newton-Raphson method in which the tangent stiffness is updated at each step.
- `InitialStiffness`: Use the initial stiffness matrix and the the tangent stiffness matrix is never updated.
- `ModifiedInitialStiffness`: Use the initial stiffness matrix as modified by Shing for hybrid simulation. This is limited to transient analysis.

1.2.3 Iteration for element

The `IterationEle` defines the iterative method adopted in the solution of the nonlinear system of equations in element level. In `Mercury`, this command is only used in `FlexibilityBased2DBeamColumn`.

```
IterationEle = 'IterationEleMode'
```

where `IterationEleMode` can be

- `NewtonRaphson`: Use the Newton-Raphson method in which the tangent stiffness matrix is updated at each iteration.
- `InitialStiffness`: Use the initial stiffness matrix and the the tangent stiffness matrix is never updated.

1.2.4 Integration

The `Integration` defines the types of numerical integration used in the transient analysis.

- Newmark β method
`Integration` = {'`Newmark`', {α_m, `a`, `b`, β, γ}}
- Hilber-Hughes-Taylor method(HHT method)
`Integration` = {'`HHT`', {α_m, `a`, `b`, α}}

- `IntegrationMode`
  - Newmark: Use the Newmark β method for transient analysis with ground acceleration.
  - HHT: Use the Hilber-Hughes-Taylor method for transient analysis with ground acceleration. This method is often referred to as 'α' method.
- α_m: Coefficient premultiplying the rotational mass in a beam column. Recall that $[\mathbf{m}] = \frac{m}{2}\lfloor\ 1 \quad \alpha_m L^2/210 \quad 1 \quad \alpha_m L^2/210\ \rfloor$, (**?**).
- `a`: Rayleigh damping coefficient of mass matrix
- `b`: Rayleigh damping coefficient of stiffness matrix
- β: Coefficient β in the Newmark β method
- γ: Coefficient γ in the Newmark β method
- α: Coefficient α in the HHT method

For Example: **Integration = {'NewtonRaphson', {1/78, 0.02, 0.08, 1/6, 1/2 }}** for linear acceleration in the Newmark β method,

1.2.5 MassInput

Lumped masses are automatically determined by Mercury. If additional masses are to be assigned, then the `MassInput` command must be used and a mass specified for each node.

`MassInput`	= {	`nodtag`^{1}, m_1^1, m_2^1 [m_3^1, m_4^1, m_5^1, m_6^1]	;
		...	;
		`nodtag`^{i}, m_1^i, m_2^i [m_3^i, m_4^i, m_5^i, m_6^i]	;
		...	;
		`nodtag`^{n}, m_1^n, m_2^n [m_3^n, m_4^n, m_5^n, m_6^n]	}

MassInput is based on lumped mass. Where n is smaller than or eqaul to the total number of nodes. At each component m_j^i, j means the j^{th} degree of freedom of the i^{th} nodtag

1.2.6 Convergence Criteria

`StrMiter`	= StrMax
`EleMiter`	= EleMax
`Convergence`	= Norm
`ConvergenceEle`	= NormEle
`Tolerance`	= Tol

Where

- `StrMax`: Maximum number of iterations
- `EleMax`: Maximum number of iterations within an element when using flexibility-based 2D beam-column element with internal iteration
- `Norm`: User can select norm criterion for structural level. Mercury support three types, 'DispNorm', 'ForceNorm' and 'EnergyNorm' criterion.
- `NormEle`: User can select norm criterion for element level when using flexibility-based 2D beam-column element. Mercury support three types, 'DispNorm', 'ForceNorm' and 'EnergyNorm' criterion.
- `Tol`: Convergence criteria on the residuals

Example:
`FlexibilityBased2DBeamColumn` in Sec. 1.4, then we could have: `StrMiter = 20`; `EleMiter = 50`; `Convergence = 'ForceNorm'`; `ConvergenceEle = 'EnergyNorm'`; `Tolerance = 1.0e-8`

1.3 Geometry Block

The geometry block defines nodal coordinates and their constraints assuming a right handed coordinate system.

1.3.1 Nodal coordinates

The `nodcoord` assigns coordinates of nodes.

`nodcoord`	=	{	`nodtag`$_1$, x_1, y_1 [z_1] ;
			... ;
			`nodtag`$_i$, x_i, y_i [z_i] ;
			... ;
			`nodtag`$_n$, x_n, y_n [z_n] }

for example:

```
Node = { 1,  0.0,  0.0 ;
         2,  1.0,  3.0 ;
         3,  2.0,  0.0 }
```

1.3.2 Boundary condition

The `constraint` command assigns boundary conditions to the nodes. Each node has to have as many constraint as d.o.f's per node.

`constraint`	=	{	`nodtag`^{1}, id_1^1, id_2^1 [id_3^1, id_4^1, id_5^1, id_6^1] ;
			... ;
			`nodtag`^{i}, id_1^i, id_2^i [id_3^i, id_4^i, id_5^i, id_6^i] ;
			... ;
			`nodtag`^{n}, id_1^n, id_2^n [id_3^n, id_4^n, id_5^n, id_6^n] }

Where 0 corresponds to a free dof, and 1 to a fixed one. For example:

```
constraint = { 3,  1,  1 ;
               5,  1,  0 }
```

1.4 Element Block

The `elements` command defines element type, nodal connectivity, and basic sectional information. These may vary with the element type.

`elements`	=	{	`eletag`$_1$, `eletype`$_1$, `in`$_1$, `jn`$_1$, { `SecInfo`$_1$ } ;
			... ;
			`eletag`$_i$, `eletype`$_i$, `in`$_i$, `jn`$_i$, { `SecInfo`$_i$ } ;
			... ;
			`eletag`$_n$, `eletype`$_n$, `in`$_n$, `jn`$_n$, { `SecInfo`$_n$ } }

Where

- `eletag`$_i$: Sequential integer identifying the i^{th} element
- `eletype`$_i$: i^{th} element type (see below)
- `in`$_i$: First node
- `jn`$_i$: Second node
- `SecInfo`$_i$: Basic section information for the element (see below)

1.4.1 Truss element (Sec. ??)

This is the classical two noded axial element, however its cross section can be characterized by either a constant properties; general, layered or fiber.

`eletag`, 'Simple2DTruss', `in`, `jn`, { `sectag` }

Where:

- `eletag`: Element tag
- `in`: First node of element `eletag`
- `jn`: Second node of element `eletag`
- `sectag`: Integer number identifying section of the truss element `eletag`

1.4.2 Stiffness-based 2D beam-column element(Sec. ??)

Stiffness-based 2D beam-column element can have a constant, layered, or fiber section. Its numerical integration is based on Gauss-Legendre quadrature rule.

```
eletag, 'StiffnessBased2DBeamColumn', in, jn, { sectag, nIp }
```

Where:

- `eletag`: Element tag
- `in`: First node of element `eletag`
- `jn`: Second node of element `eletag`
- `sectag`: Integer number identifying section of element `eletag`
- `nIp`: Order of integration of element `eletag`

1.4.3 Flexibility-based 2D beam-column element (Sec. ??)

Flexibility-based 2D beam-column element can have a constant, layered or fiber section. Its numerical integration is based on Gauss-Lobatto quadrature rule.

```
- Flexibility-based 2D beam-column with element iteration loop
eletag, 'FlexibilityBased2DBeamColumn', in, jn, { sectag, nIp }
- Flexibility-based 2D beam-column without element iteration loop
eletag, 'FlexibilityBased2DBeamColumnNoIter', in, jn, { sectag, nIp }
```

Where

- `eletag`: Element tag
- `in`: First node of element `eletag`
- `jn`: Second node of element `eletag`
- `sectag`: Integer number identifying section of element `eletag`
- `nIp`: Order of integration of element `eletag`

1.4.4 Zero-length 2D element (Sec. ??)

Zero length element is used to model lumped plasticity. It can account for stiffness degradation in flexure and shear. It neglects axial-flexural coupling effect and depends on force and deformation history as well as on the section characteristics.

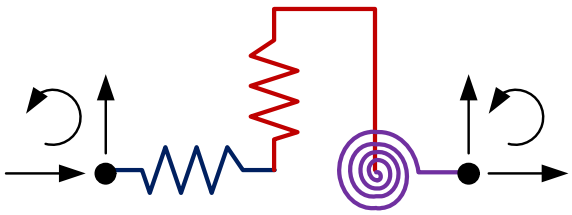

Figure 1.1: Zero-length 2D element

```
eletag, 'ZeroLength2D', in, jn, { ndofpnlocal,
            mattag1, mattag2, mattag3, angle }
```

Where:

- `eletag`: Element tag
- `in`: First node of element `eletag`
- `jn`: Second node of element `eletag`
- `ndofpnlocal`: Integer number of degrees of freedom in local reference of element `eletag`
- $mattag_i$: Integer number identifying material of ith d.o.f in local reference for element `eletag`
- *angle*: Angle (in radians) between local reference and global reference of element `eletag`

1.4.5 Zero-length 2D section element (Sec. ??)

This is the counterpart of the zero length element for layered/fiber sections. It is particularly recommended if the center of rotation in zero-length element changes with axial force and moment.

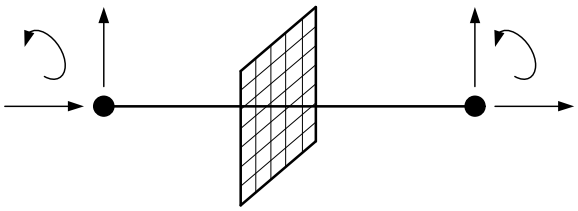

Figure 1.2: Zero-length 2D section element

```
eletag, 'ZeroLength2DSection', in, jn, { sectag, angle }
```

Where:

- `eletag`: Element tag
- `in`: First node of element `eletag`
- `jn`: Second node of element `eletag`
- `sectag`: Integer number identifying section of element `eletag`
- *angle*: Angle (in radians) between local reference and global reference of element `eletag` - Unit is radian.

For example, if the structure has `StiffnessBased2DBeamColumn`, `FlexibilityBased2DBeamColumn`, and `ZeroLength2DSection` elements,

```
elements  = {  1, 'ZeroLength2DSection', 1, 2, {1, 0}              ;
           = {  2, 'StiffnessBased2DBeamColumn', 2, 3, {1, 3}     ;
           = {  3, 'FlexibilityBased2DBeamColumn', 3, 4, {2, 5}   ;
           = {  4, 'ZeroLength2DSection', 4, 5, {2, 0}            }
```

1.5 Section Block

Section block declares section properties. `sections` defines section types, section properties, and basic material information of section. Description on sections may be different depending on types of section.

```
sections  = {  sectag1, sectype1, { SecProp1 }   ;
               ...                               ;
               sectagi, sectypei, { SecPropi }   ;
               ...                               ;
               sectagn, sectypen, { SecPropn }   }
```

Where:

- $\texttt{sectag}_i$: Sequential integer number identifying section at i^{th} section
- $\texttt{sectype}_i$: Section type at i^{th} section
- $\texttt{SecProp}_i$: Section properties and basic material information on i^{th} section

1.5.1 General section

General section has only one layer or fiber, and it in `Mercury` only supports elastic material currently. Usually, if section has nonlinear material, user may use multi-layer or multi-fiber section.

```
sectag, 'General', { mattag, A, Ix, Iy, Iz }
```

Where:

- `sectag`: Section tag
- `mattag`: Integer number identifying material with `sectag`
- `A`: Section area with `sectag`
- `Ixx`: Moment inertia on x-axis with `sectag`
- `Iyy`: Moment inertia on y-axis with `sectag`
- `Izz`: Moment inertia on z-axis with `sectag`

1.5.2 Layer section (Sec. ??)

All elements in Mercury can be layered.

```
sectag, 'Layer', { mattag_1, A_1, y-distance_1 ;
                                ... ;
                   mattag_i, A_i, y-distance_i ;
                                ... ;
                   mattag_n, A_n, y-distance_n }
```

Where:

- `sectag`: Section tag
- `mattag`$_i$: Integer number identifying material of i^{th} layer in section `sectag`
- `A`$_i$: i^{th} layer section area in section `sectag`
- `y-distance`$_i$: i^{th} layer distance from neutral axis to centroid of i^{th} layer along y-axis in section with `sectag`

1.5.3 Fiber section (Sec. ??)

All elements in Mercury can have fiber sections.

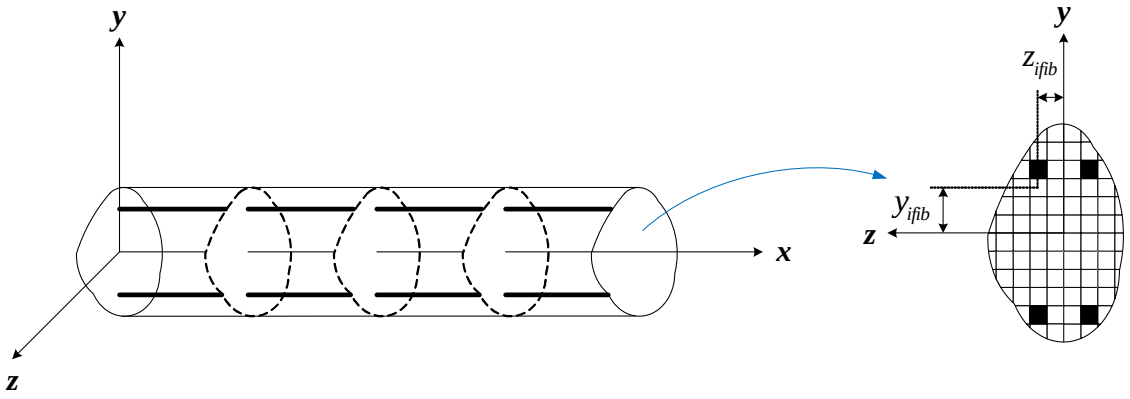

Figure 1.3: Fiber sections

```
sectag, 'Fiber', { mattag_1, A_1, y-distance_1, z-distance_1 ;
                                ... ;
                   mattag_i, A_i, y-distance_i, z-distance_i ;
                                ... ;
                   mattag_n, A_n, y-distance_n, z-distance_n }
```

Where:

- `sectag`: Section tag
- `mattag`$_i$: Integer number identifying material of i^{th} fiber in section `sectag`
- `A`$_i$: i^{th} fiber section area in section `sectag`
- `y-distance`$_i$: i^{th} fiber distance from section centroid to centroid of i^{th} fiber along y-axis in section `sectag`
- `z-distance`$_i$: i^{th} fiber distance from neutral axis to centroid of i^{th} fiber along z-axis in section `sectag`

For example, if a structure has `General` and two `Layer` sections,

```
section= {1,'General', {1, 100, 0, 0, 833.33\}
          2, 'Layer',  2, 1, 0.35 ;
                       3, 1, 0.25 ;
                       2, 1, 0.15 ;
                       2, 1, 0.05 ;
                       2, 1, -0.05 ;
                       2, 1, -0.15 ;
                       3, 1, -0.25 ;
                       2, 1, -0.25 ;
         3, 'Layer',   3, 1, 0.35 ;
                       2, 1, 0.25 ;
                       3, 1, 0.15 ;
                       3, 1, 0.05 ;
                       3, 1, -0.05 ;
                       3, 1, -0.15 ;
                       2, 1, -0.25 ;
                       3, 1, -0.25}
```

1.6 Material Block

Material block declares material properties.

`materials`	= {	`mattag`$_1$, `mattype`$_1$, `modulus`$_1$, `density`$_1$, { `MatProp`$_1$ }	;
		...	;
		`mattag`$_i$, `mattype`$_i$, `modulus`$_i$, `density`$_i$, { `MatProp`$_i$ }	;
		...	;
		`mattag`$_n$, `mattype`$_n$, `modulus`$_n$, `density`$_n$, { `MatProp`$_n$ }	}

where:

- `mattag`$_i$: Consecutive integer number identifying material at i^{th} material
- `mattype`$_i$: Material type at i^{th} material
- `modulus`$_i$: Material modulus at i^{th} material
- `density`$_i$: Density at i^{th} material
- `MatProp`$_i$: Material properties at i^{th} material

1.6.1 Elastic material

`mattag`,	'Elastic',	`modulus`,	`density`,	{ `G` }

Where:

- `mattag`: Material tag
- `modulus`: Young's modulus of a material with `mattag`
- `density`: Density of a material with `mattag`
- `G`: Shear modulus of a material with `mattag`

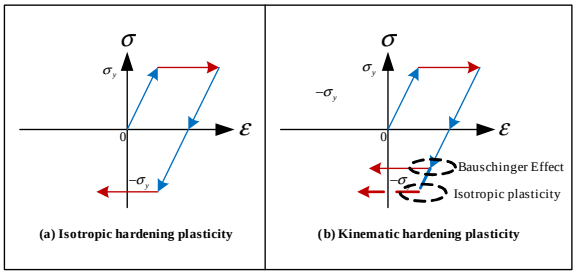

Figure 1.4: Zero-length 2D element

1.6.2 Hardening material (Sec. ??)

```
mattag, 'Hardening', modulus, density, { sigmaY0, Hiso, Hkin }
```

Where:

- `mattag`: Material tag
- `modulus`: Young's modulus of a material with `mattag`
- `density`: Density of a material with `mattag`
- `sigmaY0`: Initial yield stress of a material with `mattag`
- `Hiso`: Isotropic hardening modulus of a material with `mattag`
- `Hkin`: Kinematic hardening modulus of a material with `mattag`

1.6.3 Bilinear material with isotropic hardening (Sec. ??)

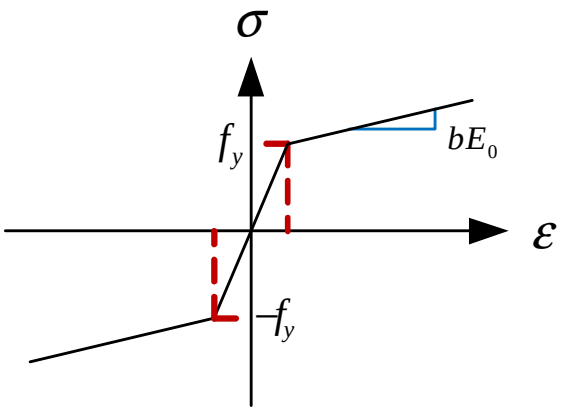

Figure 1.5: Bilinear material with isotropic hardening

```
mattag, 'Bilinear', modulus, density, { SigmaY0, b, a1, a2, a3, a4 }
```

Where

- `mattag`: Material tag
- `modulus`: Young's modulus of a material with `mattag`
- `density`: Density of a material with `mattag`
- `sigmaY0`: Initial yield stress of a material with `mattag`
- `b`: Strain-hardening ratio between post-yield tangent and Young's modulus of a material with `mattag`
- `a1`: Isotropic hardening coefficient 1 of a material with `mattag` - increase of compression yield envelope as proportion of initial yield stress after a plastic strain of `a2` × (`SigmaY0`/`modulus`)

- a2: Isotropic hardening coefficient 2 of a material with mattag
- a3: Isotropic hardening coefficient 3 of a material with mattag - increase of tension yield envelope as proportion of initial yield stress after a plastic strain of a4 × (SigmaY0/modulus)
- a4: Isotropic hardening coefficient 4 of a material with mattag

1.6.4 Modified Giuffre-Menegotto-Pinto material with isotropic hardening (Sec. ??)

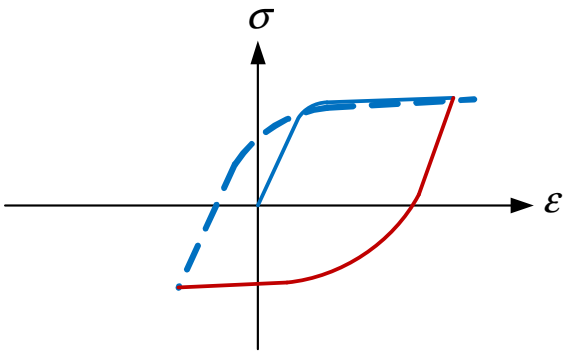

Figure 1.6: Modified Giuffre-Menegotto-Pinto material with isotropic hardening

```
mattag, 'GiuffreMenegottoPinto', modulus, density, { SigmaY0, b,
                          R0, n, cR1, cR2, a1, a2, a3, a4, σ_init }
```

Where:

- mattag: Material tag
- modulus: Young's modulus of a material with mattag
- density: Density of a material with mattag
- sigmaY0: Initial yield stress of a material with mattag
- b: Strain-hardening ratio between post-yield tangent and Young's modulus of a material with mattag
- R0: Coefficient 0 of a material with mattag to control the transition from elastic to plastic branches - value between 10 and 20 is recommended
- n: R = R0 − (R0^{n}) ∗ cR1 * ξ / (cR2 + ξ) - n can have 0 and 1.
- cR1: Coefficient 1 of a material with mattag to control the transition from elastic to plastic branches - 0.925 is recommended
- cR2: Coefficient 1 of a material with mattag to control the transition from elastic to plastic branches - 0.15 is recommended
- a1: Isotropic hardening coefficient 1 of a material with mattag - increase of compression yield envelope as proportion of initial yield stress after a plastic strain of a2 × (SigmaY0/modulus)
- a2: Isotropic hardening coefficient 2 of a material with mattag
- a3: Isotropic hardening coefficient 3 of a material with mattag - increase of tension yield envelope as proportion of initial yield stress after a plastic strain of a4 × (SigmaY0/modulus)
- a4: Isotropic hardening coefficient 4 of a material with mattag
- σ_{init}: Initial stress of a material with mattag

1.6.5 Anisotropic damage 1D material (Sec. ??)

```
mattag, 'AnisotropicDamage1D', modulus, density,
                  { ν, κ_0, Adamage, adamage, Dc }
```

Where:

- mattag: Material tag
- modulus: Young's modulus of a material with mattag
- density: Density of a material with mattag

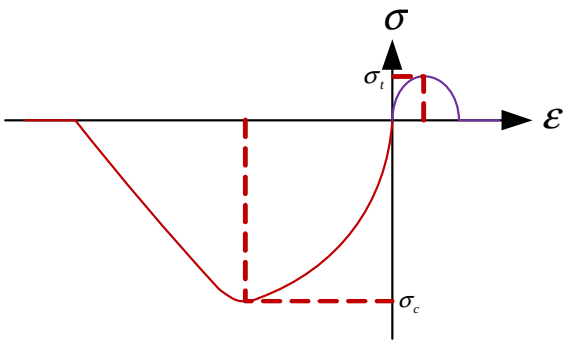

Figure 1.7: Concrete 1D anisotropic damage model

- ν: Poisson's ratio of a material with `mattag`
- κ_0: Initial elasticity threshold of a material with `mattag`
- `Adamage`: Damage coefficient A of a material with `mattag`
- `adamage`: Damage coefficient a of a material with `mattag`
- `Dc`: Damage limit of a material with `mattag`

1.6.6 Modified Kent and Park model (Sec. ??)

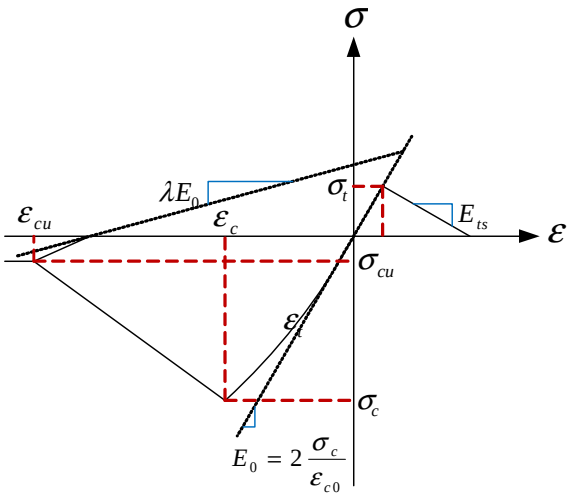

Figure 1.8: Concrete tension linear softening model

`mattag`, '`ConcreteLinearTensionSoftening`', `modulus`, `density`, $\{\ \sigma_c,\ \varepsilon_c,\ \sigma_{cu},\ \varepsilon_{cu},\ \lambda,\ \sigma_t\ \}$

Where:

- `mattag`: Material tag
- `modulus`: Tension softening stiffness(absolute value) - slope of the linear tension softening branch of a material with `mattag`
- `density`: Density of a material with `mattag`
- σ_c: Compressive yield stress of a material with `mattag` - **Negative value**
- ε_c: Compressive yield strain of a material with `mattag` - **Negative value**
- σ_{cu}: Compressive crushing stress of a material with `mattag` - **Negative value**
- ε_{cu}: Compressive crushing strain of a material with `mattag` - **Negative value**

- λ: Ratio between unloading slope at ε_c and slope Young's modulus of a material with `mattag`
- σ_t: Tensile yield stress of a material with `mattag`

For example, if the structure for static analysis has `ConcreteLinearTensionSoftening` and `Bilinear` materials,

```
materials  = {  1, 'ConcreteLinearTensionSoftening', 10, 0,    {-29, -0.00221, -5.8,
                                                                -0.02, 0.027719, 2.9}   ;
                2, 'Bilinear', 210, 0, {290, 0.01, 0.0, 1, 0, 1}                       }
```

1.7 Force Block

Force block declares all the external forces on the structure.

```
forces  = {  forcetype_1,  { ForceProp_1 }   ;
             ...                             ;
             forcetype_i,  { ForceProp_i }   ;
             ...                             ;
             forcetype_n,  { ForceProp_n }   }
```

Where:

- `forcetype`$_i$: is type of force [`NodalForce| NodalDisplacement| ElementDistributedForce| VariableNodalForce|VariableNod VariableElementDistributedForce| GreoundAcceleration| Variable|GroundAcceleration`] Force type of i^{th}
- `ForceProp`$_i$: Force properties of i^{th} `forcetype`

1.7.1 Nodal force

```
'NodalForce',  {  nodtag_1,  gx_1,  magnitude_1   ;
                              ...                 ;
                  nodtag_i,  gx_i,  magnitude_i   ;
                              ...                 ;
                  nodtag_n,  gx_n,  magnitude_n   }
```

Where

- `nodtag`$_i$: Node tag of i^{th} nodal force
- `gx`$_i$: Direction on global external force at node, [1—2—3] for X, or Y or Z global axis.
- `magnitude`$_i$: Magnitude of external force with node tag at i^{th} nodal force

1.7.2 Nodal displacement

```
'NodalDisplacement',  {  nodtag_1,  gx_1,  magnitude_1   ;
                                   ...                   ;
                         nodtag_i,  gx_i,  magnitude_i   ;
                                   ...                   ;
                         nodtag_n,  gx_n,  magnitude_n   }
```

Where:

- `nodtag`$_i$: Node tag of i^{th} nodal displacement
- `gx`$_i$: Direction on global reference of external displacement with node tag at i^{th} nodal displacement; [1—2—3] for X, or Y or Z global axis.
- `magnitude`$_i$: Magnitude of external displacement with node tag at i^{th} nodal displacement

1.7.3 Element distributed force

```
'ElementDistributedForce',  {  eletag_1,  lx_1,  magnitude_1   ;
                                          ...                  ;
                               eletag_i,  lx_i,  magnitude_i   ;
                                          ...                  ;
                               eletag_n,  lx_n,  magnitude_n   }
```

Where:

- $\texttt{eletag}_i$: Element tag at i^{th} element distributed force
- $\texttt{lx}_i$: Direction on local reference of external force with element tag at i^{th} element distributed force; [1—2—3] for X, or Y or Z global axis.
- $\texttt{magnitude}_i$: Magnitude of external force with element tag at i^{th} element distributed force

1.7.4 Variabel nodal force

$$
\begin{array}{llll}
\texttt{'VariableNodalForce'}, & \{ & \texttt{nodtag}_1,\ \texttt{gx}_1,\ \{\ \texttt{m}_1^1,\ \cdots,\ \texttt{m}_1^j,\ \cdots,\ \texttt{m}_1^m\ \} & ; \\
 & & \cdots & ; \\
 & & \texttt{nodtag}_i,\ \texttt{gx}_i,\ \{\ \texttt{m}_i^1,\ \cdots,\ \texttt{m}_i^j,\ \cdots,\ \texttt{m}_i^m\ \} & ; \\
 & & \cdots & ; \\
 & & \texttt{nodtag}_n,\ \texttt{gx}_n,\ \{\ \texttt{m}_n^1,\ \cdots,\ \texttt{m}_n^j,\ \cdots,\ \texttt{m}_n^m\ \} & \}
\end{array}
$$

Where:

- $\texttt{nodtag}_i$: Node tag for i^{th} nodal force
- $\texttt{gx}_i$: Direction on global reference of external force with node tag at i^{th} nodal force; [1—2—3] for X, or Y or Z global axis.
- $\texttt{m}_i^j$: Magnitude of external force associated with node tag at i^{th} nodal force and j^{th} external force step

1.7.5 Variable nodal displacement

$$
\begin{array}{llll}
\texttt{'VariableNodalDisplacement'}, & \{ & \texttt{nodtag}_1,\ \texttt{gx}_1,\ \{\ \texttt{m}_1^1,\ \cdots,\ \texttt{m}_1^j,\ \cdots,\ \texttt{m}_1^m\ \} & ; \\
 & & \cdots & ; \\
 & & \texttt{nodtag}_i,\ \texttt{gx}_i,\ \{\ \texttt{m}_i^1,\ \cdots,\ \texttt{m}_i^j,\ \cdots,\ \texttt{m}_i^m\ \} & ; \\
 & & \cdots & ; \\
 & & \texttt{nodtag}_n,\ \texttt{gx}_n,\ \{\ \texttt{m}_n^1,\ \cdots,\ \texttt{m}_n^j,\ \cdots,\ \texttt{m}_n^m\ \} & \}
\end{array}
$$

Where:

- $\texttt{nodtag}_i$: Node tag at i^{th} nodal displacement
- $\texttt{gx}_i$: Direction on global reference of external displacement with node tag at i^{th} nodal displacement; [1—2—3] for X, or Y or Z global axis.
- $\texttt{m}_i^j$: Magnitude of external displacement with node tag at i^{th} nodal displacement and j^{th} external displacement step

1.7.6 Variable element distributed force

$$
\begin{array}{llll}
\texttt{'VariableElementDistributedForce'}, & \{ & \texttt{eletag}_1,\ \texttt{lx}_1,\ \{\ \texttt{m}_1^1,\ \cdots,\ \texttt{m}_1^j,\ \cdots,\ \texttt{m}_1^m\ \} & ; \\
 & & \cdots & ; \\
 & & \texttt{eletag}_i,\ \texttt{lx}_i,\ \{\ \texttt{m}_i^1,\ \cdots,\ \texttt{m}_i^j,\ \cdots,\ \texttt{m}_i^m\ \} & ; \\
 & & \cdots & ; \\
 & & \texttt{eletag}_n,\ \texttt{lx}_n,\ \{\ \texttt{m}_n^1,\ \cdots,\ \texttt{m}_n^j,\ \cdots,\ \texttt{m}_n^m\ \} & \}
\end{array}
$$

Where

- $\texttt{eletag}_i$: Element tag at i^{th} element distributed force
- $\texttt{lx}_i$: Direction on local reference of external force with element tag at i^{th} element distributed force; [1—2—3] for X, or Y or Z global axis.
- $\texttt{m}_i^j$: Magnitude of external force with element tag at i^{th} element distributed force and j^{th} external element distributed force step

1.7.7 Ground acceleration

```
'GroundAcceleration',  { factor, timestep,  { m_1^x, m_1^y [ m_1^z] }  ;
                                                  ...                  ;
                                             { m_i^x, m_i^y [ m_i^z] }  ;
                                                  ...                  ;
                                             { m_n^x, m_n^y [ m_n^z] }  }
```

Where:

- `factor`: Factor on gravity acceleration
- `timestep`: Consistent time step
- $\mathtt{m}_i^X$: Magnitude of ground acceleration along X-axis at i^{th} step
- $\mathtt{m}_i^Y$: Magnitude of ground acceleration along Y-axis at i^{th} step
- $\mathtt{m}_i^Z$: Magnitude of ground acceleration along Z-axis at i^{th} step

1.7.8 Variable ground acceleration

```
'VariableGroundAcceleration',  { factor,  { time_1, m_1^X, m_1^Y [ m_1^Z] }  ;
                                                  ...                      ;
                                           { time_i, m_i^X, m_i^Y [ m_i^Z] }  ;
                                                  ...                      ;
                                           { time_n, m_n^X, m_n^Y [ m_n^Z] }  }
```

Where:

- `factor`: Factor on gravity acceleration
- $\mathtt{time}_i$: Time at i^{th} step
- $\mathtt{m}_i^X$: Magnitude on ground acceleration along x-axis at i^{th} step
- $\mathtt{m}_i^Y$: Magnitude on ground acceleration along y-axis at i^{th} step
- $\mathtt{m}_i^Z$: Magnitude on ground acceleration along z-axis at i^{th} step

For example if the structure has `NodalForce` and `VariableNodalForce` forces to node 3 along x-axis,

```
forces  = {  'NodalForce', {3, 1, 50}                                          ;
             'VariableNodalForce', {3, 1, {0, 50, 100, 150, 200, 250, 300, 350}}  }
```

1.8 Output Block

The `OutputData` command is file recorder for output data.

```
OutputData  = {  OutputType_1, Filename_1, { Info_1 }  ;
                 ...                                   ;
                 OutputType_i, Filename_i, { Info_i }  ;
                 ...                                   ;
                 OutputType_n, Filename_n, { Info_n }  }
```

Where

- $\mathtt{Outputtype}_i$: is [`NodalDisplacement| NodalVelocity| NodalAcceleration| NodalForce| SectionAxialForce| SectionAxialDeformation| SectionMoment| SectionCurvature| UniaxialStressStrain|`].
- $\mathtt{Filename}_i$: User define

1.8.1 Nodal displacement

```
'NodalDisplacement', 'Filename', { nodtag_1, ···, nodtag_i, ···, nodtag_n }
```

Where:

- $\mathtt{nodtag}_i$: Node tag at i^{th} node

1.8.2 Nodal velocity

'NodalVelocity', 'Filename', { $\texttt{nodtag}_1, \cdots, \texttt{nodtag}_i, \cdots, \texttt{nodtag}_n$ }

Where:

- $\texttt{nodtag}_i$: Node tag at i^{th} node

1.8.3 Nodal acceleration

'NodalAcceleration', 'Filename', { $\texttt{nodtag}_1, \cdots, \texttt{nodtag}_i, \cdots, \texttt{nodtag}_n$ }

Where:

- $\texttt{nodtag}_i$: Node tag at i^{th} node

1.8.4 Nodal force

'NodalForce', 'Filename', { $\texttt{nodtag}_1, \cdots, \texttt{nodtag}_i, \cdots, \texttt{nodtag}_n$ }

Where:

- $\texttt{nodtag}_i$: Node tag at i^{th} node

1.8.5 Section axial force

'SectionAxialForce', 'Filename', { $\texttt{secnum}_1, \texttt{eletag}_1$;
...;
$\texttt{secnum}_i, \texttt{eletag}_i$;
...;
$\texttt{secnum}_n, \texttt{eletag}_n$; }

Where:

- $\texttt{secnum}_i$: Section of $\texttt{secnum}^{th}$ integration point in an element with eletag
- $\texttt{eletag}_i$: Element tag at i^{th} element

1.8.6 Section axial deformation

'SectionAxialDeformation', 'Filename', { $\texttt{secnum}_1, \texttt{eletag}_1$;
...;
$\texttt{secnum}_i, \texttt{eletag}_i$;
...;
$\texttt{secnum}_n, \texttt{eletag}_n$; }

Where:

- $\texttt{secnum}_i$: Section of $\texttt{secnum}^{th}$ integration point in an element with eletag
- $\texttt{eletag}_i$: Element tag at i^{th} element

1.8.7 Section moment

'SectionMoment', 'Filename', { $\texttt{secnum}_1, \texttt{eletag}_1$;
...;
$\texttt{secnum}_i, \texttt{eletag}_i$;
...;
$\texttt{secnum}_n, \texttt{eletag}_n$; }

Where:

- $\texttt{secnum}_i$: Section of $\texttt{secnum}^{th}$ integration point in an element with eletag
- $\texttt{eletag}_i$: Element tag at i^{th} element

1.8.8 Section curvature

'SectionCurvature',	'Filename',	{ secnum$_1$, eletag$_1$	;
		...	;
		secnum$_i$, eletag$_i$	;
		...	;
		secnum$_n$, eletag$_n$	; }

Where:

- secnum$_i$: Section of secnumth integration point in an element with eletag
- eletag$_i$: Element tag at i^{th} element

1.8.9 Uniaxial stress and strain

'UniaxialStressStrain',	'Filename',	{ fibernum$_1$, secnum$_1$, eletag$_1$	;
		...	;
		fibernum$_i$, secnum$_i$, eletag$_i$	;
		...	;
		fibernum$_n$, secnum$_n$, eletag$_n$	; }

Where:

- fibernum$_i$: $fibernum^{th}$ layer/fiber with secnum and eletag
- secnum$_i$: Section of secnumth integration point in an element with eletag
- eletag$_i$: Element tag at i^{th} element

Chapter 2

C++ Version

This document[1] describes the input for C++ version of Mercury. The C++ version uses the Lua scripting language[2] (analogous to TCL in OpenSees).

2.1 nodes

The "nodes" command defines nodal coordinates, and nodal masses if material densities are not in materials.

`nodes`	=	{	{	$\texttt{nodtag}_1$, x_1, y_1 [, z_1] [, 'mass', mx_1, my_1 [, mz_1]]	};
			{	...	};
			{	$\texttt{nodtag}_i$, x_i, y_i [, z_i] [, 'mass', mx_i, my_i [, mz_i]]	};
			{	...	};
			{	$\texttt{nodtag}_n$, x_n, y_n [, z_n] [, 'mass', mx_n, my_n [, mz_n]]	} };

- $\texttt{nodtag}_i$: Tag of the i^{th} node
- x_i, y_i, and z_i are node coordinates of node i at each global coordinate.
- mx_i, my_i, and mz_i are mass quantities of node i at each global coordinate.

2.2 elements

The "elements" command defines element type, nodal connectivity, and basic section information. These may vary with the element type.

`elements`	=	{	{	Definition of element_1	};
			{	...	};
			{	Definition of element_i	};
			{	...	};
			{	Definition of element_n	} };

2.2.1 Simple 2D truss element

The simple 2D truss element is the classical two noded axial element. It is defined in **Definition of element**.

`eletag`, 'Simple2DTruss', `sn`, `en`, `Area`, `sectag`

- `eletag`: Tag of element
- `sn`: Start node of node connectivity in element
- `en`: End node of node connectivity in element
- `Area`: The cross sectional area in element
- `sectag`: Tag of section describing section properties in element

[1] In this preliminary version of Mercury, no attempt has been made to simplify (generate/automate) data entry, and there is not (yet) a mesh generator for the program. Those are simple future developments.

[2] http://www.lua.org/

2.2.2 Stiffness-based 2D beam-column element

The stiffness-based 2D beam-column element can have a constant, layered, or fiber section. Its numerical integration is based on Gauss-Legendre quadrature rule and it is defined in **Definition of element**.

`eletag`, '`StiffnessBased2DBeamColumn`', `sn`, `en`, { `sectag`, `nIp` }

- `nIp`: Number of integration points following Gauss-Legendre quadrature rule of element with `eletag`

2.2.3 Flexibility-based 2D beam-column element

The flexibility-based 2D beam-column element can have a constant, layered, or fiber section. Its numerical integration is based on Gauss-Lobatto quadrature rule and it is defined in **Definition of element**.

`eletag`, '`FlexibilityBased2DBeamColumn`', `sn`, `en`, { `sectag`, `nIp` }, {`nIter`, `tol`}

- `nIp`: Number of integration points following Gauss-Lobatto quadrature rule of element with `eletag`
- `nIter`: Number of iterations to determine the element internal nodal forces in element state determination
- `tol`: Convergence tolerance

2.2.4 Interface element

The interface element is divided into two parts; (a) Zero length 2D element is used to model lumped plasticity. It can account for stiffness degradation in flexure and shear. It neglects axial-flexural coupling effect and depends on force and deformation history as well as on the section characteristics. (b) Zero length 2D section element is the counterpart of the zero length 2D element for layered/fiber sections. It is particularly recommended if the center of rotation in zero-length element changes with axial force and moment.

`eletag`, '`InterfaceElement2D`', `sn`, `en`, `Definition of zero length 2D element`,
`Defintion of zero length 2D sectio element`

2.2.4.1 Zero length 2D element

Zero length 2D element is defined in **Definition of zero length 2D element**.

{ { $\texttt{mattag}_1$, { $active_X$, $active_Y$, $active_Z$ } ;	
{ $\texttt{mattag}_2$, { $active_X$, $active_Y$, $active_Z$ } ;	: if in a structure
{ $\texttt{mattag}_3$, { $active_X$, $active_Y$, $active_Z$ } }	
{ }	: if not in a structure

- mattag_i: Tag of uniaxiai material to represent force-deformation relationship for the element on the X, Y, or Z global coordinate.
- $active_X$, $active_Y$, or $active_Z$: If uniaxial meterial is defined, then it is 1, and not 0 on global coordinate.

2.2.4.2 Zero length 2D section element

Zero length 2D section element is defined in **Definition of zero length 2D section element**.

{ { sectag } }, {$xv1$, $xv2$, $xv3$},{$yv1$, $yv2$, $yv3$}	: if in a structure
{ }	: if not in a structure

- sectag: Tag of layered/fiber section
- $xv1$, $xv2$, and $xv3$: Vector component in global coordinates defining local x-axis
- $yv1$, $yv2$, and $yv3$: Vector component in global coordinates defining local y-axis
- The local z-axis is defined by the cross product between the vector xv and yv.

2.3 sections

The "sections" defines section types, section properties, and basic material information of section. Description on sections may be different depending on types of section.

```
sections = { Definition of section_1 ;
                        ...;
             Definition of section_i ;
                        ...;
             Definition of section_n };
```

2.3.1 General section

General section has only one layer or fiber, and it in `Mercury` only supports elastic material currently. Usually, if section has nonlinear material, user may use multi-layer or multi-fiber section. It is defined in **Definition of section**.

```
sectag, 'General', { mattag, Area, Iz }
```

- `sectag`: Tag of section
- `mattag`: Tag of material in section with `sectag`
- `Area`: The cross sectional area in section with `sectag`
- `Izz`: Moment inertia on z-axis in section with `sectag`

2.3.2 Layered section

Layered section is defined in **Definition of section**.

```
sectag, 'Layer', { mattag_1, Area_1, y-distance_1   ;
                                ...                 ;
                   mattag_i, Area_i, y-distance_i   ;
                                ...                 ;
                   mattag_n, Area_n, y-distance_n   }
```

- `mattag`$_i$: Tag of material of i^{th} layer in section with `sectag`
- `Area`$_i$: The cross sectional area of the i^{th} layer in section with `sectag`
- `y-distance`$_i$: The distance of the i^{th} layer from neutral axis to centroid of i^{th} layer along y-axis in section with `sectag`

2.3.3 Fiber section

Fiber section is defined in **Definition of section**.

```
sectag, 'Fiber', { mattag_1, Area_1, y-distance_1, z-distance_1   ;
                                ...                               ;
                   mattag_i, Area_i, y-distance_i, z-distance_i   ;
                                ...                               ;
                   mattag_n, Area_n, y-distance_n, z-distance_n   }
```

- `mattag`$_i$: Tag of material of i^{th} fiber in section with `sectag`
- `Area`$_i$: The cross sectional area of i^{th} fiber in section with `sectag`
- `y-distance`$_i$: The distance of the i^{th} fiber from neutral axis to centroid of i^{th} fiber along y-axis in section with `sectag`
- `z-distance`$_i$: The distance of the i^{th} fiber from neutral axis to centroid of i^{th} fiber along y-axis in section with `sectag`

2.4 materials

The "materials" command declares material properties associated with element and section.

`elements`	=	{	{ Definition of material$_1$	};
			{ ...	};
			{ Definition of material$_i$	};
			{ ...	};
			{ Definition of material$_n$	} };

2.4.1 Elastic model

Elastic model is defined in **Definition of material**.

`mattag`, '`elastic`', `E`, ρ

- `mattag`: Tag of material
- `E`: Elastic modulus in material with `mattag`
- ρ: Density in material with `mattag`

2.4.2 Hardening model

Hardening model is defined in **Definition of material**.

`mattag`, '`hardening`', `E`, ρ, σ_{y0}, `Hiso`, `Hkin`

- σ_{y0}: Initial yield stress in material with `mattag`
- `Hiso`: Isotropic hardening modulus in material with `mattag`
- `Hkin`: Kinematic hardening modulus in material with `mattag`

2.4.3 Bilinear model with isotropic hardening

Bilinear model with isotropic hardening is defined in **Definition of material**.

`mattag`, '`bilinear`', `E`, ρ, σ_{y0}, `b`, `a`$_1$, `a`$_2$, `a`$_3$, `a`$_4$

- `b`: Strain-hardening ratio between post-yield tangent and elastic (Young's) modulus in material with `mattag`
- `a1`: Isotropic hardening coefficient 1 in material with `mattag` - increase of compression yield envelope as proportion of initial yield stress after a plastic strain of `a2` $\times$ (σ_{y0}/`E`)
- `a2`: Isotropic hardening coefficient 2 in material with `mattag`
- `a3`: Isotropic hardening coefficient 3 in material with `mattag` - increase of tension yield envelope as proportion of initial yield stress after a plastic strain of `a4` $\times$ (σ_{y0}/`E`)
- `a4`: Isotropic hardening coefficient 4 in material with `mattag`

2.4.4 Modified Giuffre-Menegotto-Pinto model with isotropic hardening

Modified Giuffre-Menegotto-Pinto model with isotropic hardening is defined in
Definition of material.

---

mattag, 'modifiedgmpsteel', E, ρ, σ_{y0}, b, R0, cR1, cR2, a_1, a_2, a_3, a_4, σ_{init}

- R0: Coefficient 0 in material with mattag to control the transition from elastic to plastic branches - value between 10 and 20 is recommended
- cR1: Coefficient 1 in material with mattag to control the transition from elastic to plastic branches - 0.925 is recommended
- cR2: Coefficient 2 in material with mattag to control the transition from elastic to plastic branches - 0.15 is recommended
- σ_{init}: Initial stress in material with mattag

2.4.5 Modified Kent-Park model

Modified Kent-Park model is defined in **Definition of material**.

---

mattag, 'modifiedkpconcrete', E_t, ρ, σ_c, ε_c, σ_{cu}, ε_{cu}, λ, σ_t

---

- E_t: Tesion softening stiffness(absolute value) - slope of the linear tension softening branch in material with mattag
- σ_c: Compressive yield stress in material with mattag - **Negative value**
- ε_c: Compressive yield strain in material with mattag - **Negative value**
- σ_{cu}: Compressive crushing stress in material with mattag - **Negative value**
- ε_{cu}: Compressive crushing strain in material with mattag - **Negative value**
- λ: Ratio between unloading slope at ε_c and slope Young's modulus in material with mattag
- σ_t: Tensile yield stress in material with mattag

2.4.6 Anisotropic damage 1D model without permanent strain

Anisotropic damage 1D model without permanent strain is defined in
Definition of material.

---

mattag, 'anisotropicdamage', E, ρ, ν, κ_0, A_D, a_D, Dc

- ν: Poisson's ratio in material with mattag
- κ_0: Initial elasticity threshold in material with mattag
- A_D: Damage coefficient A in material with mattag
- a_D: Damage coefficient a in material with mattag
- Dc: Damage limit in material with mattag

2.4.7 Anisotropic damage 1D model with permanent strain

Anisotropic damage 1D model with permanent strain is defined in **Definition of material**.

---

mattag, 'anisotropicdamage2', E, ρ, ν, κ_0, A_D, k_h, k_d, K_h, K_d, Dc

---

- k_h: Compression hydrostatic permanent strain in material with mattag
- k_d: Compression deviatoric permanent strain in material with mattag
- K_h: Tension hydrostatic permanent strain in material with mattag
- K_d: Tension deviatoric permanent strain in material with mattag

2.4.8 Linear viscous damping model

Linear viscous damping model is defined in **Definition of material**.

---

mattag, 'linearviscous', CK_d

- C_d: Viscous damping modulus

2.5 Assign model to Mercury C++ version

2.5.1 Declare a model

The spatial dimension of the structure, and the number of degrees of freedom per node are assigned to model.

```
model = StructureModel(ndim,  ndofpn)
```

- ndim: The spatial dimension of the structure
- ndofpn: The number of degrees of freedom per node

2.5.2 Add the information on structure to the model

The information on structure (nodes, materials, sections, and elements) is added to the model

---

```
model:addNodes(nodes)
model:addMaterials(materials)
model:addSections(sections)
model:addElements(elements)
```

2.5.3 Define boundary condition of the model

```
model:constrainNode(nodtag1 , idX1 , idY1  [, idZ1])
                        ...
 model:constrainNode(nodtagi , idXi , idYi  [, idZi])
                        ...
model:constrainNode(nodtagn , idXn , idYn  [, idZn])
```

- idXi , idYi [idZi]): Describe the boundary condition on global coordinates where 0 corresponds to a free dof, and 1 to a fixed one.

2.6 External load

To apply the external load of structure, function `LoadDescription()` is used.

```
Load_Name = LoadDescription()
Load_Name:addLoad( Definition of the external load_1 )
...
Load_Name:addLoad( Definition of the external load_i )
...
Load_Name:addLoad( Definition of the external load_n )
```

- `Load_Name`: User defined name for load description

2.6.1 Static nodal load

Static nodal loads are assigned in **Definition of the external load**.

```
{ 'staticnodalload', nodtag, dof_index, m }
```

- `nodtag`: Tag of node with static nodal load
- `dof_index`: Indicate the direction of static nodal load with `nodtag`, where 1 corresponds to global X-axis, 2 to global Y-axis, and 3 to global Z axis for rotation.
- `m`: Load amplitude of node with `nodtag` on global coordinates

2.6.2 Static global load

Static global loads are applied to all nodes and assigned in **Definition of the external load**.

```
{ 'staticglobalload', dof_index, m }
```

- `m`: Load amplitude applied to dof_index of all nodes on global coordinates

2.6.3 Incremental nodal load

Incremental nodal loads are assigned in in **Definition of the external load**.

```
{ 'incrementalnodalload', nodtag, dof_index, m_1, ..., m_i, ..., m_n }
```

- $\mathtt{m}_i$: The i^{th} load amplitude of node with `nodtag` on global coordinates

2.6.4 Incremental nodal displacement

Incremental nodal displacements are assigned in **Definition of the external load**.

```
{ 'incrementalnodaldisplacement', nodtag, dof_index, m_1, ..., m_i, ..., m_n }
```

- $\mathtt{m}_i$: The i^{th} displacement amplitude of node with `nodtag` on global coordinates

2.6.5 Time series nodal load

Time series nodal loads are assigned in **Definition of the external load**.

```
{ 'timeseriesnodalload', 'data file name', nodtag, dof_index, sacale }
```

- `data file name`: The imported data file name
- `scale`: Scale factor for data file

2.6.6 Time series nodal acceleration

Time series nodal accelerations are assigned in **Definition of the external load**.

```
{ 'timeseriesnodalaccel', 'data file name, dt = time, delay = time_delay',
nodtag_1, dof_index_1, gravity_1, ..., nodtag_i, dof_index_i, gravity_i, ...,
nodtag_n, dof_index_n, gravity_n }
```

- `time`: The time interval between previous and current time
- time_delay: Delayed time to apply time series nodal acceleration in time series.
- $\texttt{nodtag}_i$: Tag of the i^{th} node with time series nodal acceleration
- $\texttt{dof\_index}_i$: Indicate the direction of acceleration with $\texttt{nodtag}_i$, where 1 corresponds to global X-axis, 2 to global Y-axis, and 3 to global Z axis for rotation.
- $\texttt{gravity}_i$: Ground acceleration of node with $\texttt{nodtag}_i$

2.6.7 Time series global acceleration

Time series global accelerations are applied to the all nodes and assigned in
Definition of the external load.

```
{ 'timeseriesglobalaccel', 'data file name, dt = time,
delay = time_delay', dof_index, gravity }
```

- time_delay: Delayed time to apply time series global acceleration in time series.
- `gravity`: Ground acceleration

2.6.8 Global gravity

Global gravity indicates the self-weight of the structure with nodal masses associated with `'mass'` in `nodes` or ρ in `materials` and is assigned in **Definition of the external load**.

```
{ 'globalgravity', dof_index, gravity }
```

- `gravity`: Ground acceleration
- Self-weight is determined with $nodalmasses \times gravity$.

2.6.9 Ground motion

Global gravity indicates the self-weight of the structure with nodal masses associated with `'mass'` in `nodes` or ρ in `materials` and is assigned in **Definition of the external load**.

```
{ 'groundmotion', 'data file name, dt = time, delay = time_delay',
dof_index, gravity }
```

- time_delay: Delayed time to apply groud motion in time series.

2.7 Solver

Assign the nonlinear solver used to analysis. If not defining solver, then Mercury C++ version implements linear elastic analysis.

```
solver = NonlinearSolver(Definition of nonlinear solver)
```

2.7.1 Newton-Raphson iterative solver

The Newton-Raphson iterative solver is defined in **Definition of nonlinear solver**. Depending on tolerance criterion, there are three types. How can we distinguish tolerance criterion from description in command?

```
"newtonraphson", displacementdeltatolerance = tol, iterations= niter
"newtonraphson", residualforcedeltatolerance = tol, iterations= niter
"newtonraphson", energytolerance = tol, iterations= niter
```

- `tol`: Convergence tolerance
- `niter`: Maximum number of iterations

2.7.2 Initial stiffness iterative solver

The initial stiffness iterative solver is defined in **Definition of nonlinear solver**. Depending on tolerance criterion, there are three types. How can we distinguish tolerance criterion from description in command?

```
"initialstiffness", displacementdeltatolerance = tol, iterations= niter
"initialstiffness", residualforcedeltatolerance = tol, iterations= niter
"initialstiffness", energytolerance = tol, iterations= niter
```

2.7.3 Multiple iterative solver

Mercury C++ version can have multiple solver to switch solver if having convergence problem. The multiple iterative solver is defined in **Definition of nonlinear solver**. Why do we need two tolerance?

```
"multisolver", "newtonraphson", tol1, tol2, niter,
"initialstiffness", tol1, tol2, niter
```

- $\texttt{tol}_1$: Convergence tolerance for residual displacements
- $\texttt{tol}_2$: Convergence tolerance for residual forces

2.7.4 Newton Trust Region solver

What is newton trust region solver? Is it newton quasi line search method? Can you give me the material on this method?

2.8 Analysis

Analysis is divided into two catagories; (a) Static analysis, (b) Transient analysis.

2.8.1 Static analysis

Mercury C++ version has below description for static analysis.

```
analysis = StaticAnalysis()          : For linear elastic analysis
analysis = StaticAnalysis(solver)    : For nonlinear analysis
analysis:setStructureModel(model)
```

2.8.2 Transient analysis

Mercur C++ version supports three types of transient analysis, Newmark β method, HHT method, and Shing method.

For Newmark β method

```
transientanalysis =
DynamicAnalysis("newmark", model, solver, Load_Name, dt, beta,gamma)
```

For HHT method

```
transientanalysis =
DynamicAnalysis("hht", model, solver, Load_Name, dt, alpha, beta,gamma)
```

Shing method

```
transientanalysis =
DynamicAnalysis("hybridhht", model, solver, Load_Name, dt, alpha, beta,
gamma, niter)
```

```
model:setRayleighCoefficients(am, bk)
transientanalysis:solve(nstep)
```

- `Load_Name`: User defined name for load description
- `dt`: Time interval between previous and current time
- `alpha`: α coefficient in HHT method
- `beta` and `gamma`: β and γ coefficient in Newmark or HHT method
- `niter`: Number of iterations in Shing method
- `am`: Reyleigh damping coefficient in terms of mass matrix
- `bk`: Reyleigh damping coefficient in terms of initial stiffness matrix
- `nstep`: Total time steps in transient analysis

2.9 Output

Mercury C++ version should make function to save output data during analysis. It uses "Lua" language.