Journey through Nonlinear Structural Modeling with OpenSees, FEDEASLab and χara

Claudio M. Perez and Filip C. Filippou

University of California, Berkeley



Presentation Objectives

- Purpose of FEDEASLab and its evolution in the last 30 years
- Introduction of χara and its Objectives (Claudio M. Perez)
- Introduce improved models for nonlinear structural response

Outline

Point of Departure

Architecture

Formulations

Final Example

Conclusion

FEDEASLab

FEDEASLab stands for Finite Elements for Design, Evaluation, and Analysis of Structures. It is a Matlab® toolbox of functions that was conceived in 1998 to assist with the instruction of structural analysis courses, and to be used for individual study and research.

It is modular in its architecture to facilitate gradual progression from simple to advanced structural analysis concepts in structural analysis courses and from simple to complex tests and simulations during the development of new materials, elements, solution strategies and simulation methods.

Data Organization of FEDEASLab

FEDEASLab functions operate on six data structures:

1. `Model` - information about the structural model and its geometry.
2. `ElemData` - element property information.
3. `Loading` - information about applied nodal forces and displacements.
4. `SolStrat` - information about the solution strategy parameters.
5. `State` - information about the state of the model.
6. `Post` - information about the element response for *post-processing*.

OpenSees Interpreters

Data is maintained *implicitly*:

```
// commands.cpp
AnalysisModel *theAnalysisModel =0;
EquiSolnAlgo *theAlgorithm =0;
LinearSOE *theSOE =0;
StaticAnalysis *theStaticAnalysis = 0;
LoadPattern* theActiveLoadPattern = 0;
```

OpenSees Interpreters

Data is maintained *implicitly*:

```
// commands.cpp
AnalysisModel *theAnalysisModel =0;
EquiSolnAlgo *theAlgorithm =0;
LinearSOE *theSOE =0;
StaticAnalysis *theStaticAnalysis = 0;
LoadPattern* theActiveLoadPattern = 0;
```

- Crashes

OpenSees Interpreters

Data is maintained *implicitly*:

```
// commands.cpp
AnalysisModel *theAnalysisModel =0;
EquiSolnAlgo *theAlgorithm =0;
LinearSOE *theSOE =0;
StaticAnalysis *theStaticAnalysis = 0;
LoadPattern* theActiveLoadPattern = 0;
```

- Crashes
- Memory leaks

OpenSees Interpreters

Data is maintained *implicitly*:

```
// commands.cpp
AnalysisModel *theAnalysisModel =0;
EquiSolnAlgo *theAlgorithm =0;
LinearSOE *theSOE =0;
StaticAnalysis *theStaticAnalysis = 0;
LoadPattern* theActiveLoadPattern = 0;
```

- Crashes
- Memory leaks
- Incorrect results

OpenSees Interpreters

Data is maintained *implicitly*:

```
// commands.cpp
AnalysisModel *theAnalysisModel =0;
EquiSolnAlgo *theAlgorithm =0;
LinearSOE *theSOE =0;
StaticAnalysis *theStaticAnalysis = 0;
LoadPattern* theActiveLoadPattern = 0;
```

- Crashes
- Memory leaks
- Incorrect results
- Prohibits multiple models or analyses on process

OpenSees Interpreters

Data is maintained *implicitly*:

```
// commands.cpp
AnalysisModel *theAnalysisModel =0;
EquiSolnAlgo *theAlgorithm =0;
LinearSOE *theSOE =0;
StaticAnalysis *theStaticAnalysis = 0;
LoadPattern* theActiveLoadPattern = 0;
```

- Crashes
- Memory leaks
- Incorrect results
- Prohibits multiple models or analyses on process
- Inhibits ecosystem development

χara: An alternative interpreter

- Inspired by FEDEASLab

χara: An alternative interpreter

- Inspired by FEDEASLab
- Not a wrapper

χara: An alternative interpreter

- Inspired by FEDEASLab
- Not a wrapper
- Same object library

χara: An alternative interpreter

- Inspired by FEDEASLab
- Not a wrapper
- Same object library
- Unified Tcl and Python

χara: An alternative interpreter

- Inspired by FEDEASLab
- Not a wrapper
- Same object library
- Unified Tcl and Python
- Backwards compatible

Interpreter: Encapsulation

OpenSeesPy:

```
import opensees.openseespy as ops
ops.model(...) # model is globally exposed
ops.node(...)  # Adds node to global model
```

χara:

```
import xara
model = xara.Model(...) # model is encapsulated
model.node(...)
```

True encapsulation without interprocess communication.

Interpreter: Explicit Referencing

OpenSeesPy:

```
ops.pattern("Plain", 1, "Linear") # Create pattern
ops.load(2, 1.0, 0.0, 0.0) # Add force at node 2
```

χara:

```
model.pattern("Plain", 1, "Linear")
model.load(2, 1.0, 0.0, 0.0, pattern=1)
```

Interpreter: Native Semantics

OpenSeesPy:

```
ops.model("basic", "-ndm", 3)
```

χara:

```
ops.model("basic", ndm=3)
```

Interpreter: Logging and Diagnostics

OpenSees:

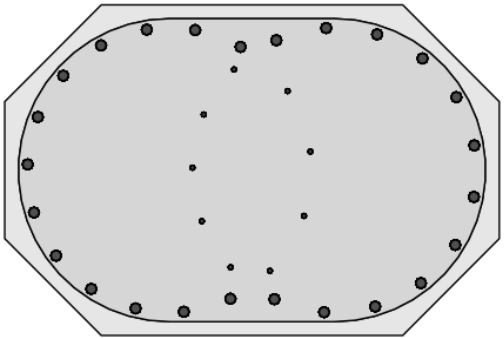
```
CTestNormDispIncr::test() - iteration: 1 current Norm: 2474.97 (max: 1e-10, Norm R: 2.2017e-08)
CTestNormDispIncr::test() - iteration: 2 current Norm: 2.70165e-08 (max: 1e-10, Norm R: 9.38875e-09)
WARNING: CTestNormDispIncr::test() - failed to converge
after: 2 iterations  current Norm: 2.70165e-08 (max: 1e-10, Norm deltaR: 9.38875e-09)
NewtonRaphson::solveCurrentStep() -the ConvergenceTest object failed in test()
StaticAnalysis::analyze() - the Algorithm failed at step: 0 with domain at load factor 200
OpenSees > analyze failed, returned: -3 error flag
```

χara:

```
   ITERATE :: Iter:     1, Norm:     2474.97, Norm deltaR:  2.2017e-08
   ITERATE :: Iter:     2, Norm: 2.70165e-08, Norm deltaR: 9.38875e-09
   FAILURE :: Iter:     2, Norm: 2.70165e-08, Norm deltaR: 9.38875e-09
```

Formulations

Sections
Elements
Geometry
Damage
Concrete



Multiaxial Materials in Sections

Fiber sections require a kinematic assumption.

Plane section:

$$T = GI_0\varkappa$$
$$V_y = GA\gamma_y$$
$$V_z = GA\gamma_z$$

Multiaxial Materials in Sections

Fiber sections require a kinematic assumption.

Plane section:

$$T = GI_0\varkappa$$
$$V_y = GA\gamma_y$$
$$V_z = GA\gamma_z$$

Preferred:

$$T = GJ\varkappa$$
$$V_y = GA_y\gamma_y$$
$$V_z = GA_z\gamma_z$$

Multiaxial Materials in Sections: Solution

`MultiaxialFiber` formulation:

- Input Data

$$\{y_i, z_i, A_i, \varphi_{ij}\}$$

- State Determination:

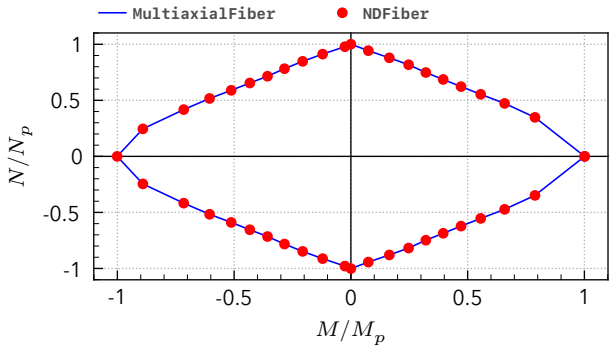
$$\boldsymbol{\varepsilon}(y, z) = \boldsymbol{A}\boldsymbol{e} + \bar{\boldsymbol{A}}\boldsymbol{\eta}$$

$$\boldsymbol{g}(\boldsymbol{\eta}) = \int \bar{\boldsymbol{A}}^{\top} \boldsymbol{\sigma}\left(\hat{\boldsymbol{\varepsilon}}(\boldsymbol{e}) + \bar{\boldsymbol{A}}\boldsymbol{\eta}\right) \, \mathrm{d}A$$

Perez, C. M. (2026). *Nonlinear modeling of frame members for rapid infrastructure assessment*. PhD thesis. University of California, Berkeley.

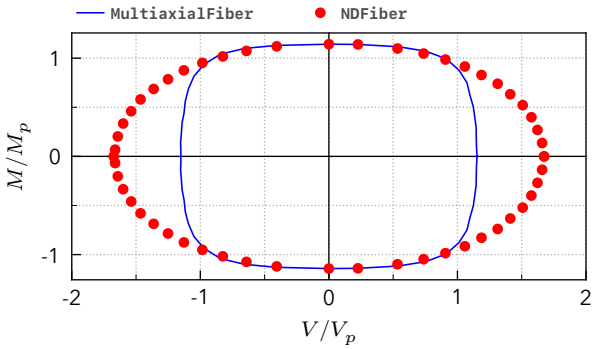
`MultiaxialFiber` Section: Plastic Response

- Axial-Flexure



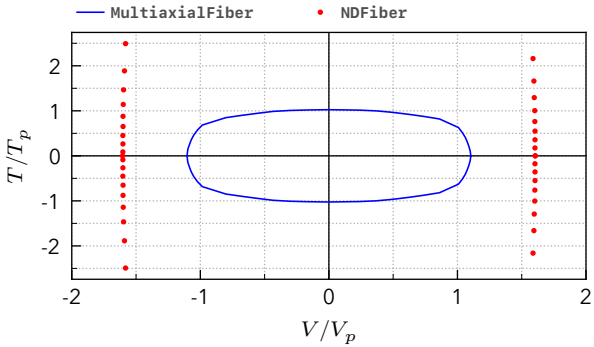
`MultiaxialFiber` Section: Plastic Response

- Axial-Flexure
- Shear-Flexure

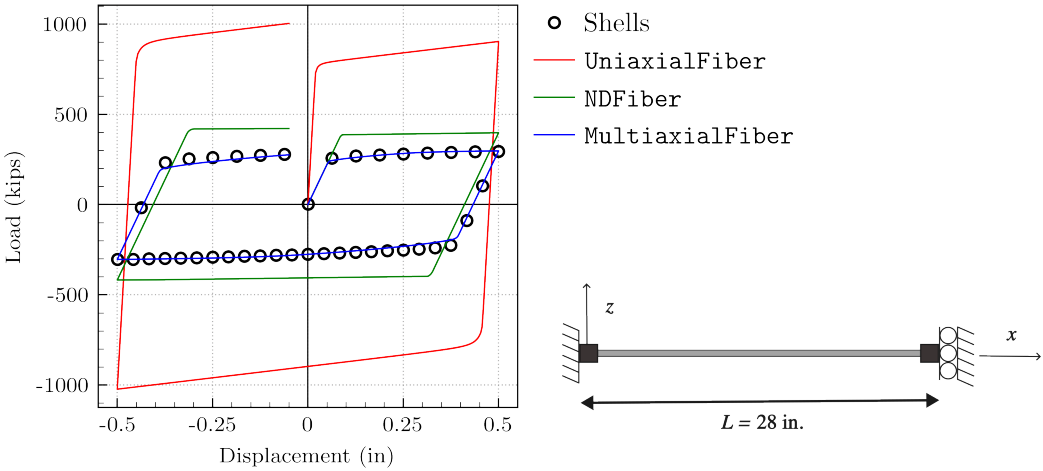


`MultiaxialFiber` Section: Plastic Response

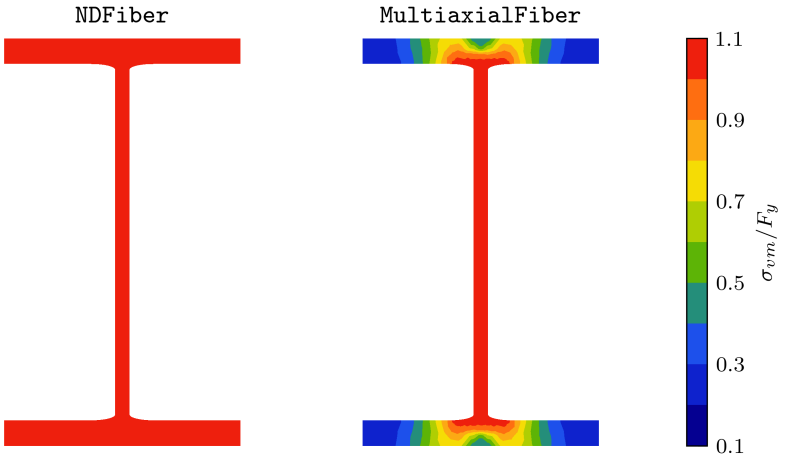
- Axial-Flexure
- Shear-Flexure
- Shear-Torsion



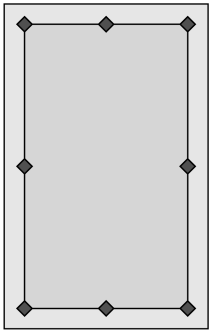
Example: Steel Shear Link, Element Response



Example: Steel Shear Link, Section Response

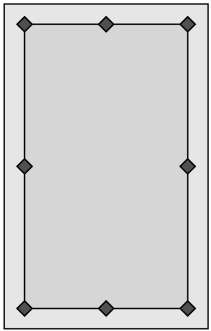
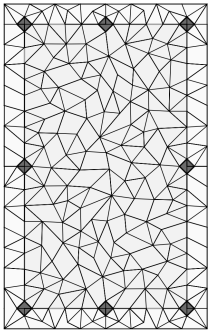
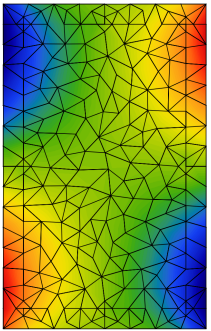
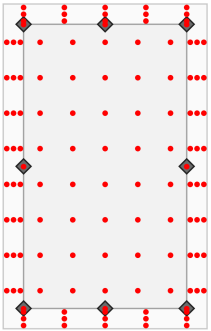


Common Section Workflow

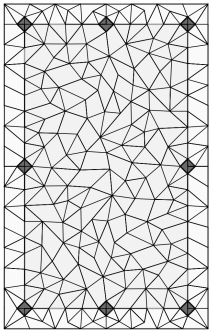


```
shape = MergedShape([
  Rectangle(    b, h,      z=0, group="cover"),
  Rectangle(b-2*c, h-2*c, z=1, group="core"),
  # Top bars
  *bar.linspace([-b/2+c, -h/2+c], [ b/2-c,-h/2+c], 3),
  # Center bars
  *bar.linspace([-b/2+c,      0], [ b/2-c,     0], 2),
  # Bottom bars
  *bar.linspace([-b/2+c,  h/2-c], [ b/2-c, h/2-c], 3)
])
```

Common Section Workflow

Common Section Workflow



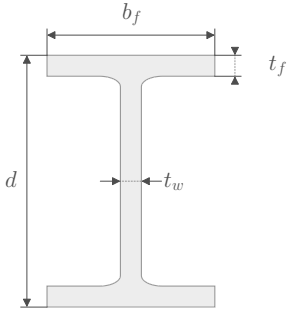
```
model.section("<type>", shape)
```

shape automatically provides:

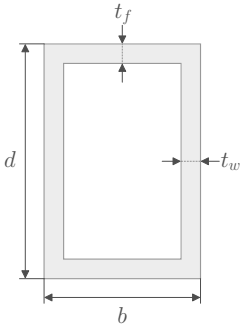
- GJ for **`UniaxialFiber`** and **`Elastic`**
- φ_{ij} for **`MultiaxialFiber`**
- κ for **`NDFiber`**

Shape Library

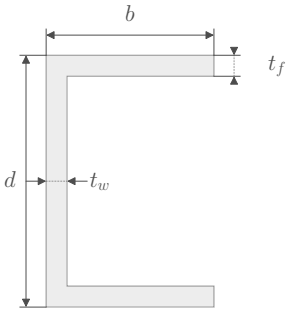
Standard European steel shapes are built-in:



IPE/HEA/HEB



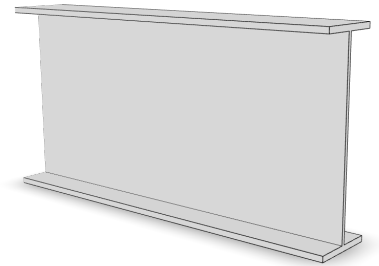
IPE/HEA/HEB



UPN, UPE, PFC

Formulations

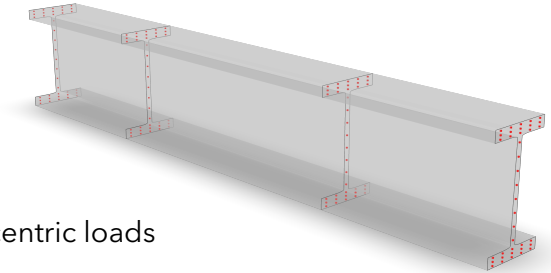
Sections
Elements
Geometry
Damage
Concrete



Beam Element: `ForceFrame`

A general-purpose inelastic beam element.

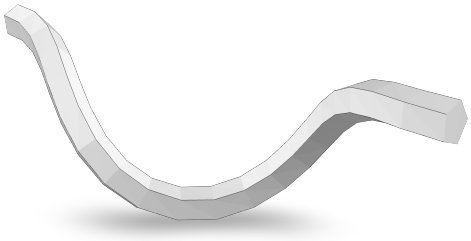
- Force (mixed) formulation
- Optional shear
- Asymmetric sections
- Optional non-uniform warping
- Conservative, follower, and eccentric loads



Beam Element: `CosseratFrame`

An inelastic beam element for flexible members like cables.

- Displacement formulation
- Exact kinematics
- Shear
- Asymmetric sections
- Optional non-uniform warping
- Supports $2 \leq n \leq 5$ nodes
- Conservative, follower, and eccentric loads



Frame Elements

```
model.element(type, tag, nodes, section, transform, **options)
```

Argument

- `type`
- `section`
- `transform`

Element Types (Standard Library)

- **`ElasticFrame`**
- **`ForceFrame`**
- **`CosseratFrame`**

Frame Elements

```
model.element(type, tag, nodes, section, transform, **options)
```

Argument

- type
- section
- transform

Types

- **Elastic**
- **UniaxialFiber**
- **MultiaxialFiber**

Frame Elements

```
model.element(type, tag, nodes, section, transform, **options)
```

Argument

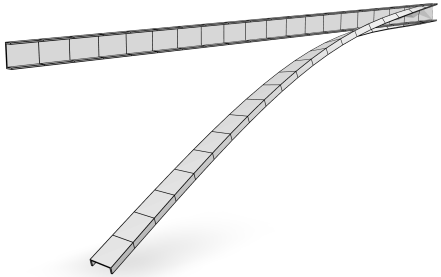
- type
- section
- transform

Types

- **Linear**
- **PDelta**
- **Corotational**

Formulations

- Sections
- Elements
- Geometry
- Damage
- Concrete



Geometry

- **PDelta**
- Corotational
- Spherical

$$\boldsymbol{K} = \boldsymbol{A}^\top \boldsymbol{K}_q \boldsymbol{A} + \frac{N}{L}\boldsymbol{K}_\Delta + \frac{1}{L}\boldsymbol{g} \otimes \boldsymbol{k}_N$$

Neglecting the asymmetric term, causes extra iterations and sometimes non-convergence.

Perez, C. M. and F. C. Filippou (June 10, 2024). *On Nonlinear Geometric Transformations of Finite Elements*. In: *International Journal for Numerical Methods in Engineering*, e7506.

Geometry

- PDelta
- **Corotational**
- Spherical



Perez, C. M. and F. C. Filippou (June 10, 2024). *On Nonlinear Geometric Transformations of Finite Elements*. In: *International Journal for Numerical Methods in Engineering*, e7506.

Geometry

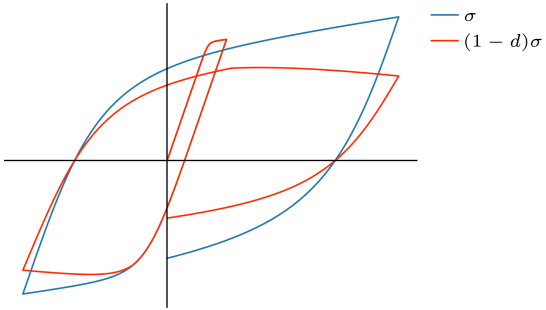
- PDelta
- Corotational
- **Spherical**



Perez, C. M. and F. C. Filippou (June 10, 2024). *On Nonlinear Geometric Transformations of Finite Elements*. In: *International Journal for Numerical Methods in Engineering*, e7506.

Formulations

Sections
Elements
Geometry
Damage
Concrete



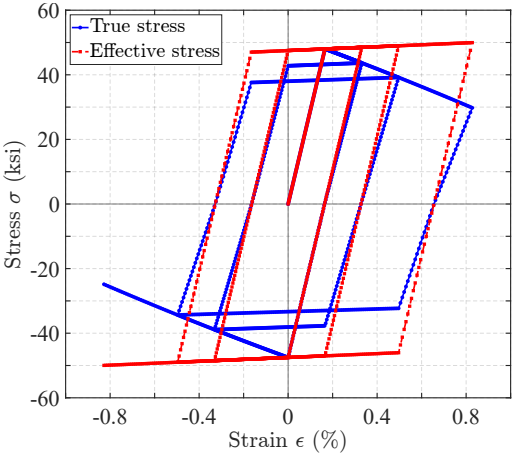
Damage Wrappers

Damage is applied to an *effective* model to produce a *true* response that reflects strength and stiffness degradation.

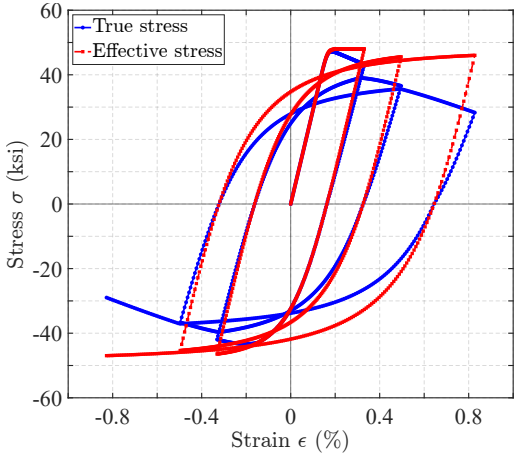
- Damage wrapper uniaxial or multi-axial material: ε, σ
- Damage wrapper section: $\boldsymbol{e}, \boldsymbol{s}$
- Damage wrapper element: $\boldsymbol{v}, \boldsymbol{q}$

Cohen, Jade E. D. (2022). *A Flexible Framework for the Damage-Based Modeling of Frame Elements With Applications to Steel Structures*. PhD thesis. University of California, Berkeley.
Do Ngoc, Thanh (2017). *Damage Assessment and Collapse Simulations of Structures under Extreme Loading Conditions*. PhD thesis. UC Berkeley.

Damage wrapper material (`UniaxialDamage`)

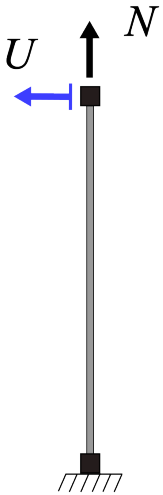


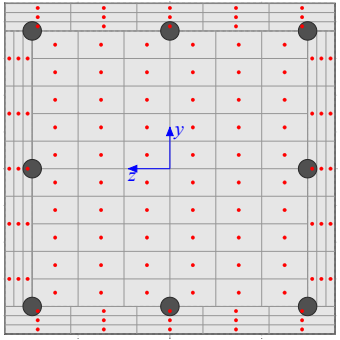
Wrapping uniaxial plasticity model

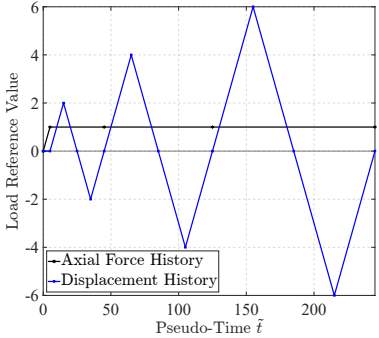


Wrapping GMP material model

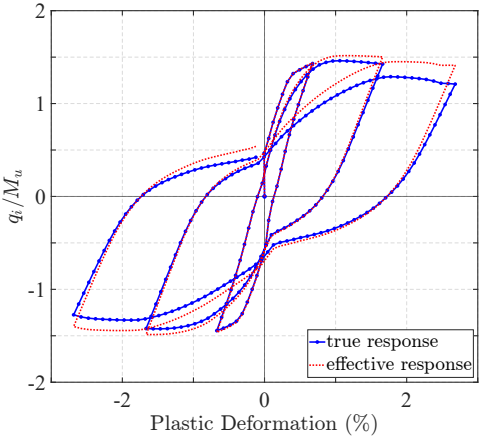
Concrete Column with Damage

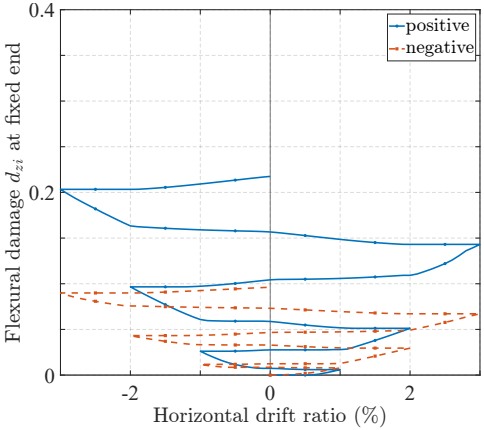




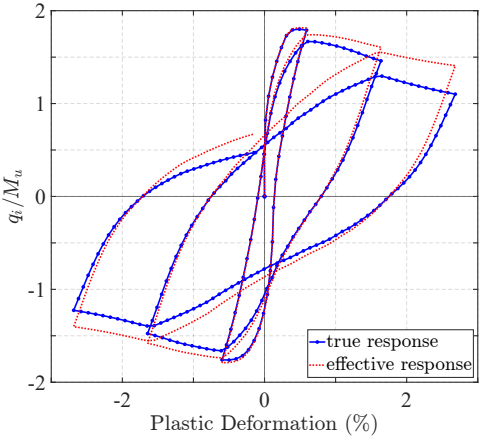


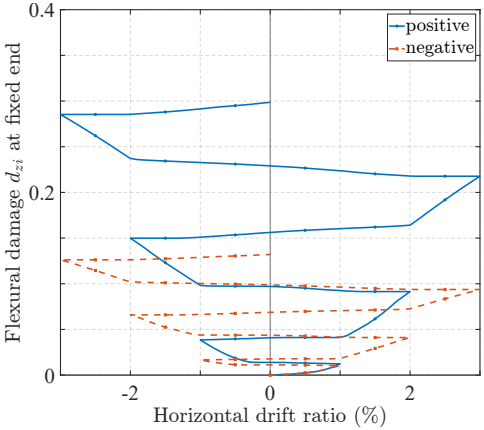
Concrete Column with Damage Wrapper, N=-0.10N_u





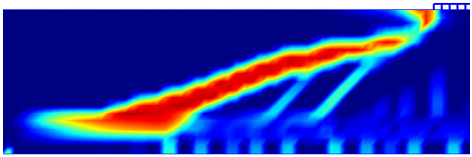
Concrete Column with Damage Wrapper, N=-0.25N_u





Formulations

Sections
Elements
Geometry
Damage
Concrete



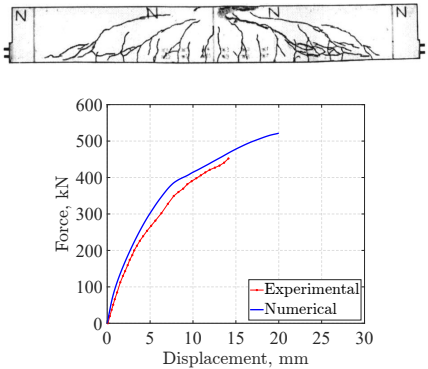
Multiaxial Concrete

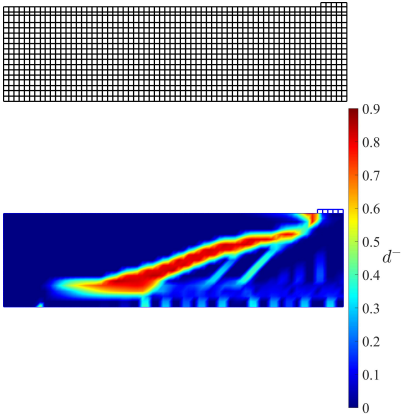
Component	Formulation
Effective stress response	Classical effective-stress plasticity with a Lubliner/Lee-Fenves type yield function and non-associated flow rule.
Tensile damage	Fourth-order compliance damage with anisotropic evolution and fracture-energy-based hyperbolic softening.
Compressive damage	Scalar compressive degradation with triaxial threshold modification and integrable hardening-softening.
RC interaction effects	Tension stiffening and compression softening added at the material level.
Regularization	Direction-aware characteristic length with crack-spacing limiters.

Velkov, Nikolay G (2025). *Anisotropic Damage-Plasticity Model for Reinforced Concrete*. PhD thesis. University of California, Berkeley.

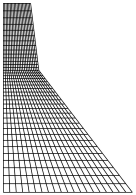
Anisotropic multi-axial damage plasticity model for concrete

Bresler-Scordelis RC beam A1

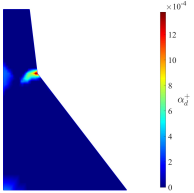




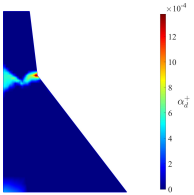
Dynamic analysis: Koyna dam



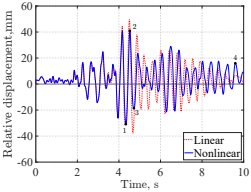
Finite element mesh



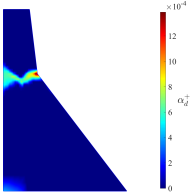
Damage snapshot 1



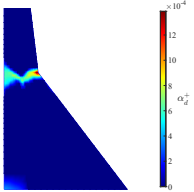
Damage snapshot 2



Crest displacement

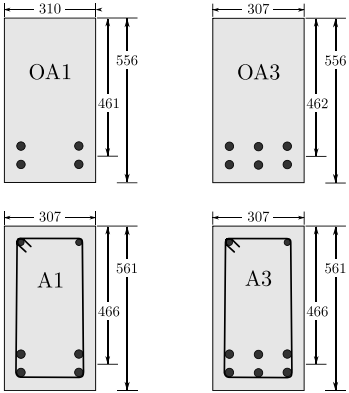


Damage snapshot 3

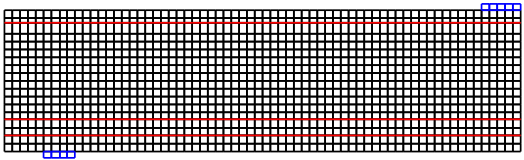


Damage snapshot 4

RC beams: Bresler and Scordelis, 1963



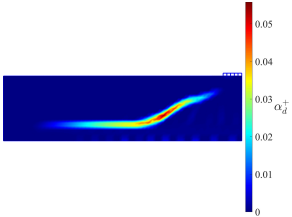
Cross section geometry, mm



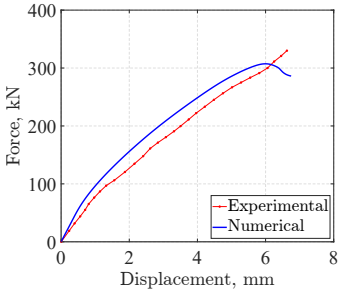
Representative FE mesh



OA1 failure state



OA1 tensile damage

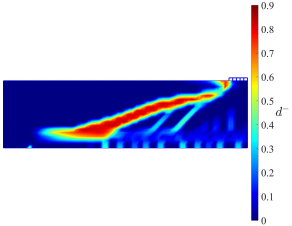


Mechanism: diagonal-tension

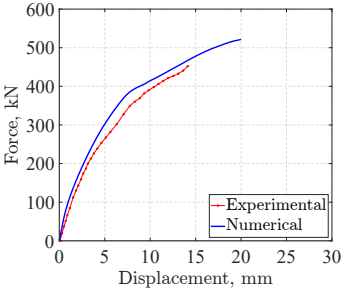
- dominant inclined crack
- splitting along tensile reinforcement



A1 failure state



A1 compressive damage

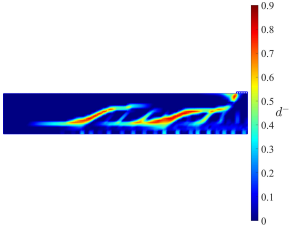


Mechanism: shear-compression

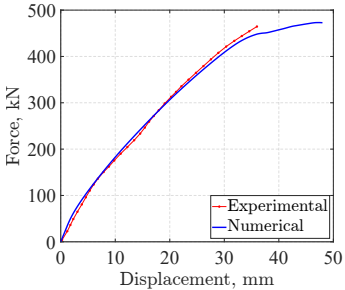
- inclined cracking
- compression strut crushing



A3 failure state



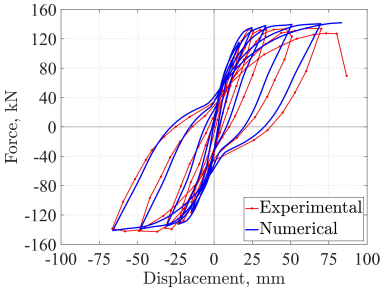
A3 compressive damage



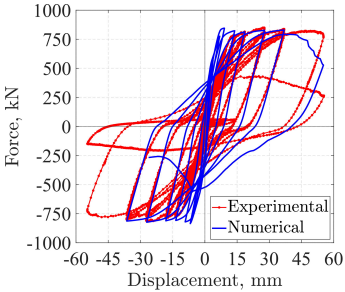
Mechanism: flexure-compression

- flexural cracking
- crushing near load point

RC shear walls



Response of slender wall



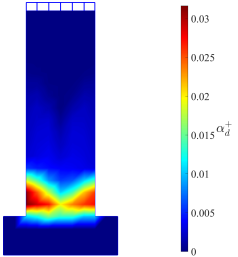
Response of squat wall

Specimen	Aspect Ratio	Behavior
RW1 (Thomsen and Wallace, 2004)	3.0	flexure-dominated
S78 (Tran and Wallace, 2015)	1.5	shear-dominated

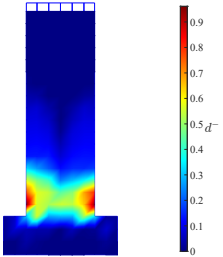
RC shear walls: Thomsen and Wallace, 2004



Failure state



Tensile damage



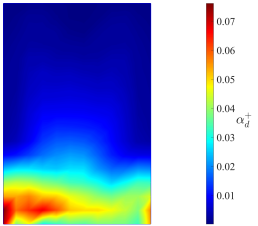
Compressive damage

- **flexure-dominated response localizes damage in boundary elements**
- **failure is governed by bar buckling and fracture, not modeled explicitly**

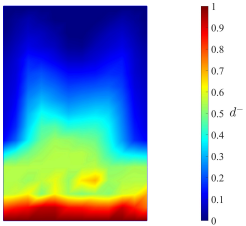
RC shear walls: Tran and Wallace, 2015



Failure state



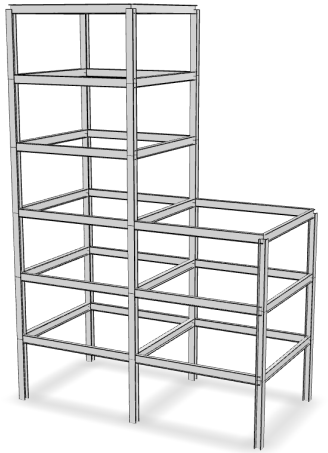
Tensile damage



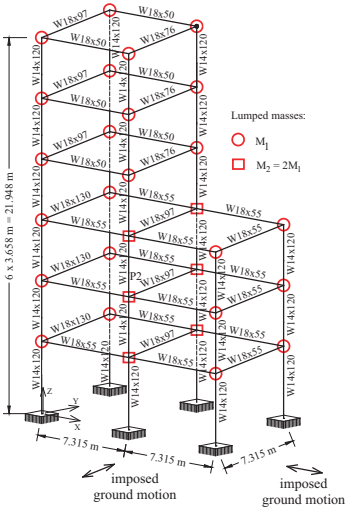
Compressive damage

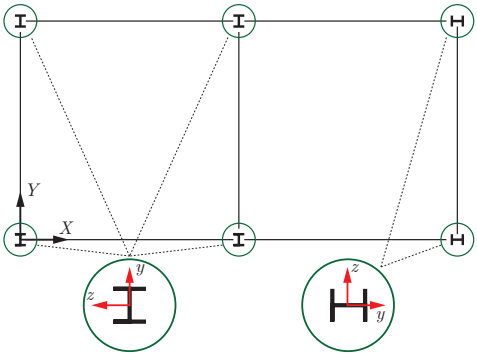
- **shear-dominated response characterized by diagonal cracking**
- **significant crushing and horizontal sliding at the base**

Final Example



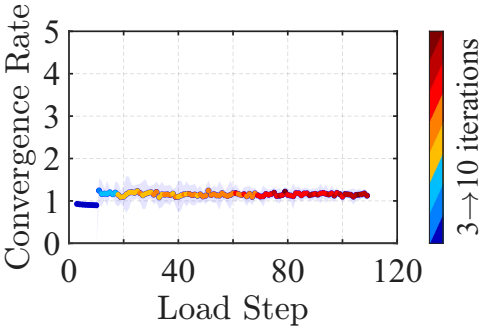
Irregular 6-Story Steel Frame



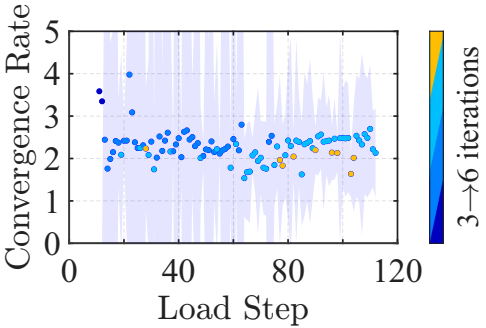


Cohen, Jade E. D. (2022). *A Flexible Framework for the Damage-Based Modeling of Frame Elements With Applications to Steel Structures*. PhD thesis. University of California, Berkeley.

6 Story Frame Pushover: $P - \Delta$

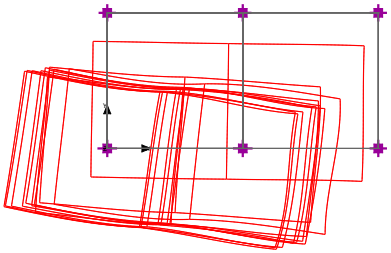


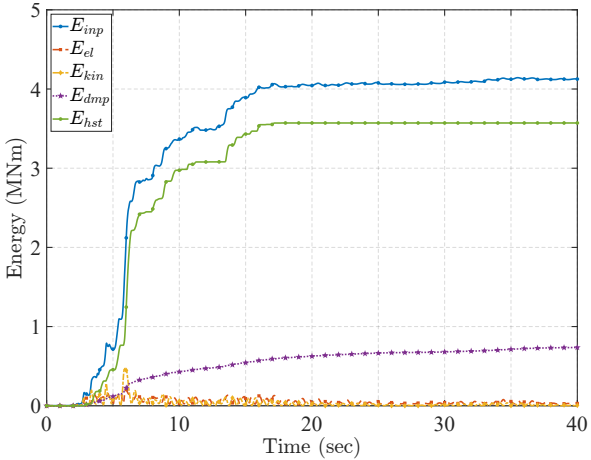
Classical



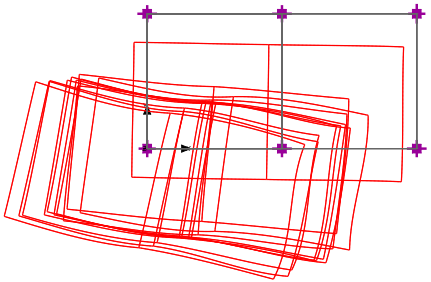
Xara

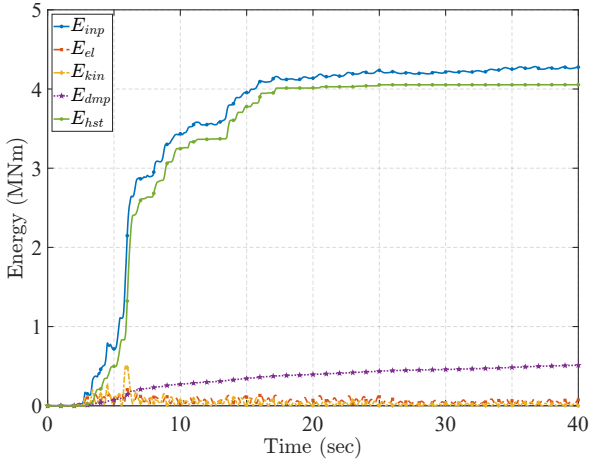
6-Story Steel Frame: Response with Damage (2%)



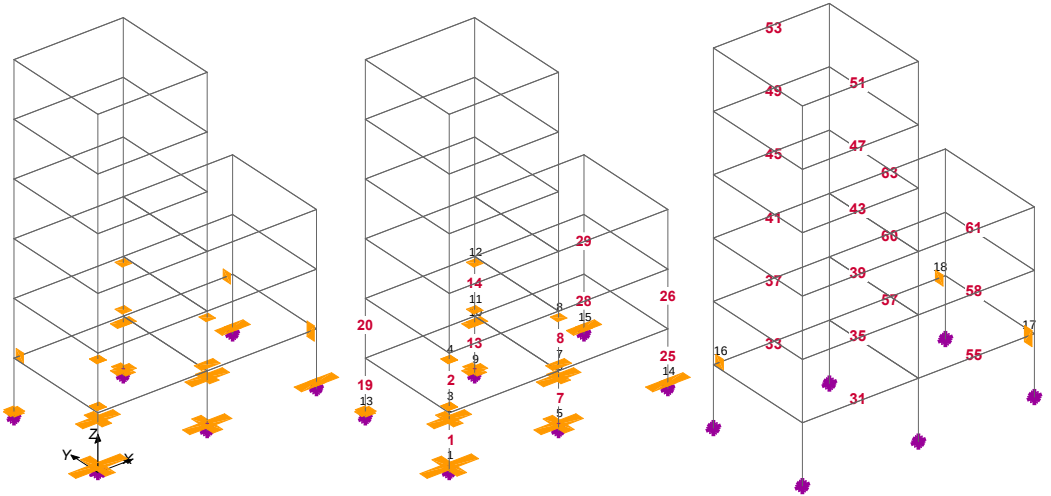


6-Story Steel Frame: Response with Damage (1.2%)





6-Story Steel Frame: Response with Damage (2%)



Summary

Section	`MultiaxialFiber`, `UniaxialFiber`
Material	GeneralizedPlasticity, NonlinearPlasticity, FariaConcrete, **VelkovConcrete**
Geometry	PDelta, Corotational, Spherical
Elements	`CosseratFrame`, `ForceFrame`
Damage	`UniaxialDamage`, **`SectionDamage`** **ElementDamage**

■: In FEDEASLab, being added to χara

■: In χara and FEDEASLab

Conclusions

- An encapsulated interpreter architecture allows the benefits of the OpenSees core architecture to be reaped by users at the interpreter level.

Conclusions

- An encapsulated interpreter architecture allows the benefits of the OpenSees core architecture to be reaped by users at the interpreter level.
- Incorporating strain shapes significantly improves the response of fiber sections with multiaxial materials.

Conclusions

- An encapsulated interpreter architecture allows the benefits of the OpenSees core architecture to be reaped by users at the interpreter level.
- Incorporating strain shapes significantly improves the response of fiber sections with multiaxial materials.
- Refactored geometric transformations improve convergence and correct an objectivity violation in the Corotational formulation.

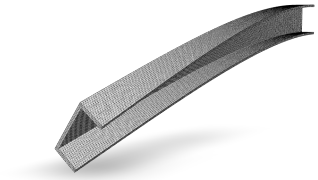
Conclusions

- An encapsulated interpreter architecture allows the benefits of the OpenSees core architecture to be reaped by users at the interpreter level.
- Incorporating strain shapes significantly improves the response of fiber sections with multiaxial materials.
- Refactored geometric transformations improve convergence and correct an objectivity violation in the Corotational formulation.
- Damage wrappers offer a powerful and intuitive tool for improving hysteretic response.

Python Ecosystem

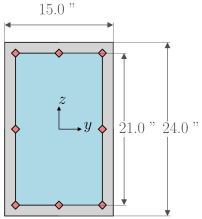
χara is supported by a robust ecosystem of extension libraries.

veux



Interactive rendering library.

xsection



Cross section analysis and streamlined fiber generation.

xcsi/xcae



Import models from products like SAP2000, CSiBridge, and Abaqus.

Availability

- FEDEASLab will be released as p-code in late 2026.
- χara is documented at:

  `https://xara.so`

- χara is available for Python 3.9–3.14:

  `pip install xara`

- All contributions and extensions are welcomed:

  `peer-open-source/xara`

Acknowledgments

- Dr. Frank McKenna, Prof. Michael Scott
- Prof. Khalid Mosalam
- OpenSees contributors
- FEDEASLab contributors:
  - Remo De Souza
  - Afsin Saritas
  - Véronique Le Corvec
  - Chin-Long Lee
  - Thanh Do Ngoc
  - Jade Cohen
  - Nikolay Velkov





