CIEM5110-2: FEM, workshop 3.2

pyJive: SolverModule, TimoshenkoModel, FrameModel

Frans van der Meer, Iuri Rocha

TU Delft

Modules and models

Generally a `pyJive` job has several modules

- Read a mesh from file
- Solve the problem
- Store and visualize results

And several models

- Dirichlet boundary conditions
- Neumann boundary conditions
- Assemble the matrix

Modules and models

Generally a `pyJive` job has several modules

- Read a mesh from file
- **Solve the problem**
- Store and visualize results

And several models

- Dirichlet boundary conditions
- Neumann boundary conditions
- **Assemble the matrix**

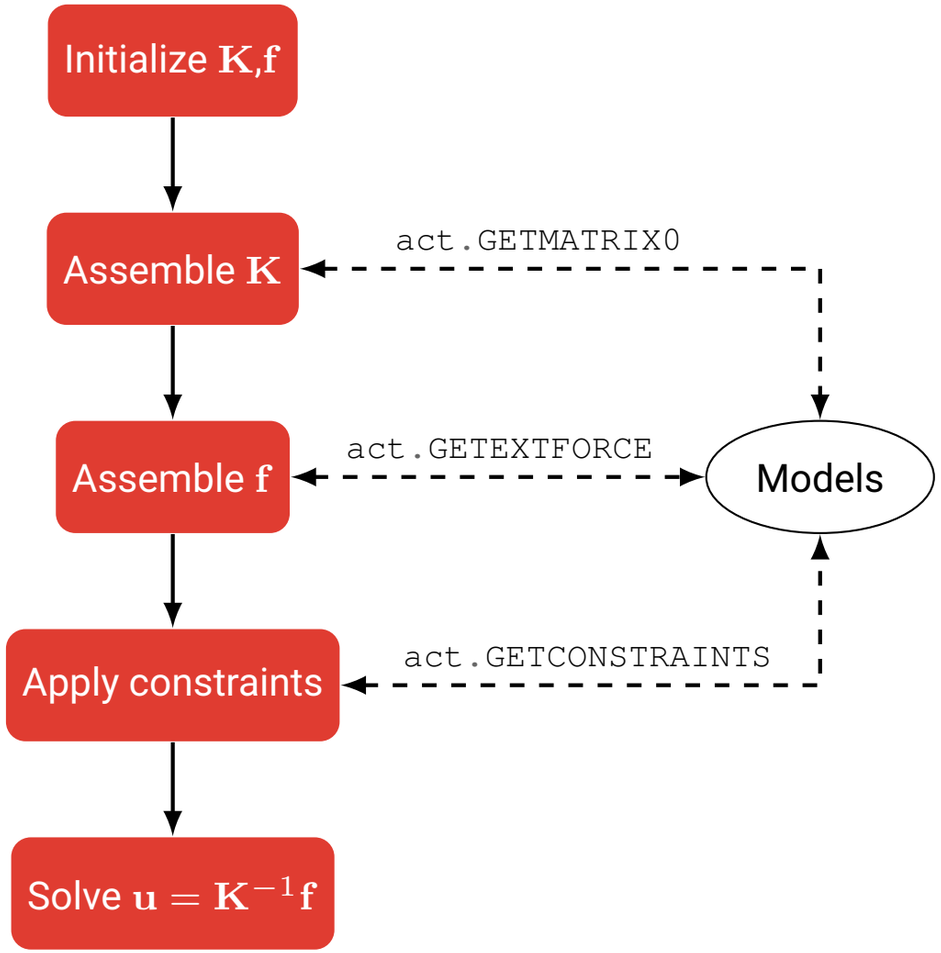
Different combinations of **‘solver module’** and **‘matrix model’** are possible

CIEM5110-2 workshops and lectures

	(Theory)	BarModel (MUDE)	SolidModel (1.2)	TimoshenkoModel (2.1)	FrameModel (4.1)
SolverModule	(1.2)	2.2	2.2	**3.2**	**3.2**
NonlinModule	(3.1)		6.1		4.1 + 4.2 + 5.1
LinBuckModule	(4.1)				4.1 + 5.1
ModeShapeModule	(6.2)		7.1		7.1 + 8.1
ExplicitTimeModule	(6.2)				7.2 + 8.1
NewmarkModule	(6.2)				7.2 + 8.1

SolverModule: run

```
# Advance time step
super().advance(globdat)
model.take_action(act.ADVANCE, params, globdat)

# Assemble K
model.take_action(act.GETMATRIX0, params, globdat)

# Assemble f
model.take_action(act.GETEXTFORCE, params, globdat)

# Get constraints
model.take_action(act.GETCONSTRAINTS, params, globdat)

# Constrain K and f
Kc, fc = c.constrain(K, f)

# Sparsify and solve
smat = sparse.csr_matrix(Kc)
u = linalg.spsolve(smat, fc)

# Store rhs and solution in Globdat
globdat[gn.EXTFORCE] = f
globdat[gn.STATE0] = u
```



Modules and models in workshops so far

Main module

- **`SolverModule`**: solves linear system of equations

Main models, with those from **this workshop** highlighted

- `BarModel`: assembles matrix for 1D bar problems
- `SolidModel`: assembles matrix (and body load) for continuum elastostatics
- **`TimoshenkoModel`**: assembles matrix for Timoshenko beam analysis
- **`FrameModel`**: assembles matrix for frame analysis

Boundary condition models

- **`DirichletModel`**: defines supports (and other prescribed displacements)
- **`NeumannModel`**: defines forces on the boundary

Postprocessing modules

- `ViewModule`: for 2D continuum fields
- **`FrameViewModule`**: postprocessor for frame analysis