CEE432/CEE532 Developing Software for Engineering Applications

Lecture 17: Computer Program for Space Truss Analysis

Specifications

- No size limitation (except OS-related)
- Consistent units
- One global X-Y-Z coordinate system
- Nodes
  - ID is an integer starting at 1
  - Can be placed anywhere in the XYZ system
  - Nodal forces are in the XYZ directions
  - Fixity conditions are FREE or SPECIFIED

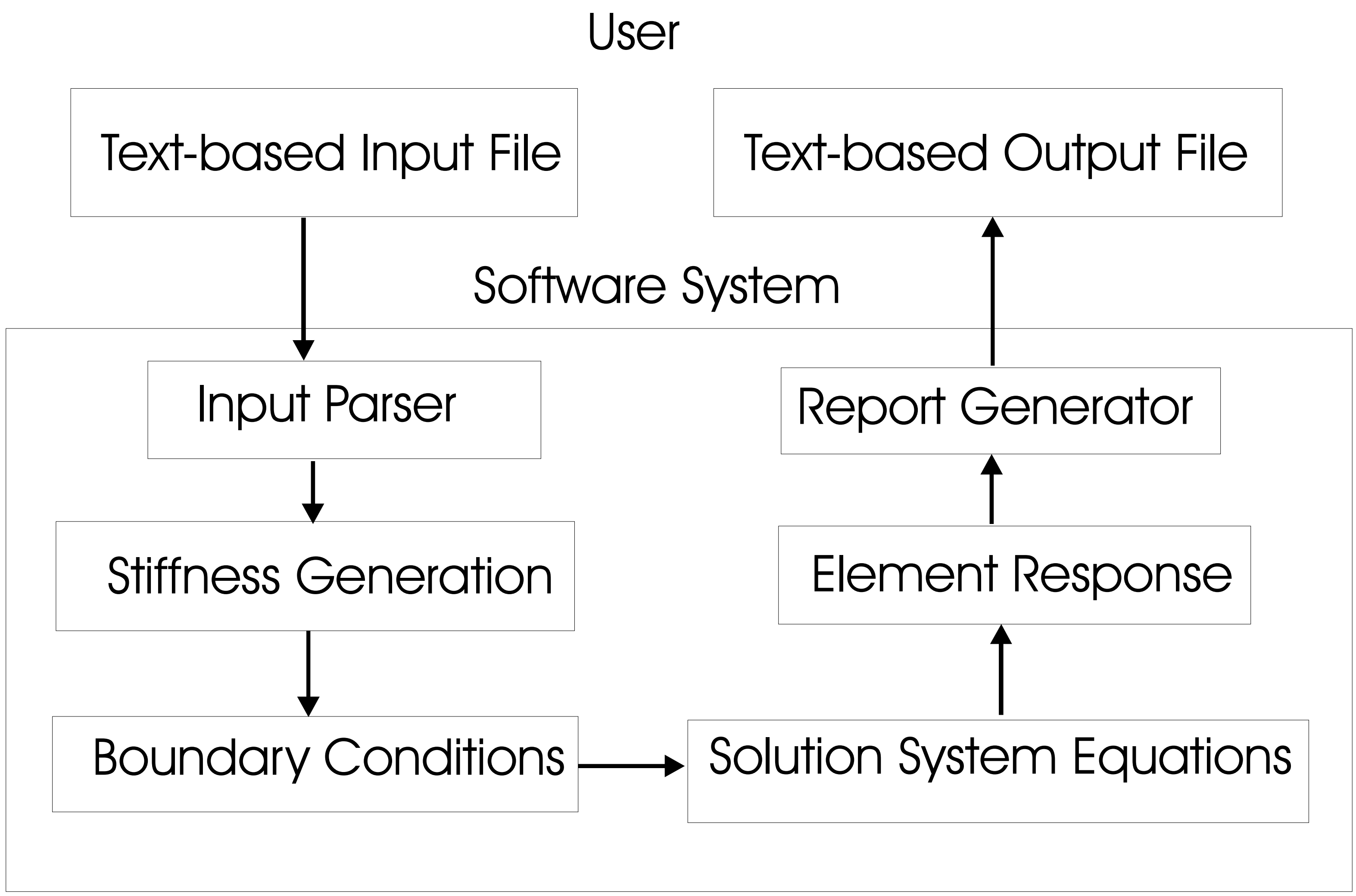
Specifications

- Elements
  - ID is an integer starting at 1
  - Defined in terms start and end node
  - Young's modulus, coef. of thermal expansion
  - Cross-sectional area
- Materially linear, small displacement, small strain analysis

Specifications

- Input
  - Text file with a specified format
  - Ask user for input file name
- Output
  - Test file with specified format
  - Ask use for output file name

Program Flow



OO Modeling (Planar Truss)

- Noun and noun clauses

Node	**Element**	**Units**	**Area**
Young's Modulus	**Nodal Loads**	**Nodal Fixity Conditions**	**Nodal Displacements**
Start Node	**End Node**	**Element strain**	**Element stress**
Support Reaction			

OO Modeling

- Aggregation leads to identification of entities
- We can start developing CRH (Component-responsibility-Helper) Cards

OO Modeling

Class: CNode

Responsibilities:

know coordinates

know fixity conditions

know loads

allow access to the above

Helpers:

OO Modeling

Class: CElement

Responsibilities:

know the 2 nodes

know A and E

allow access to the above

Helpers:

OO Modeling

Class: CNodalResponse

Responsibilities:

know x-displacement

know y-displacement

allow access to the above

Helpers:

OO Modeling

Class: CElementResponse

Responsibilities:

know strain

know stress

know force

allow access to the above

Helpers:

OO Modeling (Composition)

Class: CTruss

Responsibilities:	**Helpers:**
know nodal data	CNode
know nodal response data	CNodalResponse
know element data	CElement
know element response data	CElementResponse
allow access to the above	CVector
	CMatrix

OO Modeling

Class	Definition
`CTruss`	The one and only object that contains the truss data and behavior.
`CNode`	Node-related class.
`CElement`	Element-related class.
`CNodalResponse`	Structural response that is node-related.
`CElementResponse`	Structural response that is element-related.
`CVector`	Vector class used as a container.
`CMatrix`	Matrix class used as a container

Algorithm

Step 1: Read and check the input data.

Step 2: Initialize all objects in the program.

Step 3: Construct the element stiffness matrix and load vector. Assemble into the system equations.

Step 4: Impose the boundary conditions.

Step 5: Solve for the nodal displacements.

Step 6: Compute the strains, stress and force in each member.

Step 7: Print the results.

Algorithm (Step 3)

Substep 1: Loop through all elements, i.

Substep 2: Compute L, l and m. Construct the element stiffness matrix $\mathbf{k}_{4x4}$.

Substep 3: Form the element nodal degrees-of-freedom vector, $\mathbf{e}_{4x1}$.

Substep 4: Loop through j. Set $r=e_j$.

Substep 5: Loop through k. Set $c=e_k$.

Substep 6: Update $\mathbf{K}$.

Substep 7: End loop, k.

Substep 8: End loop, j.

Substep 9: End loop, i.

Algorithm (Step 4)

Substep 1: Loop through all nodes, *i*.

Substep 2: Check if the x-displacement and/or the y-displacement is fixed (or suppressed).

Substep 3: If not, skip these steps. If yes, do the following. Identify the degree-of-freedom, *n*, associated with the suppressed displacement.

Substep 4: Loop through all degrees-of-freedom, *j*.

Substep 5: Set $K_{j,n}=K_{n,j}=0$.

Substep 6: End loop *j*.

Substep 7: Set $K_{n,n}=0$ and $F_n=0$.

Substep 8: End loop *i*.

Algorithm (Step 6)

Substep 1: Loop through all elements, *i*.

Substep 2: Compute *L*, *l* and *m*. Form the global-to-local transformation matrix, **T**.

Substep 3: Form the element nodal degrees-of-freedom vector, $\mathbf{e}_{\mathbf{4x1}}$. From the system nodal displacements vector **D** obtain the element nodal displacements, **d**.

Substep 4: Compute the element nodal displacements, **d'=Td**. Then **f'=k'd'**.

Substep 5: End loop, *i*.