22

FEM Programs for Plane Trusses and Frames

TABLE OF CONTENTS

§22.1. Introduction

This Chapter presents a complete FEM program for analysis of plane trusses, programmed in *Mathematica*. The program includes some simple minded graphics, including animation. Exercises 22.2-22.4 discuss a complete FEM program for frame analysis for homework assignments.

The description is done in "bottom up" fashion. That means the basic modules are presented, then the driver program. Graphic modules are provided in posted Notebooks but not explained.

§22.2. Analysis Stages

As in all FEM programs, the analysis of a structure by the Direct Stiffness Method involves three major stages: (I) preprocessing or model definition, (II) processing, and (III) postprocessing.

The *preprocessing* portion of the plane truss analysis is done by the driver program, which directly sets the data structures.

I.1 Model definition by direct setting of the data structures.

I.2 Plot of the FEM mesh, including nodes and element labels.

The *processing* stage involves three steps:

II.1 Assembly of the master stiffness matrix, with a subordinate element stiffness module.

II.2 Modification of master stiffness matrix and node force vector for displacement boundary conditions.

II.3 Solution of the modified equations for displacements. For the programs presented here the built in *Mathematica* function `LinearSolve` is used.

Upon executing these three processing steps, the displacements are available The following *post-processing* steps may follow:

III.1 Recovery of forces including reactions, done through a **Ku** matrix multiplication.

III.2 Computation of internal (axial) forces in truss members.

III.3 Plotting deflected shapes and member stress levels.

These steps will be demonstrated in class from a laptop computer.

§22.3. Analysis Support Modules

We begin by listing here the modules that support various analysis steps, and which are put into separate cells for testing convenience.

§22.3.1. Assembling the Master Stiffness

The function `PlaneTrussMasterStiffness`, listed in Figure 22.1, assembles the master stiffness matrix of a plane truss. It uses the element stiffness formation module `PlaneBar2Stiffness` described in the previous Chapter. The statements at the end of the cell test this module by forming and printing **K** of the example truss.

The arguments of `PlaneTrussMasterStiffness` are

```
PlaneTrussMasterStiffness[nodcoor_,elenod_,
  elemat_,elefab_,eleopt_]:=Module[
  {numele=Length[elenod],numnod=Length[nodcoor],
  e,eNL,eftab,ni,nj,i,j,ncoor,mprop,fprop,opt,Ke,K},
  K=Table[0,{2*numnod},{2*numnod}];
  For [e=1, e<=numele, e++,
     eNL=elenod[[e]]; {ni,nj}=eNL;
     eftab={2*ni-1,2*ni,2*nj-1,2*nj};
     ncoor={nodcoor[[ni]],nodcoor[[nj]]};
     mprop=elemat[[e]]; fprop=elefab[[e]]; opt=eleopt;
     Ke=PlaneBar2Stiffness[ncoor,mprop,fprop,opt];
     neldof=Length[Ke];
     For [i=1, i<=neldof, i++, ii=eftab[[i]];
         For [j=i, j<=neldof, j++, jj=eftab[[j]];
             K[[jj,ii]]=K[[ii,jj]]+=Ke[[i,j]]
             ];
         ];
      ]; Return[K];
  ];
PlaneBar2Stiffness[ncoor_,mprop_,fprop_,opt_]:= Module[
 {x1,x2,y1,y2,x21,y21,Em,Gm,rho,alpha,A,numer,L,LL,LLL,Ke},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,rho,alpha}=mprop; {A}=fprop; {numer}=opt;
  If [numer,{x21,y21,Em,A}=N[{x21,y21,Em,A}]];
  LL=x21^2+y21^2; L=PowerExpand[Sqrt[LL]]; LLL=Simplify[LL*L];
  Ke=(Em*A/LLL)*{{ x21*x21, x21*y21,-x21*x21,-x21*y21},
                 { y21*x21, y21*y21,-y21*x21,-y21*y21},
                 {-x21*x21,-x21*y21, x21*x21, x21*y21},
                 {-y21*x21,-y21*y21, y21*x21, y21*y21}};
  Return[Ke]
];
nodcoor={{0,0},{10,0},{10,10}};
elenod= {{1,2},{2,3},{1,3}};
elemat= Table[{100,0,0,0},{3}];
elefab= {{1},{1/2},{2*Sqrt[2]}};
eleopt= {True};
K=PlaneTrussMasterStiffness[nodcoor,elenod,
                            elemat,elefab,eleopt];
Print["Master Stiffness of Example Truss:"];
Print[K//MatrixForm];
```

FIGURE 22.1. Master stiffness assembly module, with test statements in red.

`nodcoor`	Nodal coordinates arranged as a two-dimensional list: `{{x1,y1},{x2,y2}, ... {xn,yn}}`, where `n` is the total number of nodes.
`elenod`	Element end nodes arranged as a two-dimensional list: `{{i1,j1}, {i2,j2}, ... {ie,je}}`, where `e` is the total number of elements.
`elemat`	Element material properties arranged as a two-dimensional list: `{{Em1,Gm1,rho1,alpha1}, ... {Eme,Gme,rhoe,alphae}}`, where `e` is the total number of elements. Only the elastic modulus `Em` is used in this program. For the other properties zeros may be stored as placeholders.

```
ModifiedMasterStiffness[pdof_,K_] := Module[
  {i,j,k,n=Length[K],np=Length[pdof],Kmod}, Kmod=K;
  For [k=1,k<=np,k++, i=pdof[[k]];
      For [j=1,j<=n,j++, Kmod[[i,j]]=Kmod[[j,i]]=0];
      Kmod[[i,i]]=1
      ];
  Return[Kmod]
];
ModifiedNodeForces[pdof_,f_] := Module[
  {i,k,np=Length[pdof],fmod}, fmod=f;
     For [k=1,k<=np,k++, i=pdof[[k]]; fmod[[i]]=0];
     Return[fmod]
];
K=Array[Kij,{6,6}];
Print["Assembled Master Stiffness:"];Print[K//MatrixForm];
K=ModifiedMasterStiffness[{1,2,4},K];
Print["Master Stiffness Modified For Displacement B.C.:"];
Print[K//MatrixForm];
f=Array[fi,{6}];
Print["Node Force Vector:"]; Print[f];
f=ModifiedNodeForces[{1,2,4},f];
Print["Node Force Vector Modified For Displacement B.C.:"];
Print[f];
```

FIGURE 22.2. Modifying the master stiffness and node force vector for displacement boundary conditions, with test statements in red.

`elefab`	Element fabrication properties arranged as a two-dimensional list: `{ { A1 }, ... { Ae } }`, where `e` is the total number of elements, and `A` the cross section area.
`eleopt`	Element processing option: set to `{ True }` to tell `PlaneBar2Stiffness` to carry out element stiffness computations in floating-point arithmetic. Else set to `{ False }` to keep computations in exact arithmetic or symbolic form.

The assembler uses the freedom-pointer-table technique described in §3.4 for merging the element stiffness matrix into the master stiffness. The module returns the master stiffness matrix $\mathbf{K}$ in list `K`, which is stored as a full matrix.

The statements at the end of Figure 22.1 test this module by forming and printing the master stiffness matrix of the example truss. These statements are executed when the cell is initialized.

§22.3.2. Modifying the Master Stiffness Equations

Following the assembly process the master stiffness equations $\mathbf{K}\mathbf{u} = \mathbf{f}$ must be modified to account for displacement boundary conditions. This is done through the computer-oriented equation modification process described in §3.4.2. Modules that perform this operation are listed in Figure 22.2, along with test statements.

Module `ModifiedMasterStiffness` carries out this process for the master stiffness matrix $\mathbf{K}$, whereas `ModifiedNodalForces` does this for the nodal force vector $\mathbf{f}$. The logic of `ModifiedNodalForces` is considerably simplified by assuming that *all prescribed displacements are zero*, that is, the BCs are homogeneous. This is the case in the implementation shown here.

Module `ModifiedMasterStiffness` receives two arguments:

`pdof`	A list of the prescribed degrees of freedom identified by their global number. For the example truss of Chapters 2-3 this list would have three entries: {1, 2, 4}, which are the freedom numbers of u_{x1}, u_{y1} and u_{y2} in $\mathbf{u}$.
`K`	The master stiffness matrix **K** produced by the assembler module described in the previous subsection.

The module clears appropriate rows and columns of **K**, places ones on the diagonal, and returns the thus modified **K** as function value. Note the use of the *Mathematica* function `Length` to control loops: `np=Length[pdof]` sets `np` to the number of prescribed freedoms. Similarly `n=Length[K]` sets `n` to the order of the master stiffness matrix **K**, which is used to bound the row and column clearing loop. These statements may be placed in the list that declares local variables.

Module `ModifiedNodalForces` has a very similar structure and logic and need not be described in detail. It is important to note, however, that for homogeneous BCs the two module are independent of each other and may be called in any order. On the other hand, if there were nonzero prescribed displacements the force modification must be done *before* the stiffness modification. This is because stiffness coefficients that are cleared in the latter are needed for modifying the force vector.

The test statements at the bottom of Figure 22.2 are chosen to illustrate another feature of *Mathematica*: the use of the `Array` function to generate subscripted symbolic arrays of one and two dimensions. The test output should be self explanatory and is not shown here. Both the force vector and its modified form are printed as row vectors to save space.

§22.3.3. Internal Force Recovery

Module `PlaneTrussIntForces` listed in Figure 22.3 computes the internal forces (axial forces) in all truss members. The first five arguments are the same as for the assembler routine described previously. The last argument, `u`, contains the computed node displacements arranged as a flat, one dimensional list:

$$\{\texttt{ux1, uy1 ... uxn, uyn}\} \tag{22.1}$$

`PlaneTrussIntForces` makes use of `PlaneBar2IntForce`, which computes the internal force in an *individual member*. `PlaneBar2IntForce` is similar in argument sequence and logic to `PlaneBar2Stiffness` of Figure 22.1. The first four arguments are identical. The last argument, `ue`, contains the list of the four element node displacements in the global system. The logic of the recovery module is straightforward and follows the method outlined in §3.2.

The statements at the bottom of Figure 22.3 test the internal force recovery for the example truss of Chapters 2-3, a and should return forces of 0,, -1 and $2\sqrt{2}$ for members 1, 2 and 3, respectively.

§22.3.4. Graphic Modules

Graphic modules that support preprocessing are placed in Cells 4A, 4B and 4C of the *Mathematica* notebook. These plot unlabeled elements, elements and nodes with labels, and boundary conditions.

Graphic modules that support postprocessing are placed in Cells 5A and 5B of the Notebook. These plot deformed shapes and axial stress levels in color.

These modules are not listed since they are still undergoing modifications at the time of this writing. One unresolved problem is to find a way for absolute placement of supported nodes for correct deflected-shape animations.

```
PlaneTrussIntForces[nodcoor_,elenod_,elemat_,elefab_,
  eleopt_,u_]:= Module[{numele=Length[elenod],
  numnod=Length[nodcoor],e,eNL,eftab,ni,nj,i,
  ncoor,mprop,fprop,opt,ue,p},
  p=Table[0,{numele}]; ue=Table[0,{4}];
  For [e=1, e<=numele, e++,
      eNL=elenod[[e]]; {ni,nj}=eNL;
      eftab={2*ni-1,2*ni,2*nj-1,2*nj};
      ncoor={nodcoor[[ni]],nodcoor[[nj]]};
      mprop=elemat[[e]]; fprop=elefab[[e]]; opt=eleopt;
      For [i=1,i<=4,i++, ii=eftab[[i]]; ue[[i]]=u[[ii]]];
      p[[e]]=PlaneBar2IntForce[ncoor,mprop,fprop,opt,ue]
      ];
  Return[p]
];
PlaneBar2IntForce[ncoor_,mprop_,fprop_,opt_,ue_]:= Module[
  {x1,x2,y1,y2,x21,y21,Em,Gm,rho,alpha,A,numer,LL,pe},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,rho,alpha}=mprop; {A}=fprop; {numer}=opt;
  (*If [numer,{x21,y21,Em,A}=N[{x21,y21,Em,A}]];*)
  LL=x21^2+y21^2;
  pe=Em*A*(x21*(ue[[3]]-ue[[1]])+y21*(ue[[4]]-ue[[2]]))/LL;
  Return[pe]
];
nodcoor={{0,0},{10,0},{10,10}}; elenod= {{1,2},{2,3},{1,3}};
elemat= Table[{100,0,0,0},{3}]; elefab= {{1},{1/2},{2*Sqrt[2]}};
eleopt= {True}; u={0,0,0,0,0.4,-0.2};
p=PlaneTrussIntForces[nodcoor,elenod,elemat,elefab,eleopt,u];
Print["Int Forces of Example Truss:"];
Print[p];
```

FIGURE 22.3. Calculation of truss internal forces, with test statements in red.

§22.4. A Bridge Truss Example

The driver program in Cell 6 defines and runs the analysis of the 6-bay bridge truss problem defined in Figure 22.4. This truss has 12 nodes and 17 elements. It is fixed at node 1 and on rollers at node 12.

The driver is listed in Figures 22.5 and 22.6. It begins by defining the problem through specification of the following data structures:

`NodeCoordinates` Same configuration as `nodcoor`

`ElemNodeLists` Same configuration as `elenod`

`ElemMaterials` Same configuration as `elemat`

`ElemFabrication` Same configuration as `elefab`. This list is built up from four repeating cross sectional areas: `Abot`, `Atop`, `Abat` and `Adia`, for the areas of bottom longerons, top longerons, battens and diagonals, respectively.

`ProcessOptions` Same configuration as `eleopt`

`FreedomValue`. This is a two-dimensional list that specifies freedom values, node by node. It is initialized to zero on creation, then the value of nonzero applied loads is set at nodes 3, 5, 7, 9 and 11. For example `FreedomValue[[7]] = {0,-16}` specifies $f_{x7} = 0$ and $f_{y7} = -16$.

`FreedomTag`. This is a two-dimensional list that specifies, for each nodal freedom, whether the value

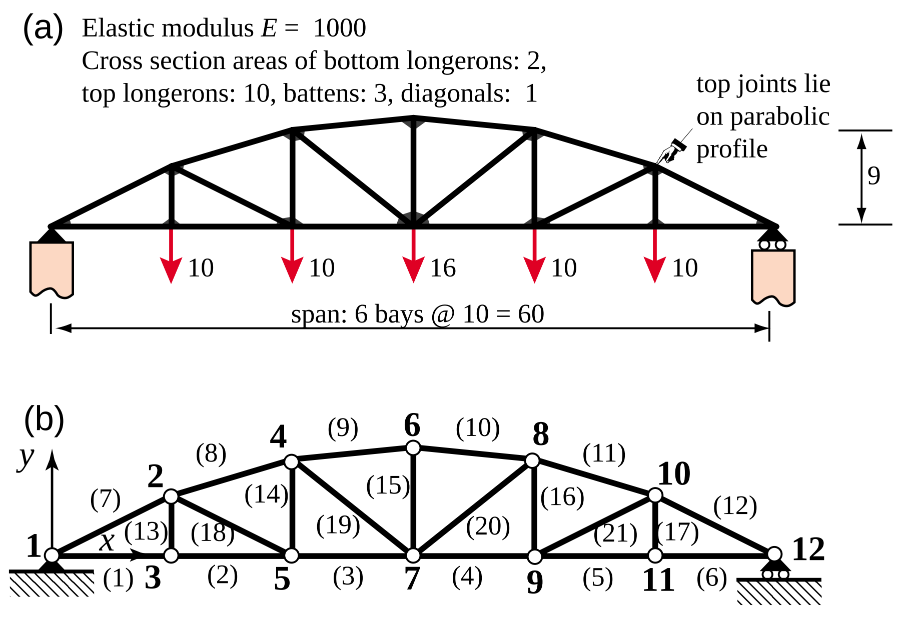

FIGURE 22.4. Six-bay bridge truss used as example problem: (a) truss structure showing supports and applied loads; (b) finite element idealization as pin-jointed truss.

of the force (tag 0) or displacement (tag 1) is prescribed. It is initialized to zero on creation, then the support tags at nodes 1 (fixed) and 12 (horizontal roller) are set. For example, `FreedomTag[[1]] = {1,1}` says that both node displacement components u_{x1} and u_{y1} of node 1 are specified.

Processing commands are listed in Figure 22.6. The master stiffness matrix is assembled by the module listed in Figure 22.1. The stiffness matrix is placed in `K`. The applied force vector stored as a one-dimensional list, is placed in `f`, which results from the application of built-in function `Flatten` to `Freedom Value`.

The prescribed degree-of-freedom array `pdof` is constructed by scanning `FreedomTag` for nonzero values. For this problem `pdof={1,2,23}`. The displacement boundary conditions are applied by the modules of Figure 22.2, which return the modified master stiffness `Kmod` and the modified node force vector `fmod`. Note that the modified stiffness matrix is stored into `Kmod` rather than `K` to save the original form of the master stiffness for the recovery of reaction forces later.

The complete displacement vector is obtained by the matrix calculation

```
u=LinearSolve[Kmod,fmod]
```
(22.2)

which takes advantage of the built-in linear solver provided by `Mathematica`.

The remaining calculations recover the node vector including reactions by the matrix-vector multiply `f = K.u` (recall that `K` contains the unmodified master stiffness matrix). The member internal forces `p` are obtained through the module listed in Figure 22.3. The program prints `u`, `f` and `p` as row vectors to conserve space.

Running the program of Figures 22.5–6 produces the output shown in Figure 22.7. Output plot results are collected in Figure 22.8.

```
ClearAll[];
NodeCoordinates={{0,0},{10,5},{10,0},{20,8},{20,0},{30,9},
         {30,0},{40,8},{40,0},{50,5},{50,0},{60,0}};
ElemNodeLists= {{1,3},{3,5},{5,7},{7,9},{9,11},{11,12},
         {1,2},{2,4},{4,6},{6,8},{8,10},{10,12},
         {2,3},{4,5},{6,7},{8,9},{10,11},
         {2,5},{4,7},{7,8},{9,10}};
numnod=Length[NodeCoordinates];
numele=Length[ElemNodeLists]; numdof=2*numnod;
ElemMaterial= Table[{1000,0,0,0},{numele}];
Abot=2; Atop=10; Abat=3; Adia=1;
ElemFabrication=Join[Table[{Abot},{6}],Table[{Atop},{6}],
    Table[{Abat},{5}],Table[{Adia},{4}]];
ProcessOptions= {True}; aspect=0;
PlotLineElements[NodeCoordinates,ElemNodeLists,aspect,
   "test mesh"];
PlotLineElementsAndNodes[NodeCoordinates,ElemNodeLists,aspect,
   "test mesh with elem & node labels",{True,0.12},{True,0.05}];

FreedomTag=FreedomValue=Table[{0,0},{numnod}];
FreedomValue[[3]]={0,-10}; FreedomValue[[5]]={0,-10};
FreedomValue[[7]]={0,-16};
FreedomValue[[9]]={0,-10}; FreedomValue[[11]]={0,-10};
Print["Applied node forces="]; Print[FreedomValue];
FreedomTag[[1]]= {1,1};     (* fixed node 1 *)
FreedomTag[[numnod]]={0,1}; (* hroller @ node 12 *)
```

FIGURE 22.5. Driver for analysis of the bridge truss: preprocessing.

```
f=Flatten[FreedomValue];
K=PlaneTrussMasterStiffness[NodeCoordinates,
  ElemNodeLists,ElemMaterial,ElemFabrication,ProcessOptions];
pdof={}; For[n=1,n<=numnod,n++, For[j=1,j<=2,j++,
            If[FreedomTag[[n,j]]>0, AppendTo[pdof,2*(n-1)+j]]]];
Kmod=ModifiedMasterStiffness[pdof,K];
fmod=ModifiedNodeForces [pdof,f];
u=LinearSolve[Kmod,fmod];  u=Chop[u];
Print["Computed Nodal Displacements:"]; Print[u];
f=Simplify[K.u]; f=Chop[f];
Print["External Node Forces Including Reactions:"]; Print[f];
p=PlaneTrussIntForces[NodeCoordinates,ElemNodeLists,
    ElemMaterial,ElemFabrication,eleopt,u]; p=Chop[p];
sigma=Table[p[[i]]/ElemFabrication[[i,1]],{i,1,numele}];
Print["Internal Member Forces:"]; Print[p];
PlotTrussDeformedShape[NodeCoordinates,ElemNodeLists,u,
    1.0,aspect,"Deformed shape"];
PlotAxialStressLevel[NodeCoordinates,ElemNodeLists,sigma,
    1.0,aspect,"Axial stress level"];
```

FIGURE 22.6. Driver for analysis of the bridge truss: processing and postprocessing.

```
Applied node forces:
{{0, 0}, {0, 0}, {0, -10}, {0, 0}, {0, -10}, {0, 0},
 {0, -16}, {0, 0}, {0, -10}, {0, 0}, {0, -10}, {0, 0}}

Computed Nodal Displacements:
{0, 0, 0.809536, -1.7756, 0.28, -1.79226, 0.899001,
 -2.29193, 0.56, -2.3166, 0.8475, -2.38594,
 0.8475, -2.42194, 0.795999, -2.29193, 1.135, -2.3166,
 0.885464, -1.7756, 1.415, -1.79226, 1.695, 0}

External Node Forces Including Reactions:
{0, 28., 0, 0, 0, -10., 0, 0, 0, -10., 0, 0, 0,
 -16., 0, 0, 0, -10., 0, 0, 0, -10., 0, 28.}

Internal Member Forces:
{56., 56., 57.5, 57.5, 56., 56., -62.6099,
 -60.0318, -60.2993, -60.2993,
 -60.0318, -62.6099, 10., 9.25, 12., 9.25,
 10., 1.67705, 3.20156, 3.20156, 1.67705}
```

FIGURE 22.7. Printed output from the bridge truss analysis.

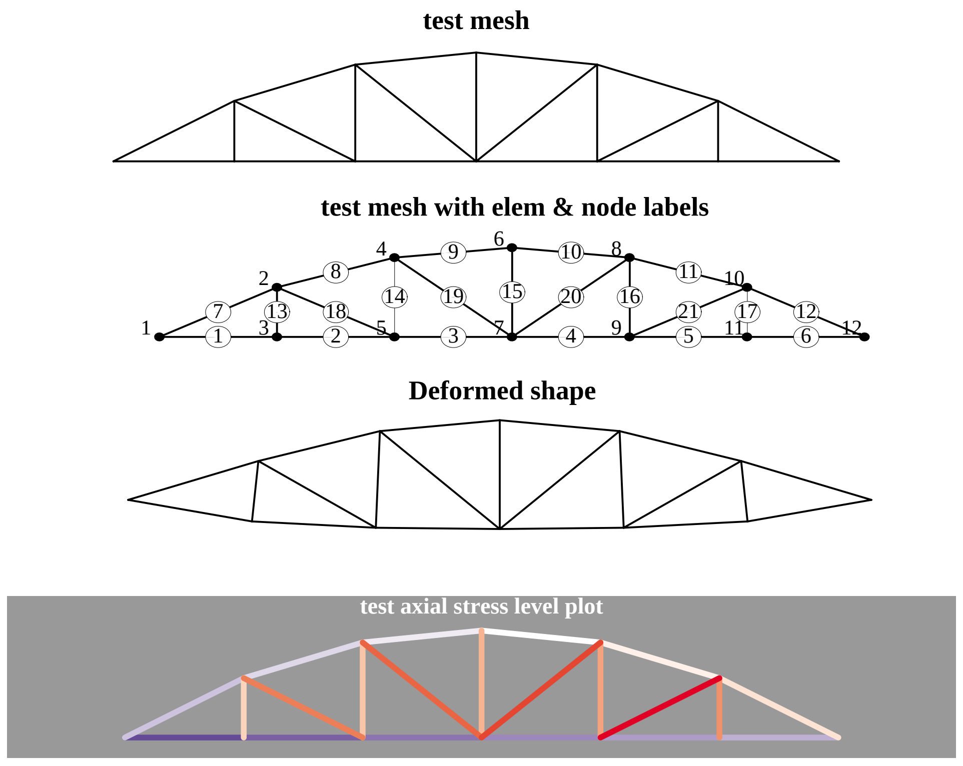
test mesh
test mesh with elem & node labels
Deformed shape
test axial stress level plot

FIGURE 22.8. Graphics output from the bridge truss analysis.

Homework Exercises for Chapter 22
FEM Programs for Plane Trusses and Frames

EXERCISE 22.1

Placeholder.

EXERCISE 22.2 [C:25] Using the `PlaneTruss.nb` Notebook posted on the web site as guide, complete the `PlaneFrame.nb` Notebook that implements the analysis of an arbitrary plane frame using the plane beam stiffness presented in Chapter 21. To begin this homework, download the `PlaneFrame.nb` Notebook file from the web site.

The modification involves selected Cells of the Notebook, as described below. For additional details refer to the Notebook; some of the modifications involve only partially filled Modules.

Cell 1: Assembler. The element stiffness routine is the beam-column stiffness presented in Chapter 21. This module, called PlaneBeamColumn2Stiffness, is included in Cell 1 of the `PlaneFrame.nb` notebook linked from the Chapter 22 index.[1] In modifying the assembler module, remember that there are three degrees of freedom per node: u_{xi}, u_{yi} and θ_i, not just two.

The incomplete assembler is shown in Figure E22.1. The test statements are shown in red after the module. The lower cell shows the test output produced by a correctly completed module.

Cell 2: BC Applicator. No modifications are necessary. The two modules should work correctly for this problem since they dont assume anything about freedoms per nodes. The same test statements can be kept.

Cell 3: Internal Force Recovery. Both modules require modifications because the beam has 3 internal forces: (i) the axial force (which is recovered exactly as for bars), (ii) the bending moment $m_i = EI\kappa_i$ at end node i, and (iii) the bending moment $m_j = EI\kappa_j$ at end node j.[2] Furthermore the element displacement vector has six degrees of freedom, not just four. Test the modules on a simple beam problem.

The incomplete internal force module is shown in Figure E22.2. The test statements are shown in red after the module. The lower cell shows the test output produced by a correctly completed module.

Cell 4-5: Graphic Modules. These are now provided ready to use, and should not be touched.

Cell 6: Driver Program: use this for Exercise 22.3. Do not touch for this one.

As solution to this Exercise return a listing of the completed Cells 1 and 3, along with the output of the test statements for those cells.

EXERCISE 22.3

[C:25] Use the program developed in the previous Exercise to analyze the plane frame shown in Figure E22.3. The frame is fixed[3] at A, B and C. It is loaded by a downward point load P at G and by a horizontal point load $\frac{1}{2}P$ at D. All members have the same cross section $a \times a$ for simplicity, and the material is the same.

[1] If you want to extract an individual cell from a Notebook to work on a separate file (a good idea for working on groups) select the cell, then pick SaveSelectionAs -> Plain Text from the Edit menu, give it a file name in the pop-up dialogue box, and save it.

[2] Two end moments are required because the beam element used here has linearly varying curvatures and hence moments. To recover the end moments refer to the pertinent equations in Chapter 13. The steps are as follows. For element e recover the transverse displacements $\bar{u}_{yi}$ and $\bar{u}_{yj}$ in the local element system $\bar{x}^{(e)}$, $\bar{y}^{(e)}$. Complete these with rotations θ_i and θ_j (which do not need to be transformed) to form the 4×1 beam local node displacement vector. Then apply the curvature-displacement relation (13.13) at $\xi = -1$ and $\xi = 1$ to get the two end curvatures $\kappa_i^{(e)}$ and $\kappa_j^{(e)}$, respectively. Finally multiply those curvatures by $E^{(e)} I_{zz}^{(e)}$ to get the bending moments.

[3] For a plane frame, a fixed condition, also known as clamped condition, means that the x, y displacements and the rotation about z are zero.

```
(* PlaneBeamColumn2Stiffness module goes here *)

PlaneFrameMasterStiffness[nodcoor_,elenod_,
  elemat_,elefab_,eleopt_]:=Module[
  {numele=Length[elenod],numnod=Length[nodcoor],
  e,eNL,eftab,ni,nj,i,j,ncoor,mprop,fprop,opt,Ke,K},
  K=Table[0,{3*numnod},{3*numnod}];
  For [e=1, e<=numele, e++,
      (* missing statements *)
       Ke=PlaneBeamColumn2Stiffness[ncoor,mprop,fprop,opt];
      (* missing statements *)
      ];
  Return[K]
  ];
nodcoor={{0,0},{10,0},{10,10}};
elenod= {{1,2},{2,3},{1,3}}; elemat= Table[{100,0,0,0},{3}];
elefab= {{1,10},{1/2,10},{2*Sqrt[2],10}}; eleopt= {True};
K=PlaneFrameMasterStiffness[nodcoor,elenod,elemat,elefab,eleopt];
Print["Master Stiffness of Example Frame:"];
Print[K//MatrixForm]; Print[Chop[Eigenvalues[K]]];
```

```
Master Stiffness of Example Frame:
 22.1213   7.87868  -21.2132  -10.   0.    0.    -12.1213  -7.87868  -21.2132
 7.87868   24.1213  81.2132    0.   -12.   60.   -7.87868  -12.1213  21.2132
-21.2132   81.2132  682.843    0.   -60.  200.   21.2132   -21.2132  141.421
  -10.       0.       0.      22.    0.   -60.   -12.        0.       -60.
   0.       -12.     -60.      0.   17.   -60.    0.       -5.        0.
   0.        60.     200.    -60.  -60.   800.   60.        0.       200.
-12.1213  -7.87868  21.2132  -12.    0.    60.   24.1213   7.87868   81.2132
-7.87868  -12.1213  -21.2132   0.   -5.    0.    7.87868   17.1213  -21.2132
-21.2132   21.2132  141.421  -60.    0.   200.   81.2132  -21.2132  682.843

{1121.7, 555.338, 531.652, 46.6129, 22.4971, 14.366, 0, 0, 0}
```

FIGURE E22.1. The incomplete assembler module for Exercise 22.2, with test statements in red. Module PlaneBeamColumn2Stiffness, which goes in the upper portion of the cell is omitted to save space; that module was listed in Figure 21.9. Output cell results are produced by a correct module.

The SI physical units to be used are: mm for lengths, N for forces, and MPa=N/mm^2 for elastic moduli. For the calculations use the following numerical data: $L = 10{,}000$ mm (10 m), $H = 6{,}000$ (6 m), $a = 500$ mm (0.5 m), $P = 4{,}800$ N, $E = 35{,}000$ MPa (high strength concrete). The member cross section area is $A = a^2$, and the flexural moment of inertia about the neutral axis is $I_{zz} = a^4/12$.

The recommended finite element discretization of two elements per member is shown in Figure E22.4.

As solution to this exercise list the driver program you used and the displacement and internal force outputs. The results of primary interest are:

1. The horizontal displacement at D, and the vertical displacement at G (mm).
2. The axial forces (N) in columns AD, BE and CF, with appropriate sign[4].
3. The maximum bending moment (N.mm) over the floor members DE and EF, and the maximum bending moment in the columns AD, BE and CF.[5]

Provide deformed shape plot (but not the animation) and the frame stress level plots as part of the homework results. Note: the Notebook on the web contains some of the actual plots produced by the complete Notebook,

[4] Plus for tension, minus for compression

[5] The bending moment varies linearly over each element. It should be continuous at all joints except E.

```
PlaneFrameIntForces[nodcoor_,elenod_,elemat_,elefab_,
  eleopt_,u_]:= Module[{numele=Length[elenod],
  numnod=Length[nodcoor],e,eNL,eftab,ni,nj,i,
  ncoor,mprop,fprop,opt,ue,p},
  p=Table[0,{numele}]; ue=Table[0,{6}];
  For [e=1, e<=numele, e++,
    eNL=elenod[[e]]; {ni,nj}=eNL;
    ncoor={nodcoor[[ni]],nodcoor[[nj]]};
    mprop=elemat[[e]]; fprop=elefab[[e]]; opt=eleopt;
    eftab={3*ni-2,3*ni-1,3*ni,3*nj-2,3*nj-1,3*nj};
    For [i=1,i<=6,i++, ii=eftab[[i]]; ue[[i]]=u[[ii]]];
    p[[e]]=PlaneBeamColumn2IntForces[ncoor,mprop,fprop,opt,ue]
    ];
  Return[p]
];
PlaneBeamColumn2IntForces[ncoor_,mprop_,fprop_,opt_,ue_]:=
 Module[{x1,x2,y1,y2,x21,y21,Em,Gm,rho,alpha,A,Izz,num,LL,L,
  dvy,cv1,cv2,pe=Table[0,{3}]},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,rho,alpha}=mprop; {A,Izz}=fprop; {num}=opt;
  If [num,{x21,y21,Em,A,Izz}=N[{x21,y21,Em,A,Izz}]];
   (* missing statements for Exercise 22.2 *)
  Return[pe]
];
ClearAll[L,Em,A1,A2,Izz1,Izz2,Izz3,uz1,uz2,uz2,uz3];
nodcoor={{0,0},{L,0},{L,L}}; elenod= {{1,2},{2,3},{1,3}};
elemat= Table[{Em,0,0,0},{3}];
elefab= {{A1,Izz1},{A2,Izz2},{A3,Izz3}}; eleopt= {False};
u={0,uy1,0, 0,uy2,0, 0,uy3,0};
p=PlaneFrameIntForces[nodcoor,elenod,elemat,elefab,eleopt,u];
Print["Int Forces of Example Frame:"]; Print[p];
```

```
Int Forces of Example Frame:
```

$$\left\{\left\{0, \frac{6\,\text{Em Izz1}\,(-\text{uy1}+\text{uy2})}{L^2}, -\frac{6\,\text{Em Izz1}\,(-\text{uy1}+\text{uy2})}{L^2}\right\}, \left\{\frac{\text{A2 Em}\,(-\text{uy2}+\text{uy3})}{L}, 0, 0\right\}, \left\{\frac{\text{A3 Em}\,(-\text{uy1}+\text{uy3})}{2\,L}, \frac{3\,\text{Em Izz3}\,(-\text{uy1}+\text{uy3})}{\sqrt{2}\,L^2}, -\frac{3\,\text{Em Izz3}\,(-\text{uy1}+\text{uy3})}{\sqrt{2}\,L^2}\right\}\right\}$$

FIGURE E22.2. Figure E22.2. Incomplete element internal force recovery module for Exercise 22.2. Test statements in red. Results in output cell are those produced by a correct module.

to be used as targets.

Partial answers for this exercise:

Largest vertical displacement: -0.232 mm (⇓) at node 4

Largest negative bending moment: -6.8×10^6 N.mm, at node 5 of element (4)

Axial forces: -2651 N over (3) and (4)

EXERCISE 22.4 [D:20] Explain why the solution given by the FEM model of Figure E22.3 is exact for the Bernoulli-Euler bending model; that is, cannot be improved by subdividing each element into more elements.

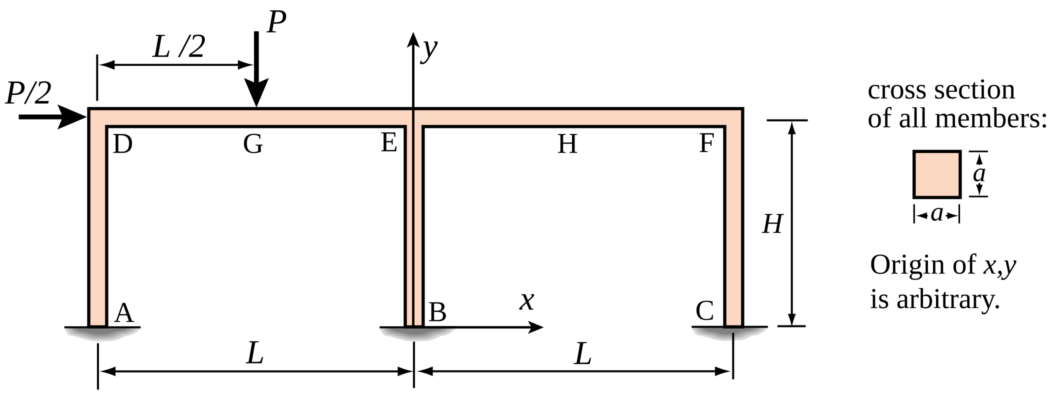

FIGURE E22.3. Plane frame structure for Exercise 22.3.

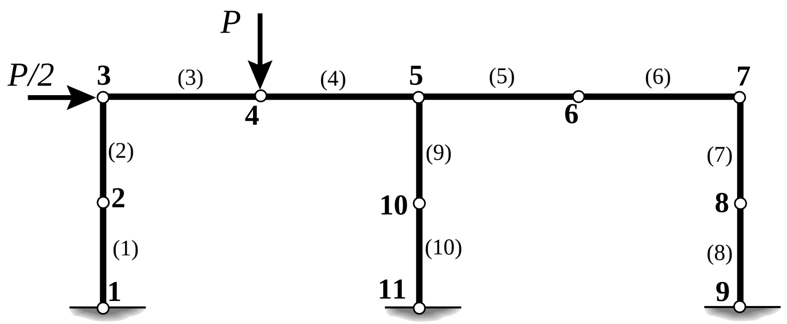

FIGURE E22.4. Recommended FEM discretization for the plane frame of the previous figure.