ECI211 Final Project

Sam Mish

December 2017

1 Introduction

The final project for this class is to use matrix structural analysis theory to develop a program for analyzing 3D frame structures. In this report, I will start with a brief overview of how this program runs, then provide a more detailed description of its core parts, and then show some examples problems. At the end of the report, the numerical values of output quantities (displacements, member forces, etc) will be tabulated for verification purposes.

2 Program Overview

The general structure of this program is as follows:

1. Define input mesh and material/section properties
    - locations of nodes, and their connectivities
    - E, Young's modulus for material
    - ν, Poisson's ratio for material
    - ρ, mass density for material
    - I_{yy}, I_{zz}, cross section second moments of area
    - A, cross sectional area
2. Compute local mass, stiffness, and transformation matrices for each beam
3. Assemble global mass and stiffness matrices and force vector
4. Solve for system modes or unconstrained displacements
5. Output
    - compute nodal reactions at supports
    - compute member forces
    - visualize

The user can choose to define these inputs by reading from files, or defining them directly in the Mathematica notebook, by explicit entry, or by scripting. Similarly, the output options are left up to the user, to avoid calculating unneeded results.

3 Function Reference

```
makeBeam[{p_, q_}, nodeIds_, E_, ν_, A_, Iy_, Iz_, ρ_] := Module[
  {
   L = FullSimplify[Norm[q - p]],
   J = Iy + Iz,
   Γ = FullSimplify[transformation[Normalize[q - p]]]
  },
  <|
   "Γ" → Γ,
   "γ" → Γ[[1 ;; 3, 1 ;; 3]],
   "M" -> FullSimplify[beamMass[ρ, A, J, L]],
   "K" → FullSimplify[beamStiffness[E, ν, A, Iy, Iz, L]],
   "endpoints" → {p, q},
   "E" → E,
   "ν" → ν,
   "ρ" → ρ,
   "A" → A,
   "Iy" → Iy,
   "Iz" → Iz,
   "L" → L,
   "nodeIds" → nodeIds,
   "dofIds" → dofExpand[nodeIds, 6],
   "displacement" → ConstantArray[0.0, 12]
   |>
 ]
```

makeBeam: this function takes in geometry information, as well as material and section properties, and creates an immutable container object (which Mathematica refers to as an "association") that stores local matrices, and information about connectivity to nodes.

```
beamMass[ρ_, A_, J_, L_] :=
```

$$\frac{\rho A L}{420}\begin{pmatrix} 140 & 0 & 0 & 0 & 0 & 0 & 70 & 0 & 0 & 0 & 0 & 0 \\ 0 & 156 & 0 & 0 & 0 & 22L & 0 & 54 & 0 & 0 & 0 & -13L \\ 0 & 0 & 156 & 0 & -22L & 0 & 0 & 0 & 54 & 0 & 13L & 0 \\ 0 & 0 & 0 & 140\frac{J}{A} & 0 & 0 & 0 & 0 & 0 & 70\frac{J}{A} & 0 & 0 \\ 0 & 0 & -22L & 0 & 4L^2 & 0 & 0 & 0 & -13L & 0 & -3L^2 & 0 \\ 0 & 22L & 0 & 0 & 0 & 4L^2 & 0 & 13L & 0 & 0 & 0 & -3L^2 \\ 70 & 0 & 0 & 0 & 0 & 0 & 140 & 0 & 0 & 0 & 0 & 0 \\ 0 & 54 & 0 & 0 & 0 & 13L & 0 & 156 & 0 & 0 & 0 & -22L \\ 0 & 0 & 54 & 0 & -13L & 0 & 0 & 0 & 156 & 0 & 22L & 0 \\ 0 & 0 & 0 & 70\frac{J}{A} & 0 & 0 & 0 & 0 & 0 & 140\frac{J}{A} & 0 & 0 \\ 0 & 0 & 13L & 0 & -3L^2 & 0 & 0 & 0 & 22L & 0 & 4L^2 & 0 \\ 0 & -13L & 0 & 0 & 0 & -3L^2 & 0 & -22L & 0 & 0 & 0 & 4L^2 \end{pmatrix};$$

```
beamStiffness[E_, ν_, A_, Iy_, Iz_, L_] := Module[{G = E/(1 + 2 ν), J = Iy + Iz},
```

$$\frac{E}{L}\begin{pmatrix} A & 0 & 0 & 0 & 0 & 0 & -A & 0 & 0 & 0 & 0 & 0 \\ 0 & 12\frac{Iz}{L^2} & 0 & 0 & 0 & 6\frac{Iz}{L} & 0 & -12\frac{Iz}{L^2} & 0 & 0 & 0 & 6\frac{Iz}{L} \\ 0 & 0 & 12\frac{Iy}{L^2} & 0 & -6\frac{Iy}{L} & 0 & 0 & 0 & -12\frac{Iy}{L^2} & 0 & -6\frac{Iy}{L} & 0 \\ 0 & 0 & 0 & \frac{GJ}{E} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{E} & 0 & 0 \\ 0 & 0 & -6\frac{Iy}{L} & 0 & 4Iy & 0 & 0 & 0 & 6\frac{Iy}{L} & 0 & 2Iy & 0 \\ 0 & 6\frac{Iz}{L} & 0 & 0 & 0 & 4Iz & 0 & -6\frac{Iz}{L} & 0 & 0 & 0 & 2Iz \\ -A & 0 & 0 & 0 & 0 & 0 & A & 0 & 0 & 0 & 0 & 0 \\ 0 & -12\frac{Iz}{L^2} & 0 & 0 & 0 & -6\frac{Iz}{L} & 0 & 12\frac{Iz}{L^2} & 0 & 0 & 0 & -6\frac{Iz}{L} \\ 0 & 0 & -12\frac{Iy}{L^2} & 0 & 6\frac{Iy}{L} & 0 & 0 & 0 & 12\frac{Iy}{L^2} & 0 & 6\frac{Iy}{L} & 0 \\ 0 & 0 & 0 & -\frac{GJ}{E} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{E} & 0 & 0 \\ 0 & 0 & -6\frac{Iy}{L} & 0 & 2Iy & 0 & 0 & 0 & 6\frac{Iy}{L} & 0 & 4Iy & 0 \\ 0 & 6\frac{Iz}{L} & 0 & 0 & 0 & 2Iz & 0 & -6\frac{Iz}{L} & 0 & 0 & 0 & 4Iz \end{pmatrix}$$

```
]
```

beamMass and beamStiffness: these functions use the properties of the beam to calculate the local 12-by-12 mass and stiffness matrices for a 3D Bernoulli beam element. These do not take into account any sort of nonlinear behavior.

```
transformation[dir_] := Module[{y = {0, 1, 0}, γ},
  γ = FullSimplify[If[Abs[dir[[2]]] > 0.99,
     ({{0, Sign[dir[[2]]], 0},
       {-Sign[dir[[2]]], 0, 0},
       {0, 0, 1}}),
     {dir, Cross[Normalize[Cross[dir, y]], dir], Normalize[Cross[dir, y]]}]
   ];
  ArrayFlatten[{{γ, 0, 0, 0},
    {0, γ, 0, 0},
    {0, 0, γ, 0},
    {0, 0, 0, γ}}]
 ]
```

transformation: this function computes a proper- orthogonal matrix corresponding to the mapping between a beam's local coordinate system and the global coordinate system. This is used in the global matrix assembly process, and also for getting member forces and post-processing.

```
assembleGlobalMatrices[beams_, ndof_] := Module[{
   Mglobal = ConstantArray[0.0, {ndof, ndof}],
   Kglobal = ConstantArray[0.0, {ndof, ndof}]
  },
  Do[
   Mglobal[[b["dofIds"], b["dofIds"]]] += Transpose[b["T"]].b["M"].b["T"];
   Kglobal[[b["dofIds"], b["dofIds"]]] += Transpose[b["T"]].b["K"].b["T"];
   ,
   {b, beams}
  ];
  {Mglobal, Kglobal}
 ]
```

assembleGlobalMatrices uses the information stored in a list of beam objects to transform and add individual contributions to global mass and stiffness matrices.

```
(* Hermite Polynomials *)
ψ[x_, L_] := {
  InterpolatingPolynomial[{{0, 1, 0}, {L, 0, 0}}, x],
  InterpolatingPolynomial[{{0, 0, 1}, {L, 0, 0}}, x],
  InterpolatingPolynomial[{{0, 0, 0}, {L, 1, 0}}, x],
  InterpolatingPolynomial[{{0, 0, 0}, {L, 0, 1}}, x]
 }
```

$$\Psi[x\_,\ L\_] := \text{Transpose}\left[\begin{pmatrix} 1-\frac{x}{L} & 0 & 0 \\ 0 & \psi[x, L][[1]] & 0 \\ 0 & 0 & \psi[x, L][[1]] \\ 0 & 0 & 0 \\ 0 & 0 & -\psi[x, L][[2]] \\ 0 & \psi[x, L][[2]] & 0 \\ \frac{x}{L} & 0 & 0 \\ 0 & \psi[x, L][[3]] & 0 \\ 0 & 0 & \psi[x, L][[3]] \\ 0 & 0 & 0 \\ 0 & 0 & -\psi[x, L][[4]] \\ 0 & \psi[x, L][[4]] & 0 \end{pmatrix}\right];$$

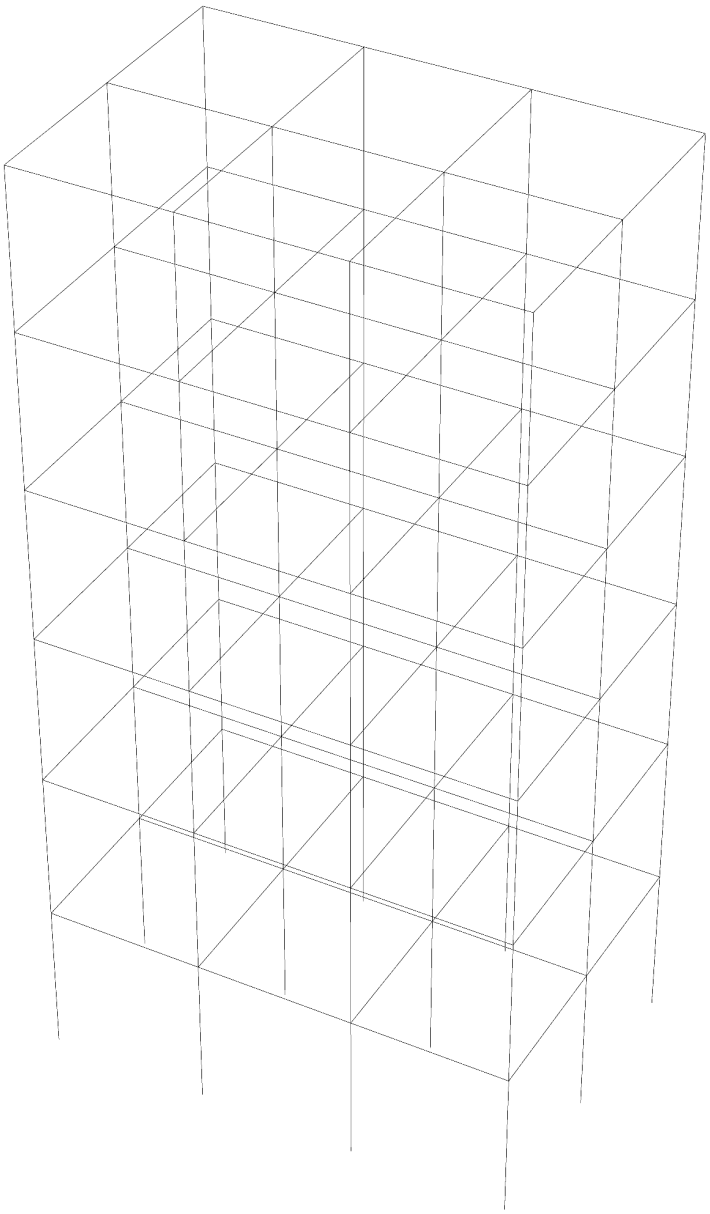
These functions are not meant to be accessed directly by the user, but they enable the use of graphics routines that can accurately depict the deformed geometry of beams. Interestingly, this Ψ operator is also the shape function matrix that is used to determine nodal forces from distributed loads. For example, for a uniform distributed load, we get:

```
Transpose[Integrate[Ψ[x, L], {x, 0, L}]] // MatrixForm
```

$$\begin{pmatrix} \frac{L}{2} & 0 & 0 \\ 0 & \frac{L}{2} & 0 \\ 0 & 0 & \frac{L}{2} \\ 0 & 0 & 0 \\ 0 & 0 & -\frac{L^2}{12} \\ 0 & \frac{L^2}{12} & 0 \\ \frac{L}{2} & 0 & 0 \\ 0 & \frac{L}{2} & 0 \\ 0 & 0 & \frac{L}{2} \\ 0 & 0 & 0 \\ 0 & 0 & \frac{L^2}{12} \\ 0 & -\frac{L^2}{12} & 0 \end{pmatrix}$$

```
{nodes, elems, fixed, free} = multiStoryFrame[{0, 1, 2, 3}, {0, 1, 2}, {0, 1, 2, 3, 4, 5, 6}];
```

multiStoryFrame takes in 3 lists of coordinates, corresponding to locations along the x, y, and z directions, and generates nodal positions on that grid, along with connectivities between them. This can be useful for quickly creating simple meshes, without requiring large inputfiles. The single line of code above produces the structure below:



4 Analysis Examples

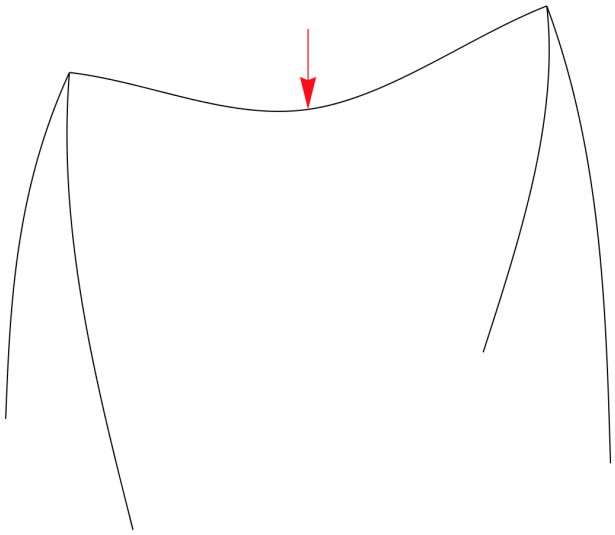
For the swingset problem given to us, the analysis is performed by running the following code:

$$\text{nodes} = \begin{pmatrix} 1000 & 0 & 0 \\ 0 & 0 & 2500 \\ -1000 & 0 & 0 \\ 0 & 1500 & 2500 \\ 1000 & 3000 & 0 \\ 0 & 3000 & 2500 \\ -1000 & 3000 & 0 \end{pmatrix}; \quad \text{elems} = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 2 & 4 \\ 4 & 6 \\ 5 & 6 \\ 6 & 7 \end{pmatrix};$$

```
ndof = Length[nodes] * 6;
fixed = {1, 2, 3, 13, 14, 15, 25, 26, 27, 37, 38, 39};
free = Complement[Table[i, {i, 1, ndof}], fixed];

beams = Table[makeBeam[nodes[[el]], el, E, ν, A, Iy, Iz, ρ], {el, elems}];
{Mglobal, Kglobal} = assembleGlobalMatrices[beams, ndof];
fglobal = -4.5 IdentityMatrix[ndof][[21]];
uglobal = ConstantArray[0.0, ndof];
reactions = ConstantArray[0.0, ndof];
uglobal[[free]] = LinearSolve[FullSimplify[Chop[Kglobal]][[free, free]], fglobal[[free]]];
reactions[[fixed]] = Kglobal[[fixed, free]].uglobal[[free]];
reactions[[free]] = fglobal[[free]];
```

The specific reactions and member forces are given at the end of this report, with deformed configuration (scaled for illustrative purposes):



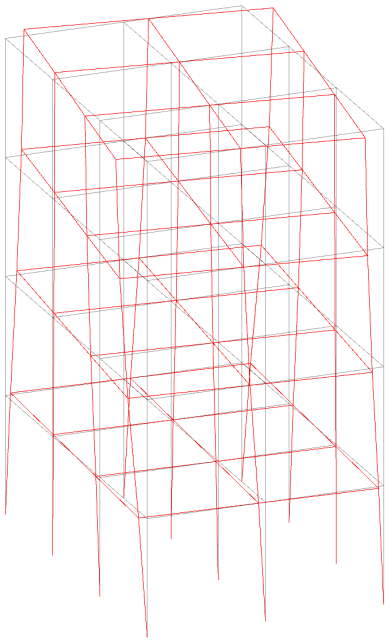
Below is an example of how to calculate the mode shapes for a structure:

```
(* get the normal modes for a multi-story frame structure *)
E = 200;
ν = 0.3;
A = 1430;
Iy = 1.26 * 10^6;
Iz = 1.26 * 10^6;
ρ = 8050/10^9;
L = 1000;
{nodes, elems, fixed, free} = multiStoryFrame[{0, 1, 2, 3} L, {0, 1, 2} L, {0, 1, 2, 3, 4} L];
ndof = Length[nodes] * 6;
beams = Table[makeBeam[nodes[[el]], el, E, ν, A, Iy, Iz, ρ], {el, elems}];
{Mglobal, Kglobal} = assembleGlobalMatrices[beams, ndof];

ϕ = ConstantArray[0.0, {Length[free], ndof}];
ϕ[[All, free]] = Reverse[Eigenvectors[{Kglobal[[free, free]], Mglobal[[free, free]]}]];
```

After calculating the mode shapes, ϕ, the user can plot them alongside the undeformed configuration:

```
mode = 3;
Graphics3D[{
  Opacity[0.4], Table[drawBeamSimple[b], {b, beams}],
  Opacity[1.0], Red, Table[drawBeamSimple[b, 1000 ϕ[[mode, b["dofIds"]]]], {b, beams}]
 }, Boxed → False
]
```



5 Conclusion

This small Mathematica notebook is capable of analyzing simple frame structure problems. It makes many simplifying assumptions about the material (linear-elastic, small deformation, homogeneous, isotropic, etc), and only allows for prismatic, straight beams. That being said, many structures can be reasonably approximated by those assumptions. In practice, it is often more time consuming to properly set up the inputs (i.e. meshes, section properties, fixities) than it is to perform the analyses.

With some additional work it could be modified to include:

- arbitrary distributed load integration
- response spectrum analysis
- settlements and earthquake ground motions

The code for this project is available on GitHub

6 Verification values for Swingset Problem

Nodal Displacements:

$$\begin{pmatrix} 0. & 0. & 0. & 0.000757374 & -2.5418\times10^{-6} & 0.0013384 \\ 3.3338\times10^{-17} & 0.00262786 & -0.0122854 & -0.00258862 & -6.93714\times10^{-20} & 8.70358\times10^{-19} \\ 0. & 0. & 0. & 0.000757374 & 2.5418\times10^{-6} & -0.0013384 \\ -5.7195\times10^{-16} & 7.09964\times10^{-13} & -4.46491 & 6.937\times10^{-17} & -3.56279\times10^{-20} & -1.47681\times10^{-19} \\ 0. & 0. & 0. & -0.000757374 & -2.5418\times10^{-6} & -0.0013384 \\ -2.58158\times10^{-18} & -0.00262786 & -0.0122854 & 0.00258862 & -1.02999\times10^{-20} & -4.27897\times10^{-19} \\ 0. & 0. & 0. & -0.000757374 & 2.5418\times10^{-6} & 0.0013384 \end{pmatrix}$$

Nodal Forces:

$$\begin{pmatrix} -0.44981 & 0.250522 & 1.125 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. \\ 0.44981 & 0.250522 & 1.125 & 0. & 0. & 0. \\ 0. & 0. & -4.5 & 0. & 0. & 0. \\ -0.44981 & -0.250522 & 1.125 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. \\ 0.44981 & -0.250522 & 1.125 & 0. & 0. & 0. \end{pmatrix}$$

element forces:

$$\begin{pmatrix} 1.21159 & 1.21159 & 0.501045 & 0.501045 & 1.21159 & 1.21159 \\ 0.250522 & -0.250522 & -5.66989\times10^{-17} & 1.23366\times10^{-16} & -0.250522 & 0.250522 \\ -0.000176699 & 0.000176699 & 2.25 & -2.25 & -0.000176699 & 0.000176699 \\ -9.27373\times10^{-15} & 2.27374\times10^{-13} & -7.08613\times10^{-15} & -5.31888\times10^{-15} & 3.22676\times10^{-14} & -5.68434\times10^{-14} \\ 4.61212\times10^{-18} & -0.475776 & -1252.61 & 2122.39 & -4.05707\times10^{-17} & -0.475776 \\ -3.66145\times10^{-15} & -674.552 & 1.28506\times10^{-13} & 1.39601\times10^{-13} & 8.65896\times10^{-14} & 674.552 \\ -1.21159 & -1.21159 & -0.501045 & -0.501045 & -1.21159 & -1.21159 \\ -0.250522 & 0.250522 & 5.66989\times10^{-17} & -1.23366\times10^{-16} & 0.250522 & -0.250522 \\ 0.000176699 & -0.000176699 & -2.25 & 2.25 & 0.000176699 & -0.000176699 \\ 9.27373\times10^{-15} & -2.27374\times10^{-13} & 7.08613\times10^{-15} & 5.31888\times10^{-15} & -3.22676\times10^{-14} & 5.68434\times10^{-14} \\ 0.475776 & 3.23075\times10^{-14} & -2122.39 & 1252.61 & 0.475776 & 0. \\ 674.552 & 2.84217\times10^{-14} & -2.13555\times10^{-13} & 4.54481\times10^{-14} & -674.552 & 5.68434\times10^{-14} \end{pmatrix}$$