2D Frame Solver — Static and Modal Analysis

1. Overview

This tool performs complete 2D frame structural analysis, including static displacement/force recovery, internal member forces (N, V, M), bending stresses, and global modal analysis. The interface allows the user to build a planar frame by drawing members, assigning material and section properties, applying nodal loads, springs, and supports, and evaluating the resulting behavior under both static loading and free vibration.

2. How to Use the Tool

2.1 Building the Model

• Draw members by selecting 'Draw Members' and dragging between points.
• Move, edit, or delete nodes and members via the 'Select/Edit' mode.
• Boundary conditions (fixed, simple, roller-X, roller-Y) are assigned at nodes.
• Nodal loads (Fx, Fy, Mz) and springs (Kx, Ky, Kθ) can be added through the node editor.
• Member properties include E, A, I, S, and weight density wd for self-weight.

2.2 Solving and Viewing Results

• Press 'Solve' to run the static analysis.
• Use 'Deflections' to visualize the displaced shape (with deflection scale).
• Use 'Stresses' to view member-by-member plots of σ_top, σ_bottom, N, V, and M.
• The Results panel lists nodal displacements and member stress summaries.

2.3 Modal Solver

• Press 'Modes' to compute global frame vibration modes.
• The tool extracts and sorts eigenpairs by ascending natural frequency.
• Only the lowest three physical modes are retained.
• The mode selector switches between static shape and modal shapes.
• The frequency scaling slider adjusts animation speed for visualization.

3. Static Solution Methodology

3.1 Degrees of Freedom

Each node has 3 DOFs: horizontal displacement (ux), vertical displacement (uy), and rotation (θz). Supports impose constraints by removing DOFs from the free set.

3.2 Element Formulation

A standard 2D Euler-Bernoulli frame element is used.
Local DOF order: [u_i, v_i, θ_i, u_j, v_j, θ_j]
The local stiffness matrix k_loc(E, A, I, L) follows the classical formulation.

3.3 Transformation to Global Coordinates

Element stiffness is transformed by $T^T k_{loc} T$ using the direction cosines (c, s) between the element and global x-axis.

3.4 Loads

• External nodal loads are assembled directly into the global load vector.
• Self-weight uses wd × A as the distributed load per unit length.
• The load is decomposed into local axial and transverse components, generating equivalent nodal forces and end moments.

3.5 Solving for Displacements

The global system $K_{ff} U_f = F_f$ is solved using numeric.js linear solver.
Constrained DOFs are removed, free DOFs solved, and full U is reconstructed.

3.6 Internal Forces and Stresses

Internal end forces are computed using $f_{loc} = k_{loc} * u_{loc}$ minus fixed-end forces from distributed loads. Forces are linearly interpolated along the member to compute N(x), V(x), M(x), and stresses σ_{top} and σ_{bottom} using:

$$\sigma = N/A \pm M/S$$

4. Modal Solution Methodology

4.1 Mass Matrix

A lumped (diagonal) mass matrix is assembled by distributing half of each member's self-weight mass to its end nodes. Nodal weight Wn is added directly as mass. Rotational inertia uses a small fraction (≈5%) of nodal mass.

4.2 Generalized Eigenproblem

The free-vibration problem solves:

$$K_{ff}\, \varphi = \omega^2 M_{ff}\, \varphi$$

A Cholesky factorization $M_{ff} = L L^T$ is used to convert it to a standard eigenproblem:

$$(L^{-1} K_{ff} L^{-T})\, y = \omega^2 y\,, \quad \varphi = L^{-T} y$$

Eigenpairs are computed using numeric.eig.

4.3 Mode Selection and Normalization

Eigenvalues are converted to natural frequencies (Hz). Modes are sorted, the lowest three retained, and eigenvectors normalized so their maximum DOF magnitude is 1. Modal shapes use the same subdivision routine as static deflections.

5. References

• Gere, J. M. & Timoshenko, S. P., *Mechanics of Materials*.
• Hibbeler, R. C., *Structural Analysis*.
• Cook, Malkus, Plesha, *Concepts and Applications of Finite Element Analysis*.
• Weaver & Gere, *Matrix Analysis of Framed Structures*.

• Clough & Penzien, *Dynamics of Structures*.
• numeric.js library for matrix operations and eigenvalue extraction.