Static Beam Solver

User & Method Documentation

Generated: January 26, 2026

1. Overview

This document describes the web-based Beam Solver module implemented in a single HTML file using JavaScript, Canvas, and Chart.js. The current implementation solves a linear, small-deflection Euler–Bernoulli beam problem discretized into up to 10 beam segments (elements) with node-based boundary conditions, point loads, end moments, and optional translational/rotational springs. Distributed loads may vary linearly along each segment. Results include deflection, bending moment, shear, and bending stress, displayed as plots and exportable as CSV/JSON.

This module performs a static analysis (equilibrium solution of K·u = f).

2. User Interface

2.1 Top Controls

Segments (max 10): Sets the number of beam segments (finite elements). The tool creates N segments and N+1 nodes.
Build Beam: Initializes (or resizes) the model data structures, seeds default properties, and redraws the canvas.
Solve: Assembles the global stiffness matrix and load vector, applies constraints, solves for DOFs, and updates plots.
Export JSON / Export CSV: Exports sampled results (deflection, shear, moment, stresses) at 21 points per segment.
EI scale: Visual-only scaling factor used to adjust the drawn beam height proportional to E·I; it does not change results.

2.2 Canvas Interaction

The beam is displayed in an HTML5 canvas. The diagram is interactive:
• Click near a node (blue dot) to open the Node Properties dialog.
• Click on a segment body to open the Segment Properties dialog.
The click logic maps screen pixels to canvas coordinates to remain robust under responsive resizing.

2.3 Node Properties Dialog

- Each node supports the following inputs:

  - BC (Boundary Condition): free, pinned (w=0), fixed (w=0 and θ=0).
  - F (Pos=Down): point transverse force applied at the node vertical DOF.
  - M: point bending moment applied at the node rotational DOF.
  - Kv: translational spring stiffness at the node vertical DOF.
  - Km: rotational spring stiffness at the node rotational DOF (suppressed for hinges).
  - Pin Joint: enables a hinge (independent left/right element rotations at that node).
  - Set w: prescribes w at the node (additional constraint).
  - Set θ: prescribes θ at the node (additional constraint).

2.4 Segment Properties Dialog

- Each segment supports the following inputs:

  - E: elastic modulus.
  - I: second moment of area about the bending axis.
  - A: cross-sectional area (used for self-weight via wd·A).
  - L: segment length.
  - St / Sb: top and bottom section moduli for bending stress.
  - qL / qR (Pos=Down): distributed load values at left/right ends of the segment; interpolated linearly.
  - wd: weight density; converted to an equivalent line load qw = −wd·A in the solver sign convention.
  - Material presets: Steel/Aluminum/Wood buttons populate E and wd (per unit system setting).
  - Section calculator (Rect): computes A, I, and St/Sb for a rectangular section and writes the values back into the segment.

2.5 Output Plots

After Solve, four plots are produced using Chart.js:
1) Deflection w(x)
2) Bending Moment M(x)
3) Shear V(x)
4) Bending Stress σ(x) at top/bottom fibers (using St/Sb).

3. Model Inputs

3.1 Segment Fields

Field	Meaning	Notes

E	Elastic modulus	Units consistent with loads and geometry.
I	Second moment of area	Units: length^4.
A	Cross-sectional area	Used to compute self-weight line load wd·A.
L	Segment length	Total beam length = ΣL.
St	Top section modulus	σ_top = −M/St.
Sb	Bottom section modulus	σ_bottom = +M/Sb.
qL, qR	Distributed load endpoints	Linearly varying q(x); UI indicates positive downward.
wd	Weight density	Self-weight modeled as qw = −wd·A (sign convention in solver).

3.2 Node Fields

Field	Meaning	Notes
BC	Boundary condition	free, pinned, fixed.
F	Point transverse force	Applied to w DOF; UI: positive = down.
M	Point moment	Applied to θ DOF; split across hinge rotations if hinge.
Kv	Translational spring	Adds stiffness to K[w,w].
Km	Rotational spring	Adds stiffness to K[θ,θ] for non-hinge node.
Pin Joint	Hinge behavior	Independent left/right rotations at node.
Set w	Prescribed displacement	Constrains w to specified value.
Set θ	Prescribed rotation	Constrains θ to specified value (both sides if hinge).

4. Solution Method (Static Finite Element)

4.1 Beam Element and Shape Functions

Each segment is modeled as an Euler–Bernoulli beam element with 4 DOFs: [w1, θ1, w2, θ2]. The implementation uses the standard Hermite cubic interpolation for transverse displacement w(x), consistent with Euler–Bernoulli bending.

4.2 Element Stiffness Matrix

For each segment with length L and flexural rigidity EI, the element stiffness matrix is:

k_e = (EI/L^3) ·
[[12, 6L, -12, 6L],
 [6L, 4L^2, -6L, 2L^2],

[-12, -6L, 12, -6L],
[6L, 2L^2, -6L, 4L^2]]

4.3 Hinges (Pin Joints) and DOF Map

A DOF mapping routine assigns one vertical displacement DOF w per node. For rotations:
• Non-hinge node: θ is shared by adjacent elements (C1 continuity).
• Hinge node: left and right elements each receive their own rotation DOF, allowing a slope discontinuity while preserving displacement continuity.

4.4 Distributed Loads and Equivalent Nodal Forces

Distributed load q(x) varies linearly from qL to qR along each segment. Self-weight is included as qw = −wd·A and added to qL and qR. The equivalent nodal force vector f_e is computed by numerical integration:
f_e = ∫_0^L N(x)^T q(x) dx

The integral is evaluated using 3-point Gauss quadrature over the element, which is adequate for the linear q(x) and the polynomial shape functions used.

4.5 Global Assembly and Constraints

The global stiffness matrix K and load vector F are assembled by summing element contributions. Node-level terms are then added:
• Point forces and moments.
• Translational and rotational springs (diagonal stiffness additions).
Boundary conditions and prescribed values are enforced by partitioning DOFs into constrained and free sets, forming the reduced system:
K_ff u_f = F_f − K_fs u_s

The reduced system is solved with numeric.js. If the reduced matrix is singular, the configuration is unstable (e.g., free-free with no springs).

4.6 Force Recovery, Diagrams, and Stress

Element end forces are recovered using:
f_int = k_e u_e − f_eq
where u_e are the element DOFs and f_eq is the equivalent nodal load vector. The tool reports end shear/moment values and evaluates continuous shear and moment along each segment using closed-form expressions consistent with linearly varying distributed load. Bending stress uses σ = ±M/S, with St and Sb for top and bottom fibers.

5. Export Formats

CSV export writes: x, w, V, M, SigTop, SigBot. JSON export writes a metadata header and a results array. Both exports sample 21 points per segment (including segment endpoints).

6. Practical Notes and Limitations

- This version is a static solver (no transient dynamics).
- The solve uses dense linear algebra (appropriate for ≤10 segments).
- Euler–Bernoulli theory neglects shear deformation and rotary inertia.
- Units are user-managed; the tool assumes consistent units across inputs.

To model a true dropped weight (impact) you would typically add: a mass matrix, a damping model, a time integrator (e.g., Newmark-β, generalized-α, or RKF45), and a contact/impact force law at the impact node.

7. References

- Bathe, K.-J., Finite Element Procedures, Prentice Hall.
- Cook, R. D., Malkus, D. S., Plesha, M. E., and Witt, R. J., Concepts and Applications of Finite Element Analysis, Wiley.
- Gere, J. M., and Goodno, B. J., Mechanics of Materials, Cengage Learning (beam bending relations; section modulus).
- Timoshenko, S. P., and Gere, J. M., Strength of Materials / Theory of Elasticity (classical beam theory background).
- Abramowitz & Stegun / standard numerical methods texts for Gauss–Legendre quadrature.
- Chart.js documentation (plotting).
- SweetAlert2 documentation (modal UI).
- numeric.js documentation (linear algebra solver).