Documentación Técnica: Graficadora de Vigas

Integrantes:
Palacios Rivera Joel Moisés
Malvaceda Roca Aldahir Arturo
Arosemena Elescano, Alejandro Miguel
Docente: M.Sc. Ing. Edén Angel Capcha Molina

Universidad: Nacional de Ingeniería (UNI)

Facultad: Facultad de Ingeniería Mecánica
Curso: Resistencia de Materiales - Ciclo 2025-II

6 de diciembre de 2025

Resumen

El presente documento ofrece una descripción exhaustiva del diseño, la arquitectura frontend y la implementación numérica back-end de la aplicación web "Viga Isostática Web". El objetivo es proporcionar una herramienta interactiva, con un diseño moderno (Glassmorphism), para calcular las reacciones y generar los diagramas de esfuerzos internos (axial, cortante, momento flector) en vigas isostáticas. Se detalla la formulación matricial del problema de equilibrio y la solución numérica mediante el método de eliminación de Gauss-Jordan.

1 Introducción

La aplicación resuelve el problema fundamental de la Estática: determinar las fuerzas internas y externas en un elemento estructural unidimensional sometido a cargas. El desarrollo se centra en dos pilares: precisión algorítmica y usabilidad mediante un diseño atractivo.

1.1 Definición y Grado de Indeterminación

Una viga es **isostática** si el número de incógnitas de reacción (r) es igual al número de ecuaciones de equilibrio (e). En un sistema 2D, se cumple:

$$I = r - e = 0 \quad \text{donde} \quad e = 3 + c$$

Siendo c el número de articulaciones internas. La clase `Viga` implementa una verificación estricta de esta condición en `resolverReacciones()`. Si $I \neq 0$, se notifica al usuario que la estructura es hipostática ($I < 0$) o hiperestática ($I > 0$) y el cálculo se detiene.

2 Arquitectura y Diseño Frontend

La aplicación sigue un diseño monolítico en un único archivo HTML, donde se integra la estructura (`HTML`), el estilo (`CSS`) y la lógica (`JavaScript`), siguiendo las mejores prácticas para aplicaciones ligeras y auto-contenidas.

2.1 Estructura General de la Página

La estructura se basa en una arquitectura de una sola página (SPA simple) con secciones modulares para la interacción y la visualización.

Cuadro 1: Componentes Principales del HTML

Sección / Elemento	Contenido
`<head>`	Metadatos, CSS interno, y la inclusión de la librería **Plotly.js**.
`<body>`	Contiene el `<header>`, el `.container` de la aplicación, y el `<footer>`.
`<header>`	Logo, título GIF y enlaces a integrantes. Utiliza el efecto **Glassmorphism**.
`.container`	Contiene los formularios de entrada (*Geometría, Apoyos, Cargas*) y el botón `CALCULAR`.
`.resultados`	Sección inicialmente oculta para mostrar diagramas y tabla de reacciones.

2.2 Diseño Visual: Estilo Glassmorphism y Animación

El diseño de la interfaz de usuario (UI) se basa en el principio de **Glassmorphism**, aprovechando los efectos de desenfoque y transparencia para crear un entorno visual moderno.

1. **Fondo Animado Dinámico:** El fondo utiliza la animación `@keyframes colorChange` con un ciclo de 10 segundos, variando sutilmente la tonalidad del azul oscuro (de `#2b3592` a `#4c5dbe`). Esto proporciona una sensación de dinamismo y reduce la fatiga visual.

2. **Contenedores de Vidrio (Glassmorphism):** Los elementos principales (`.container` y `.section`) y el encabezado (`header`) emplean `rgba` para la transparencia y `backdrop-filter: blur(8px)` en CSS para el efecto de vidrio esmerilado. Esto es crucial para la estética de la aplicación.

3. **Layout y Estilos:** El layout utiliza `display: flex` en `.flex-row-container` y `<header>` para organizar las secciones de entrada de datos de forma responsiva.

4. **Interacción de Nombres:** Los enlaces de los integrantes en el encabezado (`.nombres a`) presentan una animación sutil con `transform: translateY(-2px)` en el evento `:hover`, indicando claramente la interactividad y mejorando la respuesta de la interfaz.

2.3 Generación Dinámica y Entrada de Datos

La función `crearTablas()`, ejecutada al cargar la página, es el motor para construir la interfaz de entrada de datos, ofreciendo modularidad y consistencia en los inputs.

- **Apoyos:** 3 filas para tipos (*empotrado, articulado, rodillo*) y su posición.
- **Cargas Puntuales:** 6 filas para componentes **Fy** (vertical) o **Fx** (axial).
- **Cargas Distribuidas:** 3 filas para tipos (*Rectangular, Triangular/Trapezoidal, Parabólica*).

2.4 Visualización Gráfica mediante Plotly.js y HTML Canvas

Para la salida, se emplean dos tecnologías de renderizado distintas:

1. **Plotly.js (Diagramas):** Utilizado para los diagramas de esfuerzos (N, V, M) por su capacidad para generar gráficos interactivos de alta calidad. Los gráficos utilizan el relleno `fill:'tonexty'` para colorear el área bajo la curva.
2. **HTML Canvas (Esquema):** Se emplea para dibujar el esquema geométrico de la viga. La función `dibujarEsquema(divId)` utiliza una escala (`esc = canvas.width / this.L`) para mapear las posiciones físicas (en metros) a coordenadas de píxeles, garantizando la fidelidad de la representación.

3 Lógica de Cálculo y Estática

La clase `Viga` modela la estructura y contiene toda la lógica de solución, como se detalla en la siguiente tabla de métodos clave.

Cuadro 2: Métodos Clave de la Clase `Viga`

Método	Función Principal
`constructor(L)`	Inicializa la longitud L y los arrays de cargas y apoyos.
`cargaDistParcial(d, x)`	Calcula la **Fuerza Resultante (**F**)** y el **Centroide (**C**)** de una carga distribuida hasta la posición
`resolverReacciones()`	Verifica la **Isostaticidad** ($I = R - E_e - E_c = 0$) y construye el sistema $A\mathbf{x} = \mathbf{b}$.
`gaussJordan(A, b)`	Resuelve el sistema de ecuaciones lineales $A\mathbf{x} = \mathbf{b}$ mediante eliminación.
`diagramas()`	Itera sobre 1200 puntos para calcular $N(x)$, $V(x)$, y $M(x)$.
`dibujarEsquema()`	Renderiza el esquema de la viga en un elemento `<canvas>`.

3.1 Sistema de Ecuaciones de la Estática

El método `resolverReacciones()` construye y resuelve el sistema $A\mathbf{x} = \mathbf{b}$ basado en las ecuaciones de equilibrio, donde **x** es el vector de reacciones.

1. **Equilibrio Horizontal:** $\sum F_x = 0$
2. **Equilibrio Vertical:** $\sum F_y = 0$
3. **Equilibrio Rotacional (en un apoyo de referencia):** $\sum M = 0$
4. **Ecuación de Condición (por articulación, en posición p_i):** $\sum M^{\text{izq}}_{\text{articulación } p_i} = 0$

3.2 Algoritmo de Eliminación Gauss-Jordan

La función `gaussJordan(A, b)` utiliza **pivotaje parcial** para garantizar la estabilidad numérica al intercambiar filas, minimizando así errores de redondeo al evitar la división por números cercanos a cero.

3.3 Cálculo de Fuerzas Distribuidas (`cargaDistParcial`)

La función calcula la fuerza resultante **F** y su centroide **C** para la porción de carga activa hasta una posición x.

Cuadro 3: Cálculo de Componentes Variables de Cargas

Tipo de Carga	**Fuerza Resultante (F_{var})**	**Posición del Centroide (C_{var})**
Rectangular (w)	$w \cdot l$	$x_{\text{ini}} + \frac{l}{2}$
Triangular (de w_{ini} a w_{fin})	$\frac{1}{2} \cdot h \cdot l$	$x_{\text{ini}} + \frac{2}{3} \cdot l$
Parabólica	$k \cdot \frac{l^3}{3}$	$x_{\text{ini}} + \frac{3}{4} \cdot l$

4 Convención Estructural de Signos y Diagramas

4.1 Convención de Signos

La aplicación utiliza la siguiente convención para la generación de los diagramas, alineada con la práctica estándar de ingeniería para el diseño:

- **Fuerza Axial (N):**
  - $N > 0$: **Tensión** (fuerza saliente de la sección).
  - $N < 0$: **Compresión** (fuerza entrante a la sección).
- **Fuerza Cortante (V):**
  - $V > 0$: La porción izquierda sube con respecto a la derecha (gira la sección en sentido horario).
  - $V < 0$: La porción izquierda baja con respecto a la derecha.
- **Momento Flector (M):**
  - $M > 0$: **Tracción en la fibra inferior** (Convención de ingenieros, curvatura ”carita feliz”).
  - $M < 0$: Tracción en la fibra superior (curvatura ”carita triste”).

Figura 1: Convención de signos utilizada para los diagramas de esfuerzos internos.

4.2 Algoritmo de Seccionamiento

La función `diagramas()` realiza un seccionamiento a lo largo de 1200 puntos, aplicando las relaciones de equilibrio para el cálculo de los esfuerzos internos.

1. **Fuerza Axial ($N(x)$):** $N(x) = \sum F_x$ (a la izquierda de x)
2. **Fuerza Cortante ($V(x)$):** $\sum R_y$ y C_y a la izquierda de x.
3. **Momento Flector ($M(x)$):** Suma de momentos generados por todas las fuerzas y momentos puros a la izquierda de la sección x.

$$M(x) = \sum_{p_r \leq x} [R_y \cdot (x - p_r) + M_r] - \sum_{p_c \leq x} [C_y \cdot (x - p_c)]$$

5 Proceso de Flujo de la Aplicación (`calcular()`)

La función `calcular()` actúa como el controlador principal que orquesta la secuencia de análisis.

1. **Paso 1: Recolección y Modelado (Inputs):** Recolección de datos del DOM y validación.
2. **Paso 2: Fase Crítica (Reacciones):** Llamada a `viga.resolverReacciones()`.
3. **Paso 3: Fase de Post-Proceso (Esfuerzos):** Llamada a `viga.diagramas()` para generar los vectores N, V, M.
4. **Paso 4: Renderización Final:** Se actualiza la sección `.resultados` con el esquema (Canvas) y los diagramas (Plotly.js).

6 Conclusión

La aplicación "Graficadora de Vigas" logra un equilibrio entre una interfaz de usuario estéticamente avanzada (Glassmorphism) y una base de ingeniería sólida. La implementación del método de Gauss-Jordan y el análisis de seccionamiento con alta resolución de puntos garantizan resultados precisos, consolidando esta herramienta como un valioso recurso educativo y de verificación en el campo de la Resistencia de Materiales.