G1962 - Programación

Grado en Ingeniería Civil
Práctica 6

Javier González Villa
(19 de diciembre de 2025)

Programación orientada a objetos y librerías

Aprender a trabajar y familiarizarse con el paradigma de orientación a objetos así como la utilización de algunas librerías fundamentales.

Equilibrio Mecánico Estático

Se pretende modelar un problema clásico de Equilibrio Mecánico Estático como el presentado en la figura 1, con el propósito de automatizar y facilitar la realización de los cálculos necesarios. Para ello, se pretende crear el módulo **EquilibrioCables.py**, el cual contendrá una clase principal llamada **SistemaCables**, la cual estará compuesta por dos objetos de la clase **Cable** la *distancia* D entre los anclajes y la *masa* m del objeto que soportan.

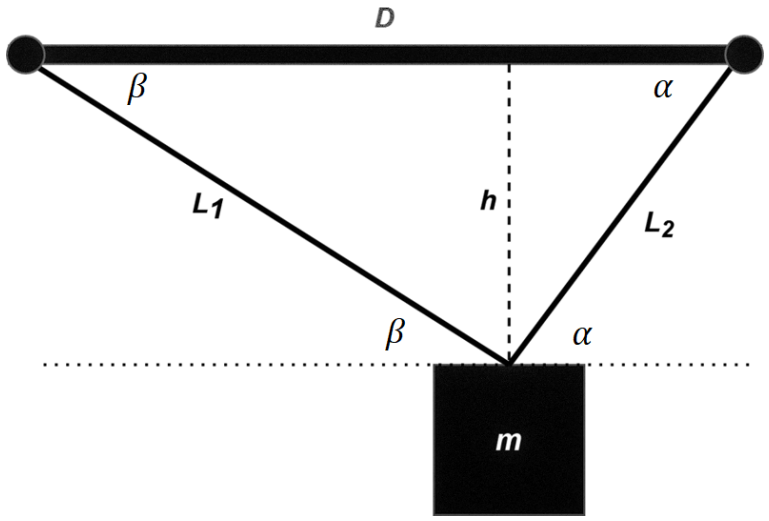

Figura 1: Esquema de equilibrio mecánico estático de cables.

Se desea poder conocer los siguientes datos del sistema:

- Ángulos α y β que forman los cables entre sí.
- Altura vertical h a la que quedara suspendido el objeto.
- Conocer si el sistema actual es estable o si se superarán las *cargas de rotura mínima*.
- En caso de rotura, la opción viable que remplace el cable roto de entre un catalogo de cables, permitiendo que el sistema sea seguro.

A continuación se proporcionan algunos de los teoremas y ecuaciones fundamentales relacionados con el problema a resolver:

- Teorema del coseno:

$$\begin{aligned} a^2 &= b^2 + c^2 - 2bc \cdot \cos A \\ b^2 &= a^2 + c^2 - 2ac \cdot \cos B \\ c^2 &= a^2 + b^2 - 2ab \cdot \cos C \end{aligned} \tag{1}$$

- Equilibrio estático de fuerzas:

$$\begin{aligned} \sum F &= 0 \\ \sum F_x &= T_{cd} \cdot \cos\alpha - T_{ci} \cdot \cos\beta \\ \sum F_y &= T_{cd} \cdot \sin\alpha + T_{ci} \cdot \sin\beta - F_p \end{aligned} \tag{2}$$

$$T_{cd} = \frac{T_{ci} \cdot \cos\beta}{\cos\alpha}, \qquad T_{ci} = \frac{\cos\alpha \cdot F_p}{(\cos\beta \cdot \sin\alpha) + (\sin\beta \cdot \cos\alpha)}$$

- Segunda Ley de Newton:

$$F = m \cdot g \tag{3}$$

Objetivos específicos:

Tras el planteamiento inicial del problema, se pide realizar una implementación en Python siguiendo los objetivos específicos:

- Implementar la clase **Cable** que debe describir los parámetros para la resolución del problema, siendo estos: el *nombre*, la *longitud* y la *carga de rotura mínima (CRM)*.
- Crear un repositorio o lista con los siguientes cables haciendo las veces de catalogo.

Cuadro 1: Catalogo de cables.

ID	Nombre	Longitud (m)	CRM (kg)
Cable0	1mm - 7x7	25	129
Cable1	3mm - 6x7+1	7	538
Cable2	4mm - 7x7	13	1030
Cable3	5mm - 7x19	16	1490
Cable4	8mm - 6x36+1	20	3800

- Implementar la clase **SistemaCables** que tiene que contener la definición del sistema a través de dos objetos de la clase *Cable*, la *masa* del objeto suspendido y la *distancia* entre anclajes.
- Implementar dentro de la clase **SistemaCables** los siguientes métodos con los cuales poder operar sobre el sistema:

- **Método constructor:** recibe los parámetros descritos y los almacena.
- **sistema_valido(self):** comprueba cuando el sistema planteado inicialmente tiene coherencia retornando si es o no válido mediante un booleano.
- **calcula_angulos(self):** método que calcula y retorna el par de ángulos α y β.
- **altura_vertical(self):** retorna la altura h a la que esta suspendida el objeto.
- **calcula_fuerzas(self):** retorna el par de valores de tensión de fuerzas de los cables T_{cd} y T_{ci}
- **rotura_de_cables(self):** retorna una lista de booleanos indicando si los cables debido al planteamiento del sistema romperán o no.
- **sustituye_cable_roto(self, catalogo):** cambia la configuración del sistema hasta encontrar en el catalogo un cable capaz de sustituir a alguno de los cables rotos y actualiza el sistema.

- Una vez implementado el código anterior, se pide probar los sistemas planteados en la tabla 2 y explicar los resultados obtenidos al probar los métodos del apartado anterior.

Cuadro 2: Sistemas de prueba.

Sistema	Cable Izq.	Cable Der.	Masa (kg)	Dist. Anclajes (m)
A	Cable0	Cable2	150	30
B	Cable0	Cable2	300	30
C	Cable0	Cable2	150	50
D	Cable0	Cable4	100	44
E	Cable1	Cable4	300	?

- Por ultimo para el **Sistema E** del cual se desconoce la distancia entre anclajes, se pide pintar en una gráfica, mediante el uso de la librería **matplotlib**, las funciones de altura y fuerzas para cada cable con distancia entre anclajes que oscile entre 0 y 50 m con precisión de 1 cm, calculando el punto óptimo donde el sistema ejerce el mismo ratio relativo de tensión sobre ambos cables.