

Software de cálculo para las prácticas informáticas de Estructuras I

Grupo A – Graduado en Edificación, Universidad de Granada, Curso 2022-23

ÍNDICE

- Introducción
- Instrucciones para el uso del código informático
- Comandos principales.



**UNIVERSIDAD
DE GRANADA**



escuela técnica superior
INGENIERÍA DE EDIFICACIÓN
Granada

Introducción



UNIVERSIDAD
DE GRANADA



escuela técnica superior

INGENIERÍA DE EDIFICACIÓN

Granada

Introducción a Python para Ingeniería y Ciencia

A lo largo de este curso se utilizará el lenguaje Python para la realización de las prácticas informáticas. El uso de Python está extendiéndose cada vez más en el mundo laboral en ingeniería (tanto en empresas como en investigación y desarrollo). Es muy importante desarrollar habilidades para implementar y programar los modelos matemáticos, especialmente en el contexto de la ingeniería, ya que la ingeniería necesita números, gráficas, programar tareas repetitivas, etc. Es por esto que las prácticas numéricas de esta asignatura se han planteado en este lenguaje. Indudablemente, el valor añadido que supone manejar un lenguaje como Python para un currículum es claro. Aunque no es un objetivo prioritario el aprendizaje sobre la programación propia en Python, se le recomienda que explore la documentación que existe online para adquirir un mayor grado de conocimiento sobre el mismo.

Python es un lenguaje muy sencillo, y es software libre. A lo largo del curso se presentarán aplicaciones para las cuales no se requieren conocimientos previos de Python, y sí conocimientos básicos sobre programación (variables, bucles, condicionales, funciones, etc). El objetivo será, por tanto, utilizar Python como herramienta para mejorar el aprendizaje propio de la asignatura.



Breve historia de Python

El lenguaje Python nace en el año 1980. La historia de Python comienza con el lenguaje ABC. Es un lenguaje de programación de propósito general, que fue desarrollado en Amsterdam (Holanda) en el CWI (Centrum Wiskunde & Informatica).

Guido van Rossum es su principal creador y desarrollador.

Sobre el nombre "Python", mucha gente piensa que se trata de una alusión a las serpientes. Pero el nombre está más ligado al grupo de humoristas británico Monty Python. Guido van Rossum, el creador de Python, escribió en 1996 sobre el origen del nombre de su lenguaje de programación: "Hace seis años, en diciembre de 1989, estaba buscando un hobby que me mantuviese ocupado durante la semana de Navidad. Mi oficina... estaba cerrada, pero tenía un ordenador en casa, y poco más. Decidí escribir un intérprete para el nuevo lenguaje de script que estaba pensando: una variante de ABC que sirviese a usuarios de Unix/C. Decidí usar Python como nombre de trabajo para el proyecto, a modo de broma irreverente (como fan del Monty Python's Flying Circus)".

Vea en el siguiente vídeo una entrevista a Guido van Rossum, por Charles Severance (Dr. Chuck, gran docente del mundo Python)



Breve historia de Python



https://www.youtube.com/watch?v=rTTFh7HOIC0&feature=emb_logo



UNIVERSIDAD
DE GRANADA

¿Por qué Python?

Existen numerosas razones para elegir Python como lenguaje de programación en ingeniería y ciencia.

- Aprender Python es muy sencillo.
- Python es un lenguaje potente.
- Tiene estructuras de datos de alto nivel, que permiten escribir código complejo en pocas líneas.
- La programación orientada a objetos es más sencilla que en otros lenguajes.
- Es extensible.
- Es *gratuito*.
- Tiene muchos paquetes orientados a cálculo numérico y simbólico.

En lo que sigue se resumen otras ventajas del lenguaje:

- Es fácil incorporar el intérprete de Python dentro de otras aplicaciones. (Salomé Meca, Code Aster, Revit, herramientas de ecosistemas BIM, etc).
- Python se puede extender mediante librerías (Numpy, Sympy, Matplotlib, etc).
- En aspectos numéricos y científicos, presenta ventajas en comparación con otros lenguajes interpretados. Python es un lenguaje para la ciencia. Existen numerosos recursos en la red sobre este aspecto (<https://github.com/>).



Tutorial básico de Python

Se puede acceder en el siguiente enlace:

https://colab.research.google.com/drive/12j09_6owZuHDg7RrzSgERcgxZIUaFHIp?usp=sharing

The screenshot shows a Google Colab notebook titled "Tutorial_básico_Python.ipynb". The notebook content includes:

- Tutorial básico de Python**
- Estructuras I. Grado en Edificación.**
- Enrique García-Macías**
- Departamento de Mecánica de Estructuras e Ingeniería Hidráulica**
- Universidad de Granada**
- escuela técnica superior
INGENIERÍA DE EDIFICACIÓN
Granada**
- ▼ Tipos de datos y variables**
- Variables**
- Como su nombre indica, una variable es algo que puede cambiar. Una variable es una forma de referirse a una localización en memoria utilizada por un programa de ordenador. Una variable tiene un nombre simbólico que "apunta" a una dirección de memoria física.
- En Python no se requiere una declaración de variables. Si se necesita una variable, se debe elegir un nombre y se le asignan directamente los valores. El tipo de variable que se asigna puede controlarse y cambiarse durante la ejecución del programa. Obsérvelo con el siguiente ejemplo, en el cual se usa la orden
- ```
type()
```
- Para indicar el tipo de dato.



# Otros tutoriales útiles

- <https://www.w3schools.com/python/default.asp>
- <https://stackabuse.com/python-tutorial-for-absolute-beginners/>

La documentación específica de los paquetes más usuales:

Numpy:

- <https://numpy.org/doc/stable/>

Matplotlib:

- <https://matplotlib.org/3.1.1/contents.html>

Scipy

- <https://docs.scipy.org/doc/scipy/reference/>

Pandas

- <https://pandas.pydata.org/docs/>



# Instrucciones para el uso del código informático

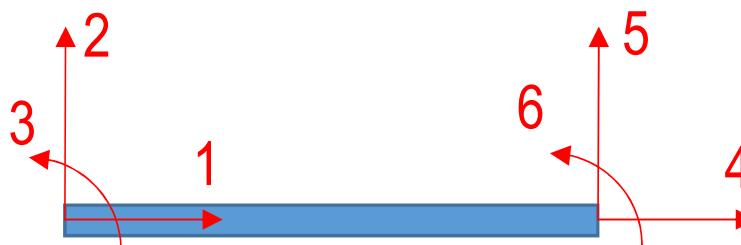


UNIVERSIDAD  
DE GRANADA



escuela técnica superior  
**INGENIERÍA DE EDIFICACIÓN**  
Granada

El código desarrollado (*anastruct*) se basa en el método de cálculo matricial de vigas, trabajando directamente sobre vigas planas de dos nodos con tres grados de libertad por nodo (desplazamiento horizontal, vertical, y giro). El código se encuentra alojado en un repositorio de “Github” ([https://github.com/EnriqueGarMac/Estructuras\\_I](https://github.com/EnriqueGarMac/Estructuras_I)), por lo que es posible trabajar completamente en **remoto** sin necesidad de instalar ningún programa.



README.md

# Calculador de Estructuras Planas

Código de cálculo matricial (direct stiffness) para el cálculo de pórticos planos y celosías. Código adaptado del paquete anaStruct (<https://github.com/ritchie46/anaStruct>) para las prácticas de la asignatura Estructuras I del grado de Edificación de la Universidad de Granada UGR

## Instalación

```
$!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git
```

UNIVERSIDAD DE GRANADA

10

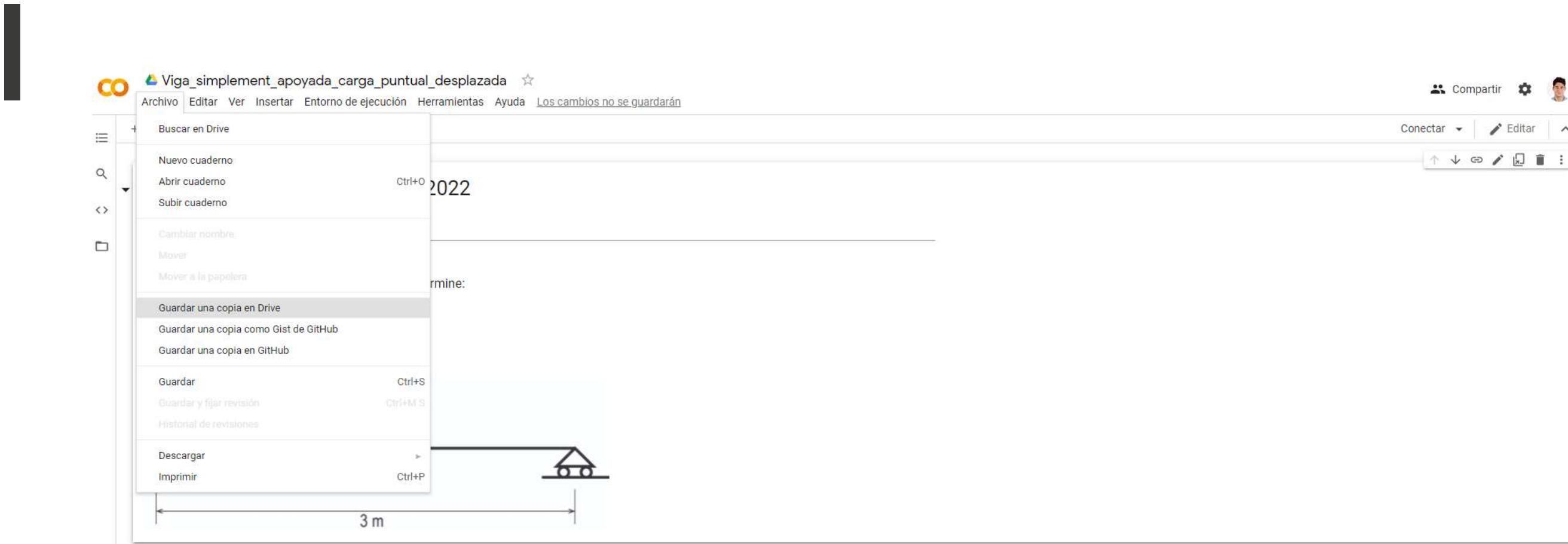


Los ejercicios básicos se proporcionan mediante un link que accede a un cuaderno de Jupyter (lenguaje Python). Por ejemplo: <https://colab.research.google.com/drive/1fnMTvEgO67geFcLD8SqjKPu-e9htuAMa?usp=sharing>  
Si vamos a ese enlace nos aparecerá algo así:

The screenshot shows a Google Colab notebook titled "Viga\_simplemente\_apoyada\_carga\_puntual\_desplazada". The notebook interface includes a toolbar with file operations like Archivo, Editar, Ver, Insertar, Entorno de ejecución, Herramientas, Ayuda, and a note about changes not being saved. Below the toolbar, there are buttons for Código, Texto, and Copiar en Drive. The main content area starts with a section titled "ESTRUCTURAS I - CURSO 2021-2022" and "Universidad de Granada". It then presents "EJERCICIO 1:" which describes a beam problem: "Dada la viga simplemente apoyada de la figura. Determine: 1. Reacciones en los apoyos 2. Ley de esfuerzos cortantes 3. Ley de esfuerzos flectores". A diagram of a beam is shown, supported by a pin at the left end and a roller at the right end. A downward-pointing arrow labeled "10 kN" is applied at a distance of "0.75 m" from the left support. The total length of the beam is indicated as "3 m". Below the diagram, there is explanatory text about connecting to a Google server and executing code using "anastruct".

Este es un cuadernillo de Jupyter con todos los códigos necesarios, únicamente hay que ir ejecutando línea a línea. Previamente, debemos guardar una copia del ejercicio en nuestro Google Drive (debemos usar nuestra cuenta go). Para ello, seleccionamos “Archivo/Guardar una copia en Drive”.





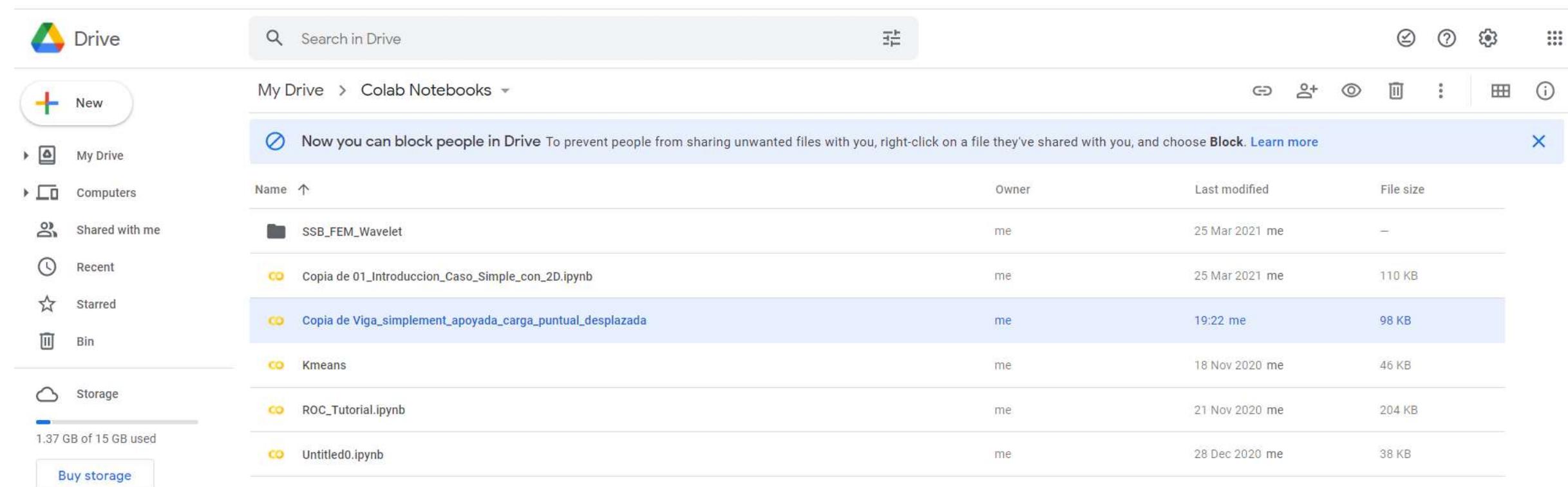
Lo primero que debemos hacer es conectarnos a un servidor de Google. Para ello, debemos clickar sobre la pestaña "Conectar" en la esquina superior derecha. Esto nos permitirá acceder a un ordenador que Google nos cederá de forma gratuita. De este modo, no será nuestro propio ordenador el que realice los cálculos, sino el ordenador al que se nos ha dado acceso. Una vez hayamos clickado en "Conectar", deberemos esperar unos segundos hasta que se nos habilite el acceso. Cuando esto ocurra, aparecerá un tick en verde, y se nos indicarán los valores de memoria RAM y de disco que se nos ha sido asignada.

A continuación, debemos cargar el paquete de código de "anastuct". Para ello, en la celda inferior tenemos el código necesario para acceder a la última versión del mismo. Cada celda se ejecuta clickando sobre el botón (▶) en la parte izquierda de la misma. Alternativamente, podemos teclear la combinación shift (↑) e intro.

Fijate que en las siguientes líneas de código, se definen muchos comentarios precedidos del símbolo (#). Python va a entender que el texto que sigue es un comentario, y por lo tanto no hará nada. Esto nos sirve para hacer anotaciones para posteriormente recordar los pasos



Si ahora vamos a nuestro Google Drive, veremos que hay una carpeta llamada “Colab Notebooks”, donde se irán almacenando todos los ejercicios que realicemos.

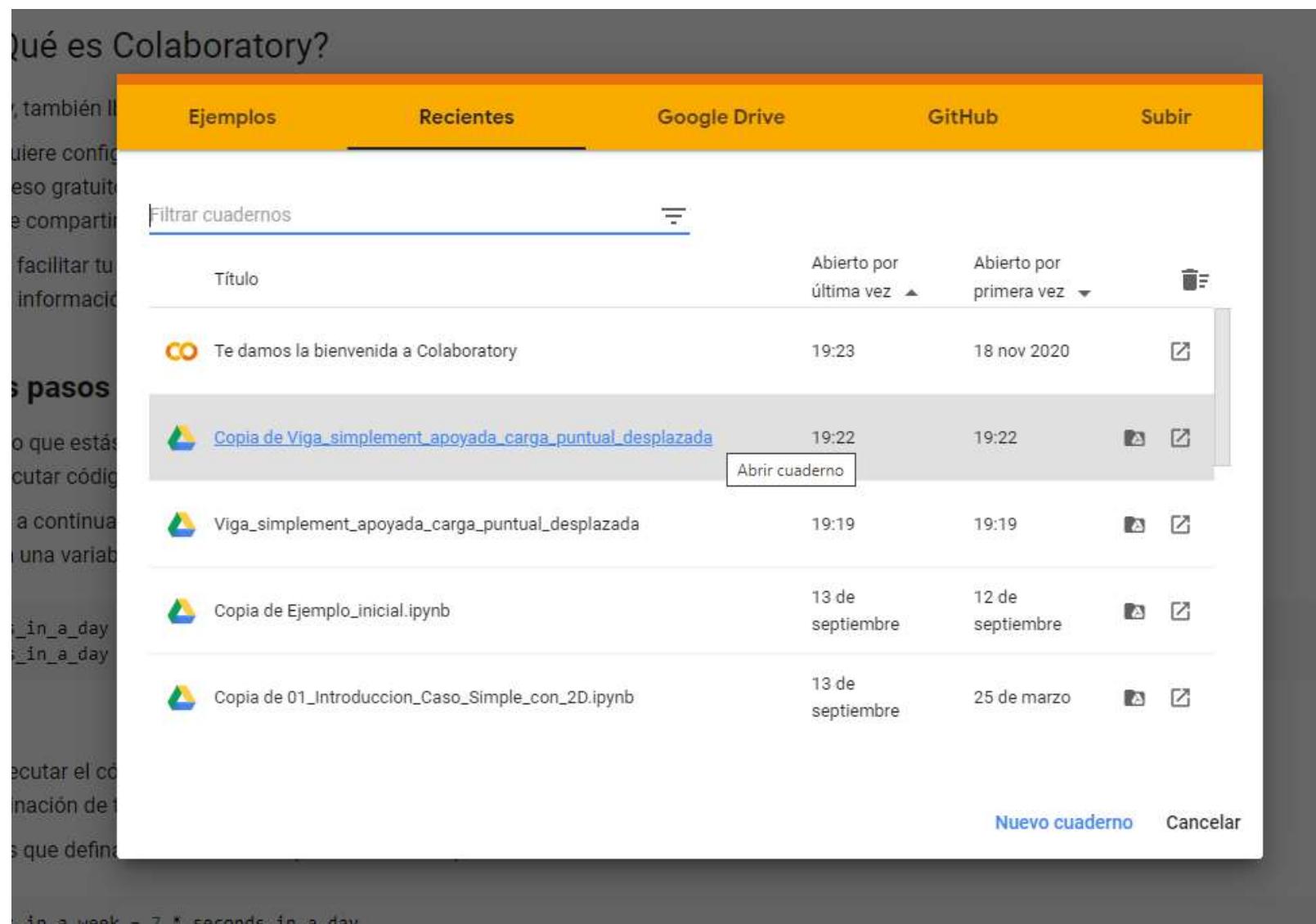


The screenshot shows the Google Drive interface. On the left, there's a sidebar with navigation links: 'My Drive', 'Computers', 'Shared with me', 'Recent', 'Starred', and 'Bin'. Below these are 'Storage' and a message about buying storage. The main area shows a folder named 'Colab Notebooks' under 'My Drive'. Inside this folder, there are several files listed in a table:

| Name                                                             | Owner | Last modified | File size |
|------------------------------------------------------------------|-------|---------------|-----------|
| SSB_FEM_Wavelet                                                  | me    | 25 Mar 2021   | -         |
| Copia de 01_Introduccion_Caso_Simple_con_2D.ipynb                | me    | 25 Mar 2021   | 110 KB    |
| <b>Copia de Viga_simplement_apoyada_carga_puntual_desplazada</b> | me    | 19:22         | 98 KB     |
| Kmeans                                                           | me    | 18 Nov 2020   | 46 KB     |
| ROC_Tutorial.ipynb                                               | me    | 21 Nov 2020   | 204 KB    |
| Untitled0.ipynb                                                  | me    | 28 Dec 2020   | 38 KB     |



Si ahora nos vamos a Google colab (<https://colab.research.google.com/>), podremos abrir cualquiera de los ejercicios que tengamos activos:





Todas las prácticas informáticas se realizarán en el marco de **Google colab**.

Colaboratory, o "Colab" para abreviar, es un producto de Google Research. Permite a cualquier usuario escribir y ejecutar código arbitrario de Python en el navegador. Es especialmente adecuado para tareas de aprendizaje automático, análisis de datos y educación. Desde un punto de vista más técnico, Colab es un servicio alojado de Jupyter Notebook que no requiere configuración y que ofrece **acceso gratuito a recursos informáticos**, como GPUs.



Trabajar en Google Colab es extremadamente sencillo. Probamos a abrir un cuaderno cualquiera. Por ejemplo: [https://colab.research.google.com/drive/12j09\\_6owZuHDg7RrzSgERcgxZIUaFHIp?usp=sharing](https://colab.research.google.com/drive/12j09_6owZuHDg7RrzSgERcgxZIUaFHIp?usp=sharing)

Inmediatamente se abrirá la siguiente pantalla.

The screenshot shows a Google Colab notebook titled "Tutorial\_basico\_Python.ipynb". The interface includes a toolbar with "Comentario", "Compartir", and "Editar" buttons. The notebook content is as follows:

Tutorial básico de Python  
Estructuras I. Grado en Edificación.  
• Enrique García-Macías  
Departamento de Mecánica de Estructuras e Ingeniería Hidráulica  
Universidad de Granada  
escuela técnica superior  
INGENIERÍA DE EDIFICACIÓN  
Granada

▼ Tipos de datos y variables

Variables

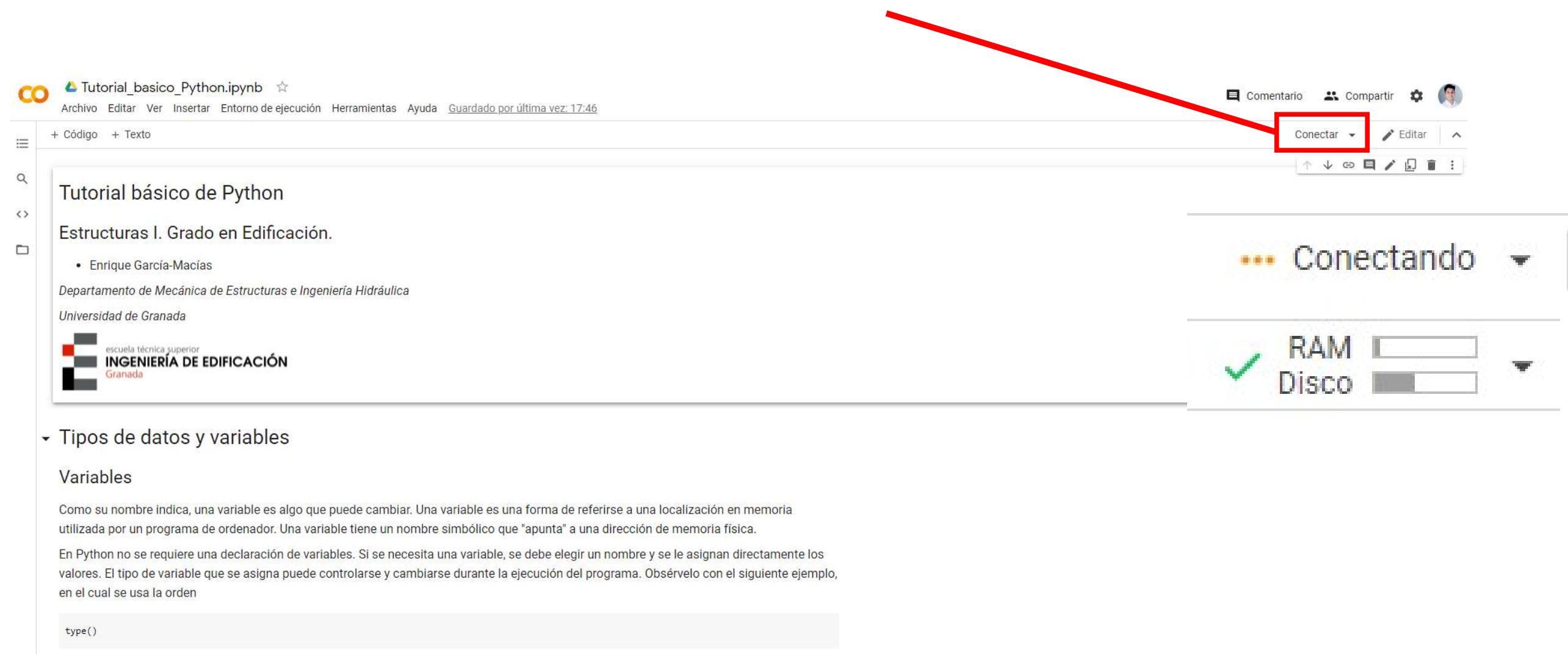
Como su nombre indica, una variable es algo que puede cambiar. Una variable es una forma de referirse a una localización en memoria utilizada por un programa de ordenador. Una variable tiene un nombre simbólico que "apunta" a una dirección de memoria física.

En Python no se requiere una declaración de variables. Si se necesita una variable, se debe elegir un nombre y se le asignan directamente los valores. El tipo de variable que se asigna puede controlarse y cambiarse durante la ejecución del programa. Obsérvelo con el siguiente ejemplo, en el cual se usa la orden

```
type()
```



Es esta pestaña podremos solicitar el acceso a una máquina en la que realizar los cálculos que deseemos. Esta máquina es gestionada por Google (normalmente un ordenador de bastante capacidad), por lo que nuestro ordenador no realizará ningún cálculo más allá de abrir el buscador web. Esto tiene enormes ventajas a la hora de lanzar cálculos pesados, ya que nuestro ordenador no necesita tener una gran capacidad de cálculo.



The screenshot shows a Jupyter Notebook interface with a red arrow pointing from the text above to the 'Connectar' button in the toolbar. The notebook title is 'Tutorial\_basico\_Python.ipynb'. The toolbar includes 'Comentario', 'Compartir', 'Editar', and a user profile icon. The main area displays a section titled 'Tutorial básico de Python' with sub-sections like 'Estructuras I. Grado en Edificación.' and 'Tipos de datos y variables'. A sidebar on the right shows 'Conectando' status with progress bars for 'RAM' and 'Disco'. The bottom code cell contains the command 'type()'.



En Google colab únicamente podremos ejecutar cuadernos de Jupyter. Estos cuadernos contiene códigos en Python organizados por celdas.

```
type()
```

Para indicar el tipo de dato.

A continuación, la siguiente celda es de cálculo. Para poder ejecutarla, simplemente haga click sobre la misma, y pulse a la vez los botones "Mayúscula derecha" (una flecha hacia arriba) y "Enter". Típicamente, este movimiento se realiza pulsando con el dedo índice la tecla de mayúscula derecha, y con el dedo corazón, la tecla enter.

```
i = 10
print ("El valor de i es", i)
print ("El tipo de i es", type(i))
```

Celda 1

```
[] i = 10.1
print ("El valor de i es", i)
print ("El tipo de i es", type(i))
```

Celda 2

El signo "=" no debe ser visto como "igual a", sino como "asignado a"

```
[] i = 10
i = i+1
print(i)
```

#### ▼ Números

Python incluye un conjunto de números (objetos) dentro de su librería standard. Los básicos son:



Cada celda se ejecuta, o bien pulsando el icono “play”, o empleando la combinación de teclas Control+Intro

En el ejemplo inicial se explican estos pasos con mayor detalle:

<https://colab.research.google.com/drive/1hP61dQwmcKchGaGCUdZUi-TsOhSQVjaL?usp=sharing>

```
✓ 0s i = 10
 print ("El valor de i es", i)
 print ("El tipo de i es", type(i))

↳ ('El valor de i es', 10)
 ('El tipo de i es', <type 'int'>

[2] i = 10.1
 print ("El valor de i es", i)
 print ("El tipo de i es", type(i))

('El valor de i es', 10.1)
('El tipo de i es', <type 'float'>)
```

El signo "=" no debe ser visto como "igual a", sino como "asignado a"

```
✓ 0s i = 10
 i = i+1
 print(i)

11
```

Una vez ejecutada la celda, aparecerá inmediatamente el resultado de la misma.

# Comandos principales



UNIVERSIDAD  
DE GRANADA



escuela técnica superior  
**INGENIERÍA DE EDIFICACIÓN**  
Granada

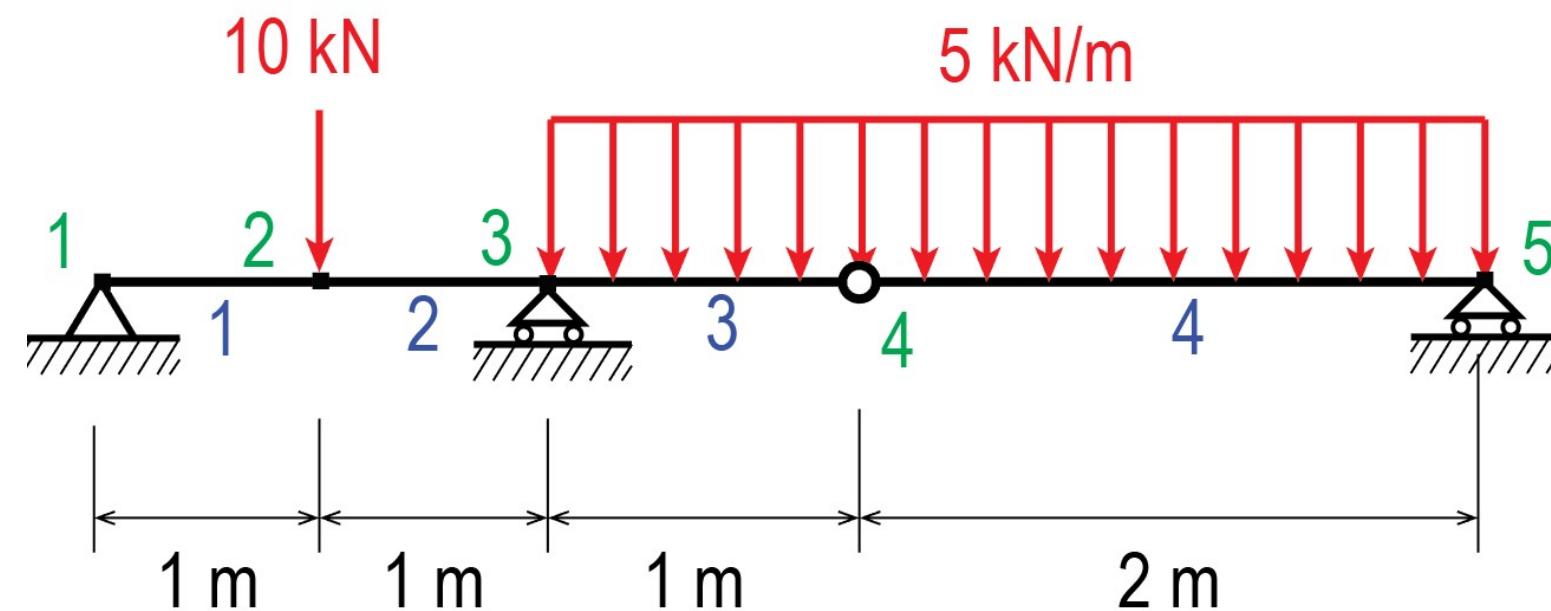
El software desarrollado, denominado anaStruct, se estructura como cualquier otro programa de elementos finitos. Los módulos principales se estructuran como sigue:

- 1. Generación de la geometría – Nudos y vigas.**
- 2. Definición de los materiales**
- 3. Definición de las condiciones de contorno.**
- 4. Definición de las condiciones de carga.**
- 5. Resolución**
- 6. Postproceso**



## 1. Definición de la geometría

Debemos definir nuestra estructura mediante nodos y vigas. Es recomendable organizar la numeración de los mismos previamente en un papel. Véase el siguiente ejemplo.



## 1. Definición de la geometría y materiales

El primer comando que debemos introducir es la importación del paquete de software anaStruct.

# Instalar paquete anastruct

```
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git
```

# Éste comando descargará la versión más reciente del software

# Cargamos el software anastruct

```
from anastruct import SystemElements
```

Definimos un objeto que representará nuestra estructura

# Cargamos el objeto de nuestra estructura

```
ss = SystemElements()
```



## 1. Definición de la geometría y materiales

Introducción de elementos tipo viga (rigidez axial y a flexión)

El comando básico se define como sigue

```
ss.add_element(location, EA=None, EI=None, g=0, mp=None, spring=None, **kwargs)[source]
```

Argumentos de entrada

- Location – Localización de los nodos de la viga (Ej. `location=[[0, 0], [1, 1]]`, para una viga que va de (0,0) a (1,1))
- EA – Módulo de rigidez axial (E: Módulo de Young, A: Área). *Para el cálculo de esfuerzos no será necesario.*
- EI – Módulo de rigidez a flexión (E: Módulo de Young, I: Inercia). *Para el cálculo de esfuerzos no será necesario.*
- `spring={nodo: k}` – Nos permite definir muelles en los nodos de los extremos. Se definen por pares nodo: k, con k siendo la rigidez del muelle. Si queremos definir una rótula, podemos indicar k=0.



## 1. Definición de la geometría y materiales

Introducción de elementos tipo tirante (únicamente rigidez a tensión/compresión)

El comando básico se define como sigue

```
ss.add_truss_element(location, EA=None, **kwargs)
```

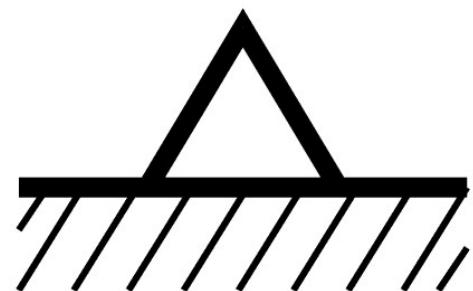
Argumentos de entrada

- Location – Localización de los nodos de la viga (Ej. `location=[[0, 0], [1, 1]]`, para un tirante que va de (0,0) a (1,1))
- EA – Módulo de rigidez axial (E: Módulo de Young, A: Área). *Para el cálculo de esfuerzos no será necesario.*



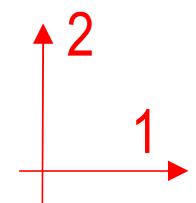
### 3. Definición de las condiciones de contorno

- Apoyo fijo

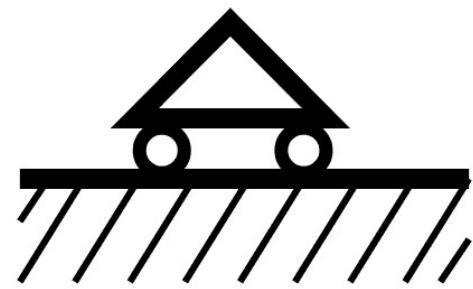


ss.add\_support\_hinged(node\_id=1)

ID del nodo



- Carrito



ss.add\_support\_roll(node\_id=3, direction=2)

ID del nodo

Dirección

- Empotramiento



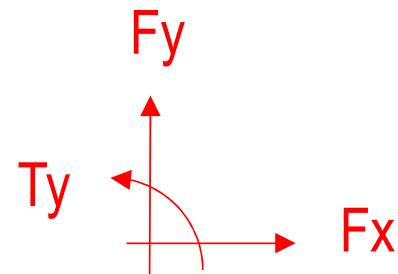
ss.add\_support\_fixed(node\_id=1)

ID del nodo

## 4. Definición de las condiciones de carga

- Cargas puntuales

ID del nodo  
↓  
`ss.point_load(2, Fx=0, Fy=-10)`

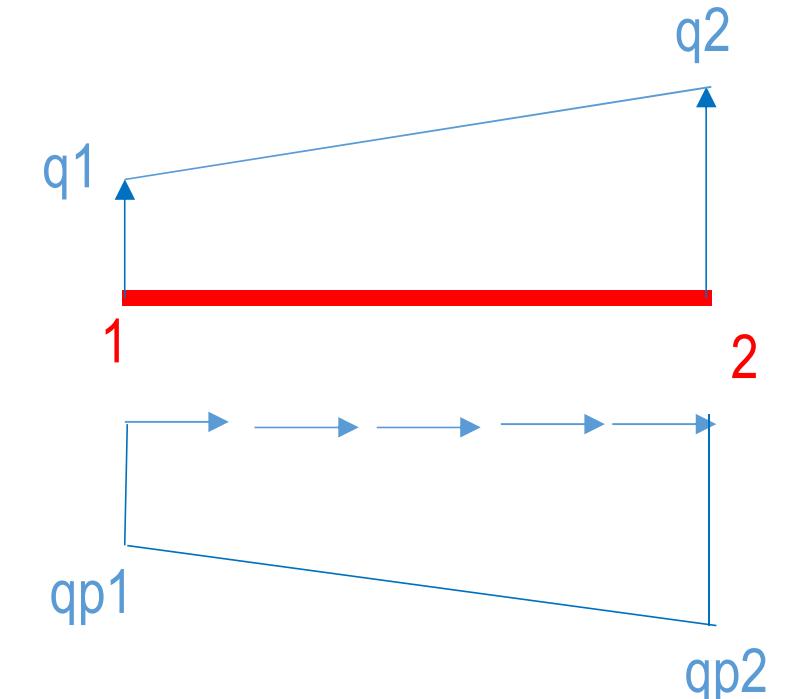


- Momentos puntuales

ID del nodo  
↓  
`ss.moment_load(2, Ty=10)`

- Cargas distribuidas

ID del elemento  
↓  
`ss.q_load(element_id=1, q=(q1,q2), q_perp=(qp1,qp2))`



## 5. Resolución

```
Resolvemos la estructura
ss.solve()
```

## 6. Postproceso

```
Mostramos las reacciones
ss.show_reaction_force()
```

```
Mostramos cortantes
#ss.show_shear_force()

Mostramos flectores
#ss.show_bending_moment()

Mostramos axiles
#ss.show_axial_force()
```

```
Mostramos deformada
ss.show_displacement()

Mostramos todos los resultados juntos
ss.plotter.results_plot()
```



Ejercicios - [https://github.com/EnriqueGarMac/Estructuras\\_I/tree/main/Ejemplos/Cuadernillos\\_Jupyter](https://github.com/EnriqueGarMac/Estructuras_I/tree/main/Ejemplos/Cuadernillos_Jupyter)

## Ejemplos de cálculo

A continuación se listan los ejemplos de práctica disponibles

- Tutorial básico de Python --- [!\[\]\(d27108a7e74860dfc560bbe498136a6f\_img.jpg\) Open in Colab](#)
- Ejemplo inicial: Viga simplemente apoyada con carga puntual\_desplazada --- [!\[\]\(ee590f550fd5e60389538d46d1a0187b\_img.jpg\) Open in Colab](#)
- Práctica I: Cálculo de esfuerzos en vigas --- [!\[\]\(77bd2f78a921d6e2320ca465db0b5aeb\_img.jpg\) Open in Colab](#)
- Viga simplemente apoyada con carga puntual centrada --- [!\[\]\(51d212cc7ca9d1dde65a43bdf8b65882\_img.jpg\) Open in Colab](#)
- Viga simplemente apoyada con carga puntual desplazada --- [!\[\]\(4005f954b4d07655bfd557cac2c2dc85\_img.jpg\) Open in Colab](#)
- Viga simplemente apoyada con momento puntual --- [!\[\]\(bc5d6d1deb5672428ea1b9ebc683873b\_img.jpg\) Open in Colab](#)
- Viga simplemente apoyada con fuerza distribuida --- [!\[\]\(ff9a33531f36aa0f7a00ff2858d57af0\_img.jpg\) Open in Colab](#)
- Voladizo con carga uniforme --- [!\[\]\(91fcc4842c503f0b385cc3eae70e47a3\_img.jpg\) Open in Colab](#)
- Voladizo con carga triangular --- [!\[\]\(ba4ba9b32c2f5d2b3a8b8066703e4d0c\_img.jpg\) Open in Colab](#)
- Voladizo con fuerza puntual --- [!\[\]\(161885ba55b9606824698754cec2e4e8\_img.jpg\) Open in Colab](#)
- Voladizo con fuerza puntual desplazada --- [!\[\]\(b9c4855c734177dc544f1fc67f13882b\_img.jpg\) Open in Colab](#)
- Voladizo con momento puntual --- [!\[\]\(93247fdfb09b6b46ae9210294b87533c\_img.jpg\) Open in Colab](#)
- Ejemplo pórtico con cargas puntuales --- [!\[\]\(1f48c46215cf87232a980db9d68451df\_img.jpg\) Open in Colab](#)
- Pórtico Ejercicio 6 relación --- [!\[\]\(90d0e9bd370f33eb11da5c0366724fa4\_img.jpg\) Open in Colab](#)
- Celosía Warren --- [!\[\]\(60cfabffc51f8d2e6794a7731a6e3704\_img.jpg\) Open in Colab](#)



# Tutorial básico de Python

## Estructuras I. Grado en Edificación.

- Enrique García-Macías

Departamento de Mecánica de Estructuras e Ingeniería Hidráulica

Universidad de Granada



## Tipos de datos y variables

### Variables

Como su nombre indica, una variable es algo que puede cambiar. Una variable es una forma de referirse a una localización en memoria utilizada por un programa de ordenador. Una variable tiene un nombre simbólico que "apunta" a una dirección de memoria física.

En Python no se requiere una declaración de variables. Si se necesita una variable, se debe elegir un nombre y se le asignan directamente los valores. El tipo de variable que se asigna puede controlarse y cambiarse durante la ejecución del programa. Obsérvelo con el siguiente ejemplo, en el cual se usa la orden

```
type()
```

Para indicar el tipo de dato.

A continuación, la siguiente celda es de cálculo. Para poder ejecutarla, simplemente haga click sobre la misma, y pulse a la vez los botones "Mayúscula derecha" (una flecha hacia arriba) y "Enter". Típicamente, este movimiento se realiza pulsando con el dedo índice la tecla de mayúscula derecha, y con el dedo corazón, la tecla enter.

```
In [1]: i = 10
print ("El valor de i es", i)
print ("El tipo de i es", type(i))

('El valor de i es', 10)
('El tipo de i es', <type 'int'>)
```

```
In [2]: i = 10.1
print ("El valor de i es", i)
print ("El tipo de i es", type(i))

('El valor de i es', 10.1)
('El tipo de i es', <type 'float'>)
```

El signo "=" no debe ser visto como "igual a", sino como "asignado a"

```
In [3]: i = 10
i = i+1
print(i)

11
```

### Números

Python incluye un conjunto de números (objetos) dentro de su librería standard. Los básicos son:

- Enteros.
- Punto flotante.
- Complejos

### Números enteros

Se definen de forma análoga a otros lenguajes: número sin punto decimal.

```
In []: a = 2
```

```
type (a)
```

## Números de punto flotante.

Si se introduce el punto decimal, y se añaden decimales, Python interpretará el número como punto flotante.

```
In []: b = 12.0
type (b)
```

## Números de punto flotante.

La variable compleja se escribe introduciendo la unidad imaginaria con la letra *j*

```
In []: c = 12 + 2j
type(c)
```

## Herramientas matemáticas en Python: librerías Numpy, SciPy, Matplotlib y SymPy

Python puede utilizarse como calculadora, para realizar operaciones sobre expresiones. Existen tutoriales de Python en los cuales pueden verse operaciones sobre la librería standard, la cual es muy limitada. Para extender Python al mundo del cálculo científico, se han desarrollado librerías específicas. Dentro de ellas, las más importantes son dos:

- La librería [Numpy](#)
- La librería [SciPy](#)

## La librería Numpy

Python está organizado en módulos, que son archivos con extensión `.py` que contienen funciones, variables y otros objetos, y paquetes, que son conjuntos de módulos. Cuando queremos utilizar objetos que están definidos en un módulo tenemos que *importarlo*, y una vez que lo hemos hecho podemos usar el operador `.` para ir descendiendo en la jerarquía de paquetes y acceder al objeto que necesitamos. Por ejemplo, de esta manera importamos NumPy:

```
In []: import numpy
```

Y de esta manera accedemos a la función `norm`, que calcula la norma (o módulo) de un vector (array):

```
In []: numpy.linalg.norm
```

La función `norm` está dentro del paquete `linalg`, que a su vez está dentro del paquete NumPy.

La convención para importar NumPy es introducir un nombre corto. Suele utilizarse las letras "np"

```
In []: import numpy as np
```

Lo que hacemos es crear un *alias* al paquete NumPy de nombre `np`. Es simplemente una forma de abreviar el código. Esta forma de separar las funciones en paquetes (que se llaman **espacios de nombres** o *namespaces*) conduce a una mayor legibilidad del código y a la supresión de ambigüedades.

## Constantes y funciones matemáticas

Además de arrays (vectores), NumPy contiene también constantes y funciones matemáticas de uso cotidiano.

```
In []: np.e
```

```
In []: np.pi
```

```
In []: np.log(2)
```

## ¿Qué es exactamente un array de Numpy?

Un array de NumPy es la forma en la que se representarán vectores. Particularmente los vectores de Numpy están planteados para realizar operaciones automáticas sobre los índices, de forma rápida y eficiente, evitando bucles. Su definición es sencilla.

Los arrays de NumPy son *homogéneos*, es decir, todos sus elementos son del mismo tipo. Si le pasamos a `np.array` una secuencia con objetos de tipos diferentes, promocionará todos al tipo con más información. Para acceder al tipo del array, podemos usar el atributo `dtype`.

Observe el siguiente ejemplo, en el que se genera un vector de números enteros.

```
In []: a = np.array([1, 2, 3])
print(a)
print(a.dtype)
```

```
In []: type(a)
```

Observe el siguiente ejemplo, con tipos de número diferentes:

```
In []: a = np.array([1, 2, 3.0])
print(a.dtype)
```

Los arrays permiten realizar operaciones. Observe en los siguientes ejemplos los productos escalar y vectorial entre los vectores

$\vec{a}$

y

$\vec{b}$

.

```
In []: a = np.array([1,2,3])
b = np.array([4,5,6])
p_escalar = a.dot(b)
p_vectorial = np.cross(a,b)

print ("El producto escalar es", p_escalar)
print ("El producto vectorial es", p_vectorial)
```

En Mecánica, utilizaremos con frecuencia el producto escalar y vectorial. ¡Vamos a practicar su uso con unos ejercicios!

### Ejercicios (uso del producto escalar y vectorial)

1. Considere los vectores

$\vec{a}$

y

$\vec{b}$

. Determine:

a) Producto vectorial. b) Ángulo formado por los mismos.

Solución: En primer lugar determinemos su producto vectorial.

```
In []: a = np.array([1,2,-1])
b = np.array([-2,1,3])
c = np.cross(a,b)
c
```

A continuación, mediante el producto escalar, y sabiendo que el producto escalar verifica:

*[Math Processing Error]*

Podemos despejar el ángulo. Para calcular el módulo, utilizaremos la función:

```
np.linalg.norm
```

Observe cómo funciona. A continuación se calculará el módulo de

$\vec{a}$

y se comparará con el valor calculado de forma analítica.

```
In []: modulo_a = np.linalg.norm(a)
print ("Modulo de a"),modulo_a
modulo_calculado = np.sqrt(a[0]**2 + a[1]**2 + a[2] **2) # Atención: Python indexa desde el 0
print (modulo_calculado)
```

A continuación se plantea el bloque de código completo que permite obtener el ángulo

$\theta$

.

```
In []: modulo_b = np.linalg.norm(b)
pescalar = a.dot(b)
coseno = pescalar / (modulo_a * modulo_b)
alfa = np.arccos(coseno)
print ("Ángulo, en radianes"), alfa
```

Para pasar este valor a grados sexagesimales, puede hacerse de forma directa o empleando una función de Numpy.

```
In []: np.degrees(alfa)
```

```
In []: alfa*180/np.pi # De forma directa
```

Aunque existan dos ángulos con el mismo coseno, esta forma de calcular el ángulo permite obtener el MENOR ángulo formado por ambos vectores.

```
In []:
```

Typesetting math: 100%

# ESTRUCTURAS I - CURSO 2021-2022

Universidad de Granada

## EJERCICIO 1:

Dada la viga simplemente apoyada de la figura. Determine:

1. Reacciones en los apoyos
2. Ley de esfuerzos cortantes
3. Ley de esfuerzos flectores



Lo primero que debemos hacer es conectarnos a un servidor de Google. Para ello, debemos clickar sobre la pestaña "Conectar" en la esquina superior derecha. Esto nos permitirá acceder a un ordenador que Google nos cederá de forma gratuita. De este modo, no será nuestro propio ordenador el que realice los cálculos, sino el ordenador al que se nos ha dado acceso. Una vez hayamos clickado en "Conectar", deberemos esperar unos segundos hasta que se nos habilite el acceso. Cuando esto ocurra, aparecerá un tick en verde, y se nos indicarán los valores de memoria RAM y de disco que se nos ha sido asignada.

A continuación, debemos cargar el paquete de código de "anastruct". Para ello, en la celda inferior tenemos el código necesario para acceder a la última versión del mismo. Cada celda se ejecuta clickando sobre el botón (►) en la parte izquierda de la misma.

Alternativamente, podemos teclear la combinación shift (↑) e intro.

Fíjate que en las siguientes líneas de código, se definen muchos comentarios precedidos del símbolo (#). Python va a entender que el texto que sigue es un comentario, y por lo tanto no hará nada. Esto nos sirve para hacer anotaciones para posteriormente recordar los pasos seguidos.

```
In [2]: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-ki92o6vh
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-ki92o6vh
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.4.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages
(from matplotlib>=3.0->anastruct==0.0.0)
(2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(1.3.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0)
(1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=58944 sha256=8168439fc3a82cde5900a01fad39cb3bc3f88b47ffe6d348655ef17a458dfb72
 Stored in directory: /tmp/pip-ephem-wheel-cache-7w8zd1zj/wheels/23/97/1a/d460d2d29cccd27f0842a8e862290c78834c55be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

```
In [3]: # Cargamos el software anastruct
from anastruct import SystemElements

Cargamos el objeto de nuestra estructura
```

```
ss = SystemElements()
```

```
In [4]: # Vamos a definir ahora algunas variables
L = 3.0 # Longitud de la barra [m]
P = -1000.0 # Carga puntual [N]
a = 0.75 # Distancia a la que se encuentra la carga aplicada [m]
```

```
In [5]: # Construcción de la estructura
```

```
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [a, 0]]);
Añadimos elemento barra 2
ss.add_element(location=[[a, 0], [L, 0]]);
```

```
In [6]: # Añadimos apoyo fijo al nodo 1
```

```
ss.add_support_hinged(node_id=1)
Añadimos carrito al nodo 3
ss.add_support_roll(node_id=3, direction=2)
```

```
In [7]: # Añadimos carga puntual al nodo 2
```

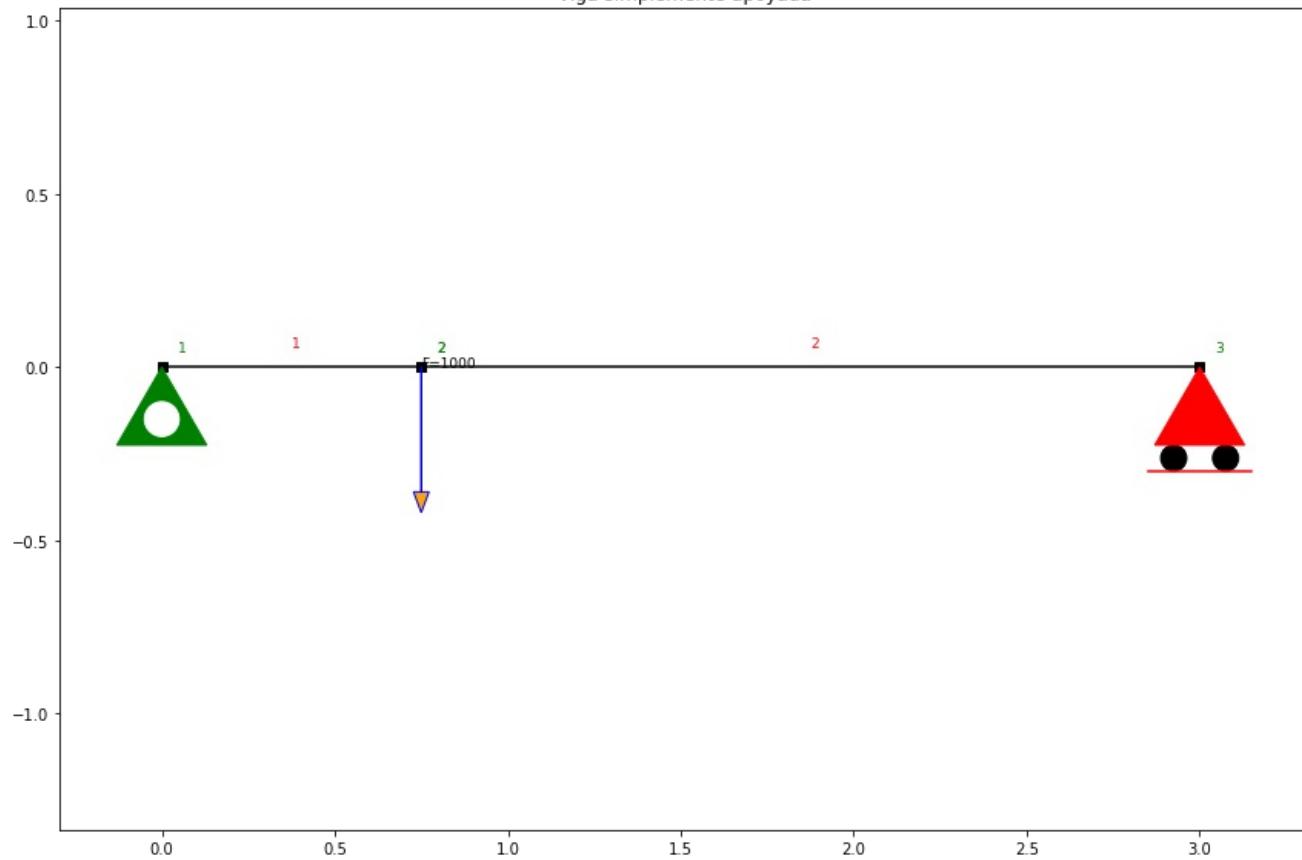
```
ss.point_load(2, Fx=0, Fy=P)
```

```
In [8]: # Mostramos estructura generada
```

```
ss.show_structure(title='Viga simplemente apoyada')
```

-0.0

Viga simplemente apoyada

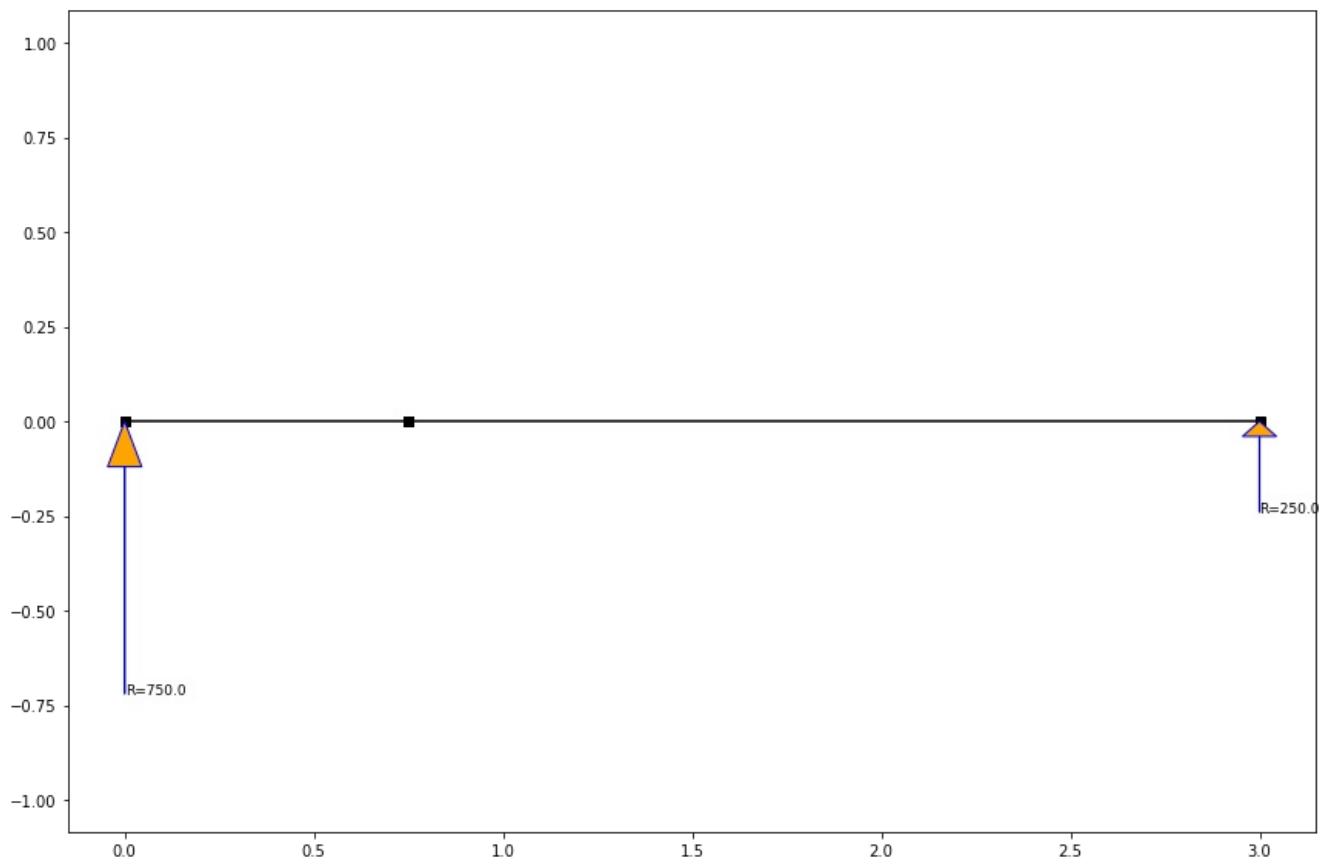


```
In []: # Resolvemos la estructura
ss.solve();
```

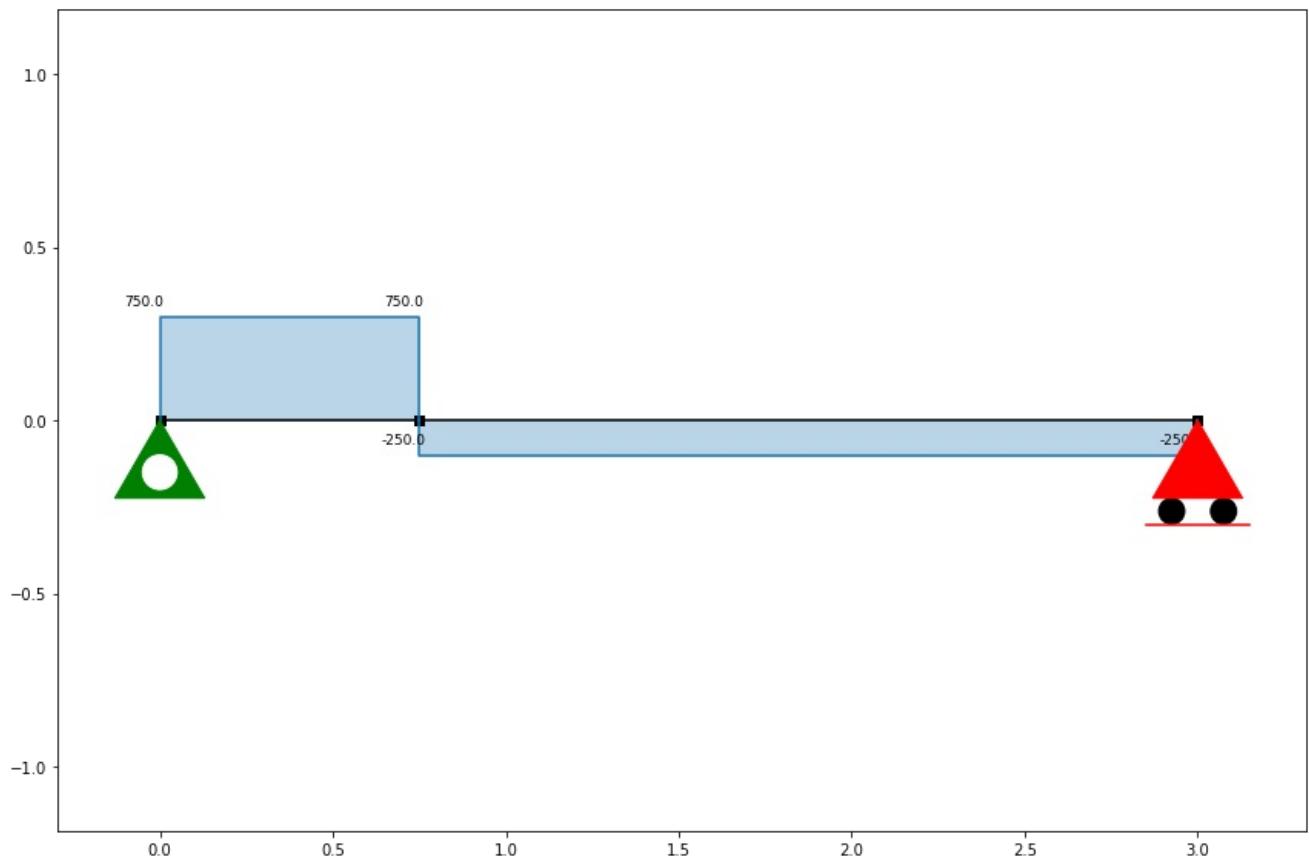
```
In []: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

*Nodo: 1
Reacción Fy: 750.0
*Nodo: 3
Reacción Fy: 250.0000000000023
```

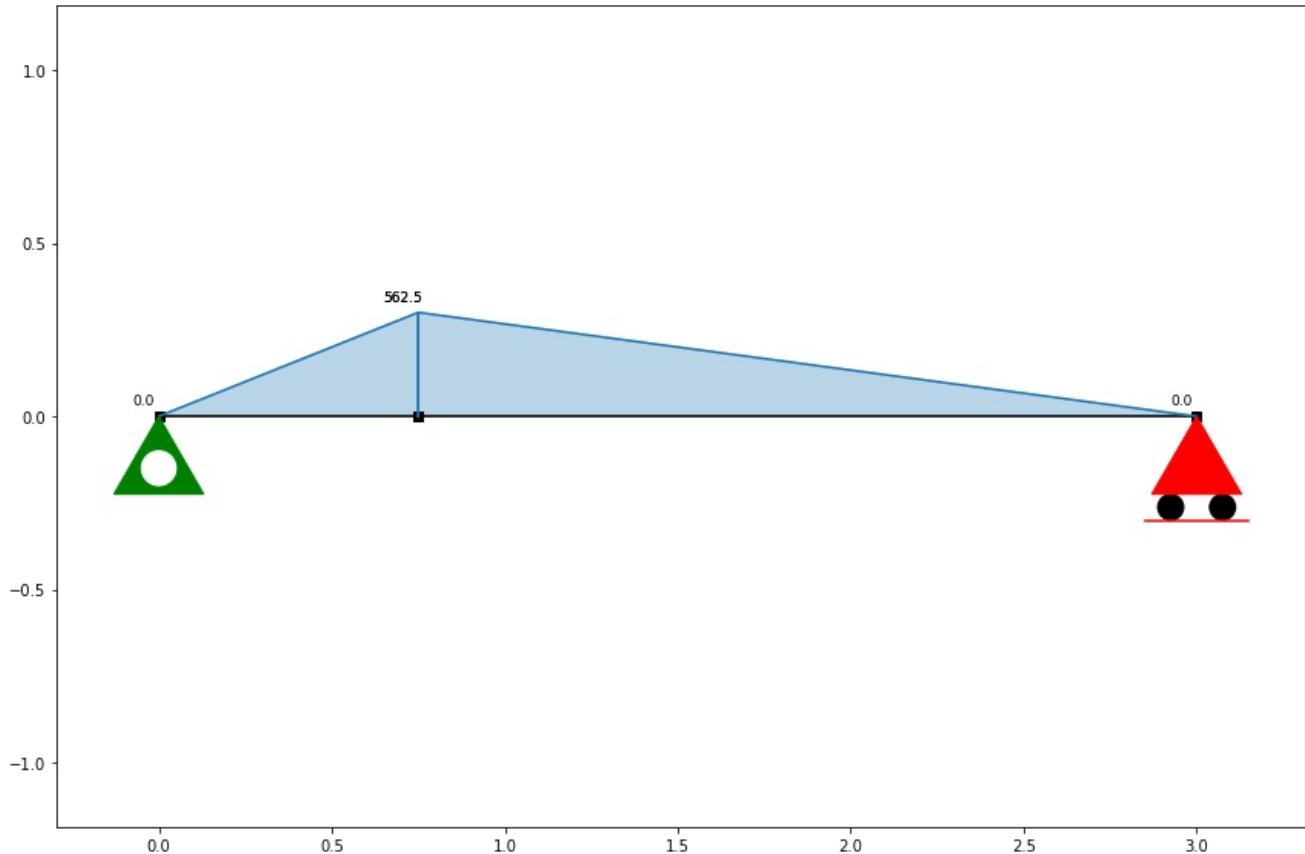


```
In []: # Mostramos cortantes
ss.show_shear_force()
```

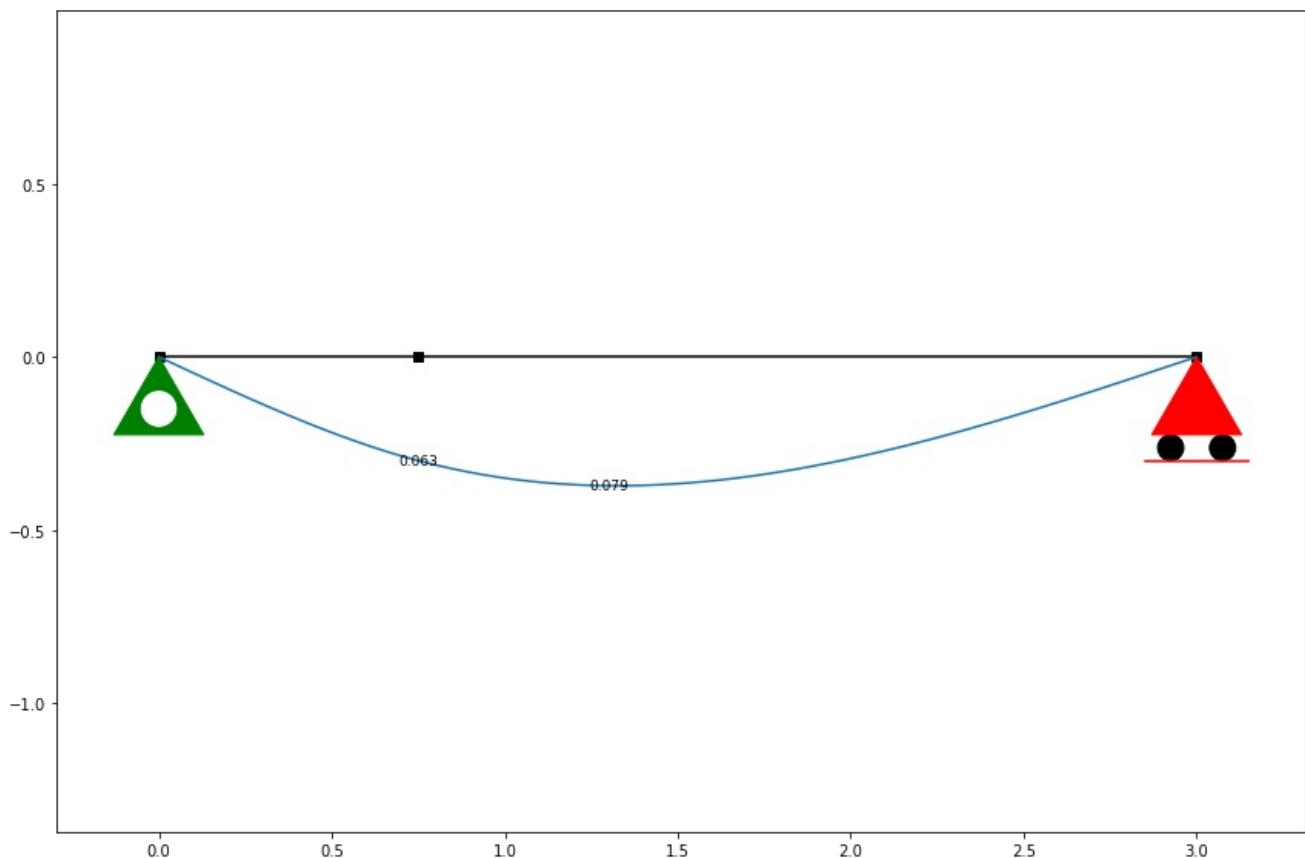


```
In []: # Mostramos flexores
```

```
In []: ss.show_bending_moment()
```



```
In []: # Mostramos deformada
ss.show_displacement()
```



```
In []:
```

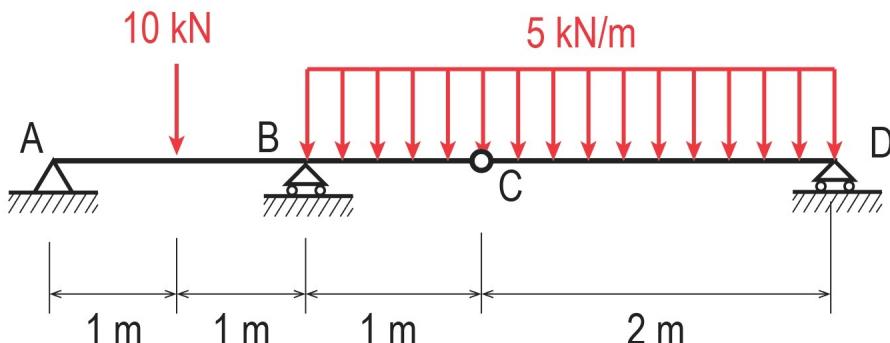
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

# ESTRUCTURAS I - CURSO 2021-2022

Universidad de Granada

## PRÁCTICA 1:

Determinar los diagramas de esfuerzos de la siguiente estructura.



```
In []: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-mnhohi3r
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-mnhohi3r
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.4.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (1.3.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.4.7)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0) (1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=58914 sha256=e86b408b3ca2546aaa09ed9cd1aeed46f62dbecc28ebd6fc0797d68bc4b656fd
 Stored in directory: /tmp/pip-ephem-wheel-cache-qm6gzco7/wheels/23/97/1a/d460d2d29ccd27f0842a8e862290c78834c55be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

```
In []: # Cargamos el software anastruct
from anastruct import SystemElements

Cargamos el objeto de nuestra estructura
ss = SystemElements()
```

```
In []: # Construcción de la estructura

Viga 1
ss.add_element(location=[[0, 0], [1, 0]]);
Viga 2
ss.add_element(location=[[1, 0], [2, 0]]);
Viga 3
ss.add_element(location=[[2, 0], [3, 0]], spring={2: 0});
Viga 4
ss.add_element(location=[[3, 0], [5, 0]]);
```

```
In []: # Condiciones de contorno

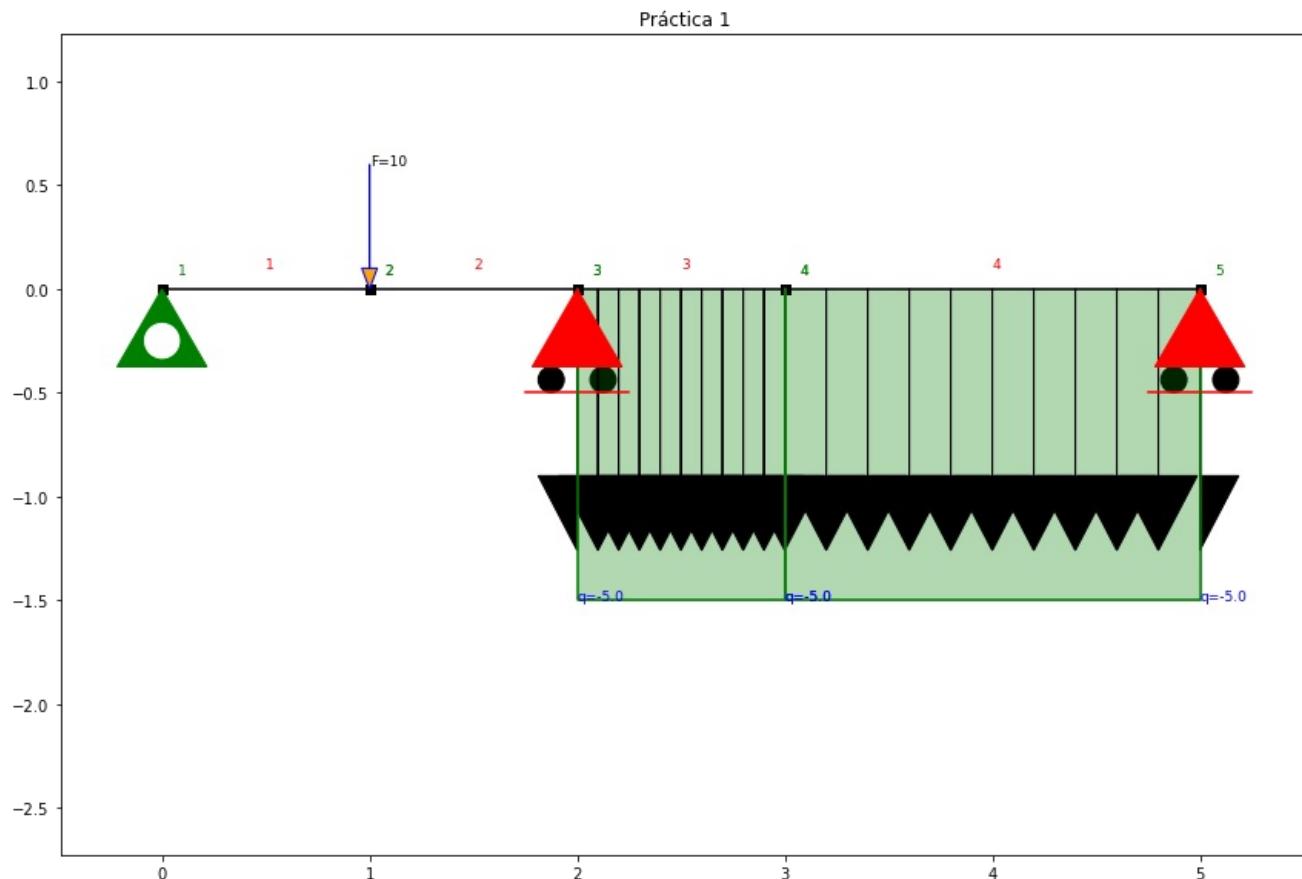
Añadimos apoyo fijo al nodo 1
ss.add_support_hinged(node_id=1)
Añadimos carrito al nodo 3
ss.add_support_roll(node_id=3, direction=2)
Añadimos carrito al nodo 5
ss.add_support_roll(node_id=5, direction=2)
```

```
In []: # Cargas
Añadimos carga puntual al nodo 2
ss.point_load(2, Fx=0, Fy=-10)

Añadimos carga distribuida
ss.q_load(element_id=3, q=5)

Añadimos carga distribuida
ss.q_load(element_id=4, q=5)
```

```
In []: # Mostramos estructura generada
ss.show_structure(title='Práctica 1')
```

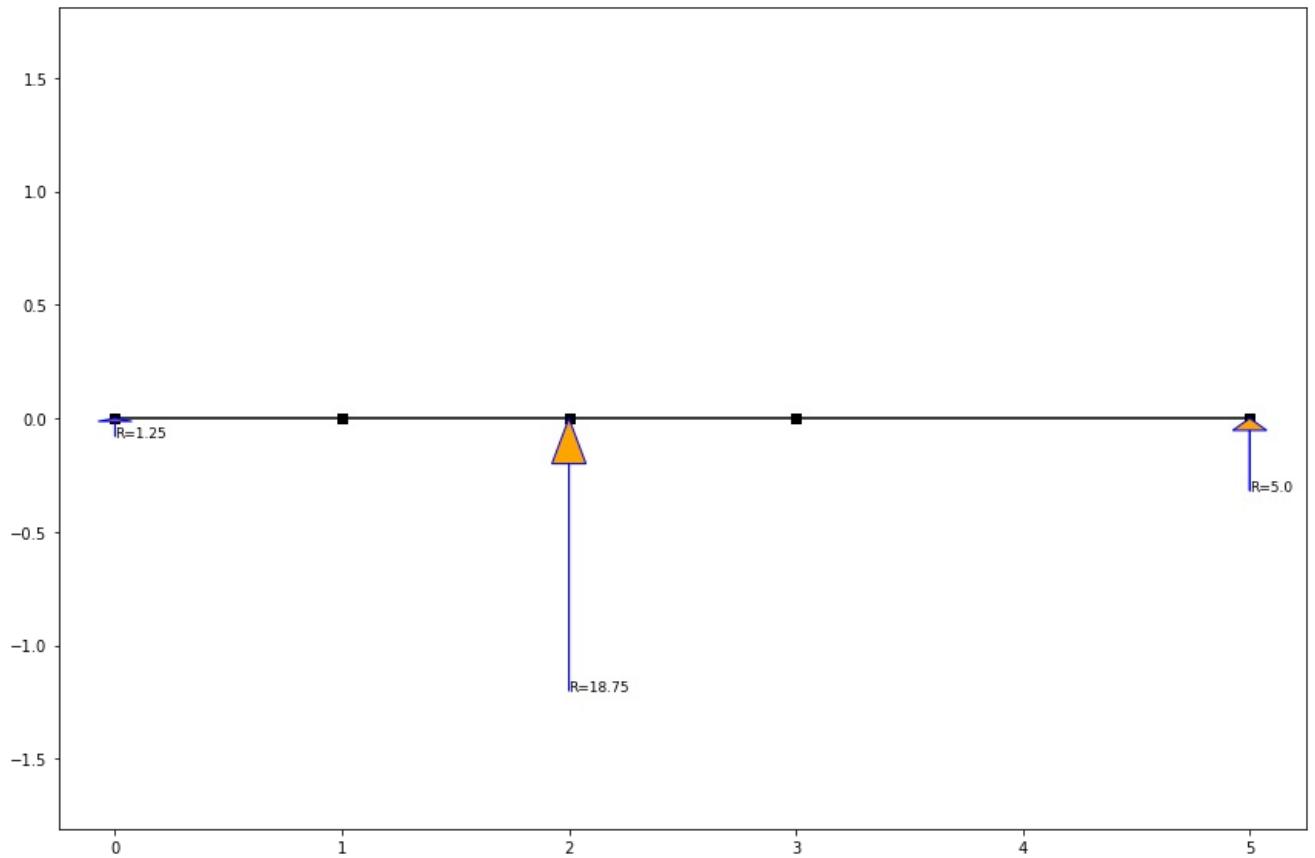


```
In []: # Resolvemos la estructura
ss.solve();
```

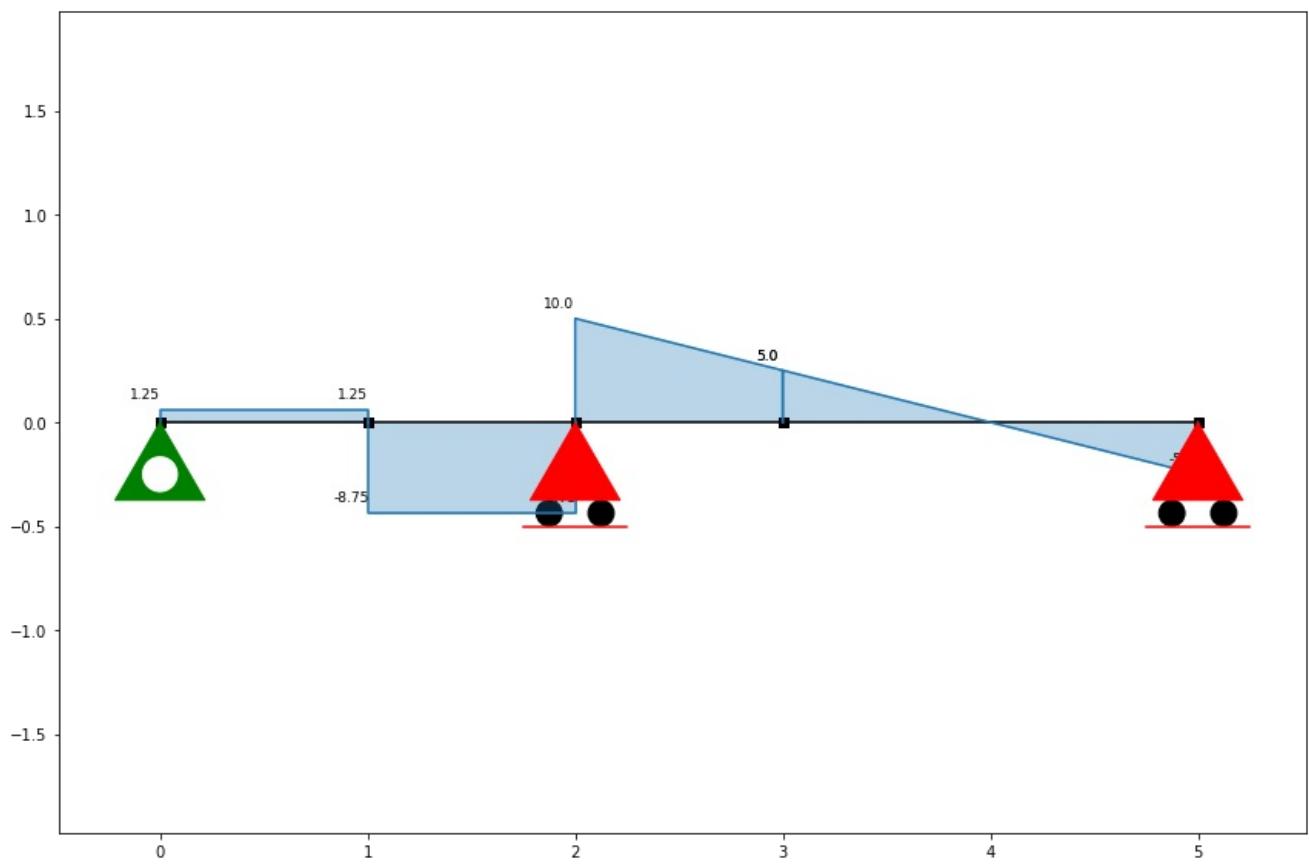
```
In []: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

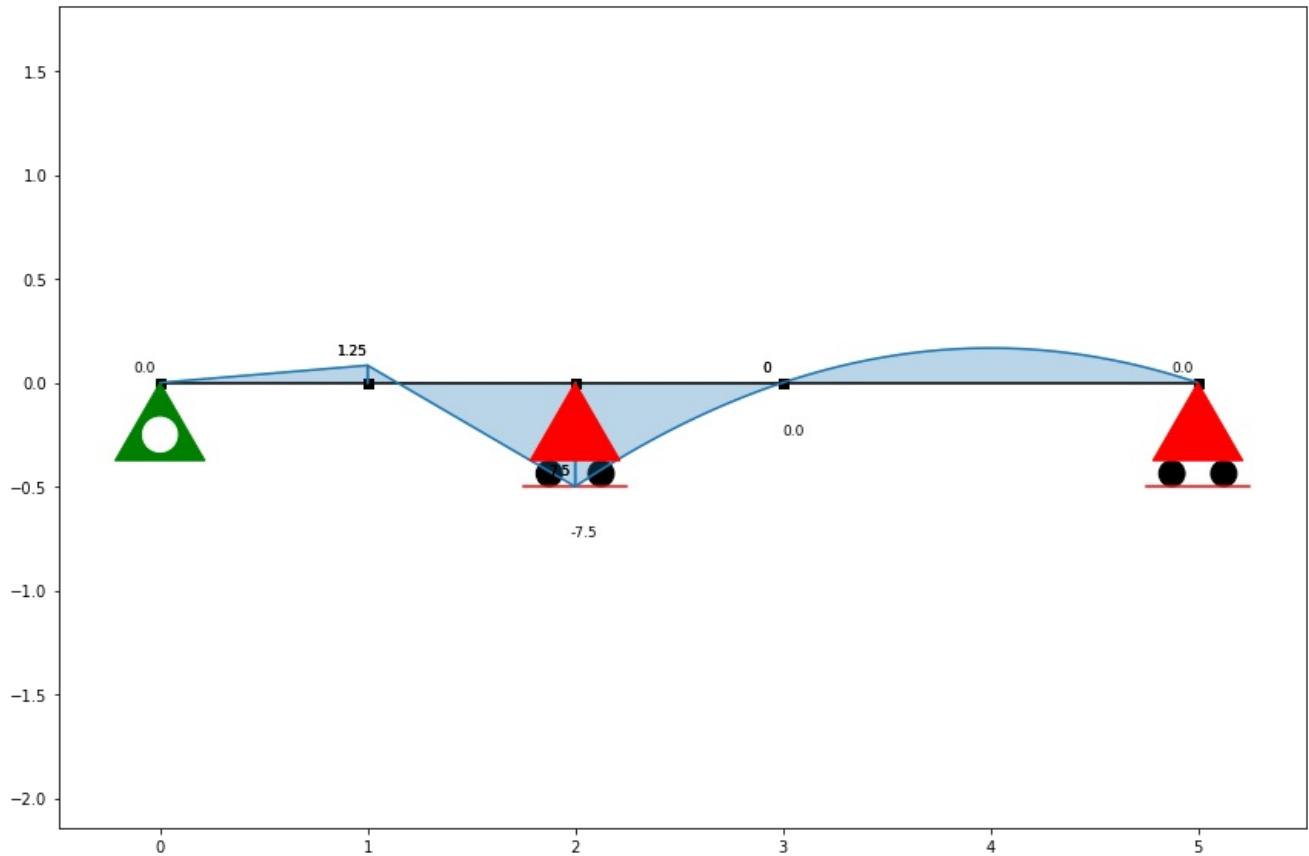
*Nodo: 1
Reacción Fy: 1.2500000000000002
*Nodo: 3
Reacción Fy: 18.75
*Nodo: 5
Reacción Fy: 5.0
```



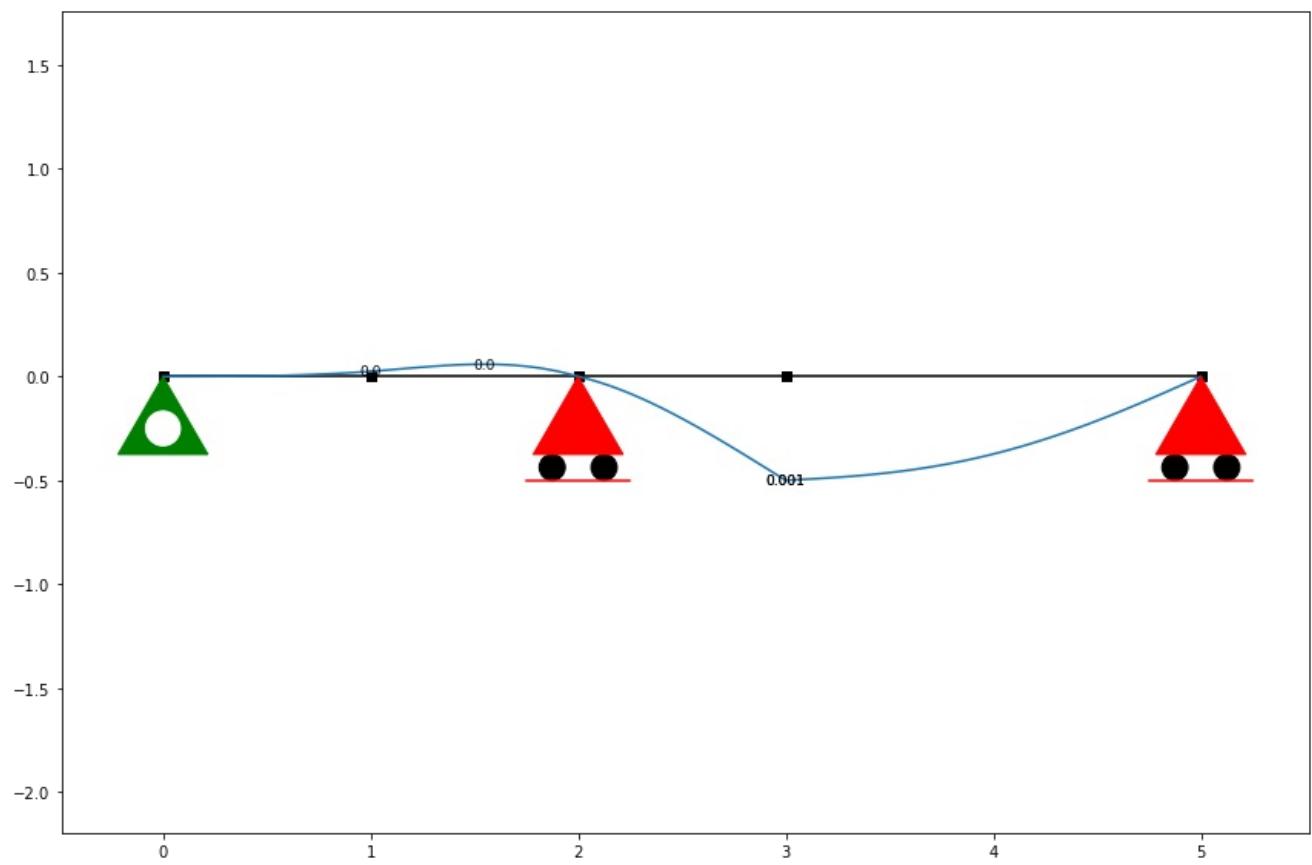
```
In []: # Mostramos cortantes
ss.show_shear_force()
```



```
In []: # Mostramos flectores
ss.show_bending_moment()
```



```
In []: # Mostramos deformada
ss.show_displacement()
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In [1]: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-s911xqvq
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-s911xqvq
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.4.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!>2.1.2,!>2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages
(from matplotlib>=3.0->anastruct==0.0.0)
(2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(1.3.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0)
(1.15.0)
```

```
In [2]: from anastruct import SystemElements
ss = SystemElements()

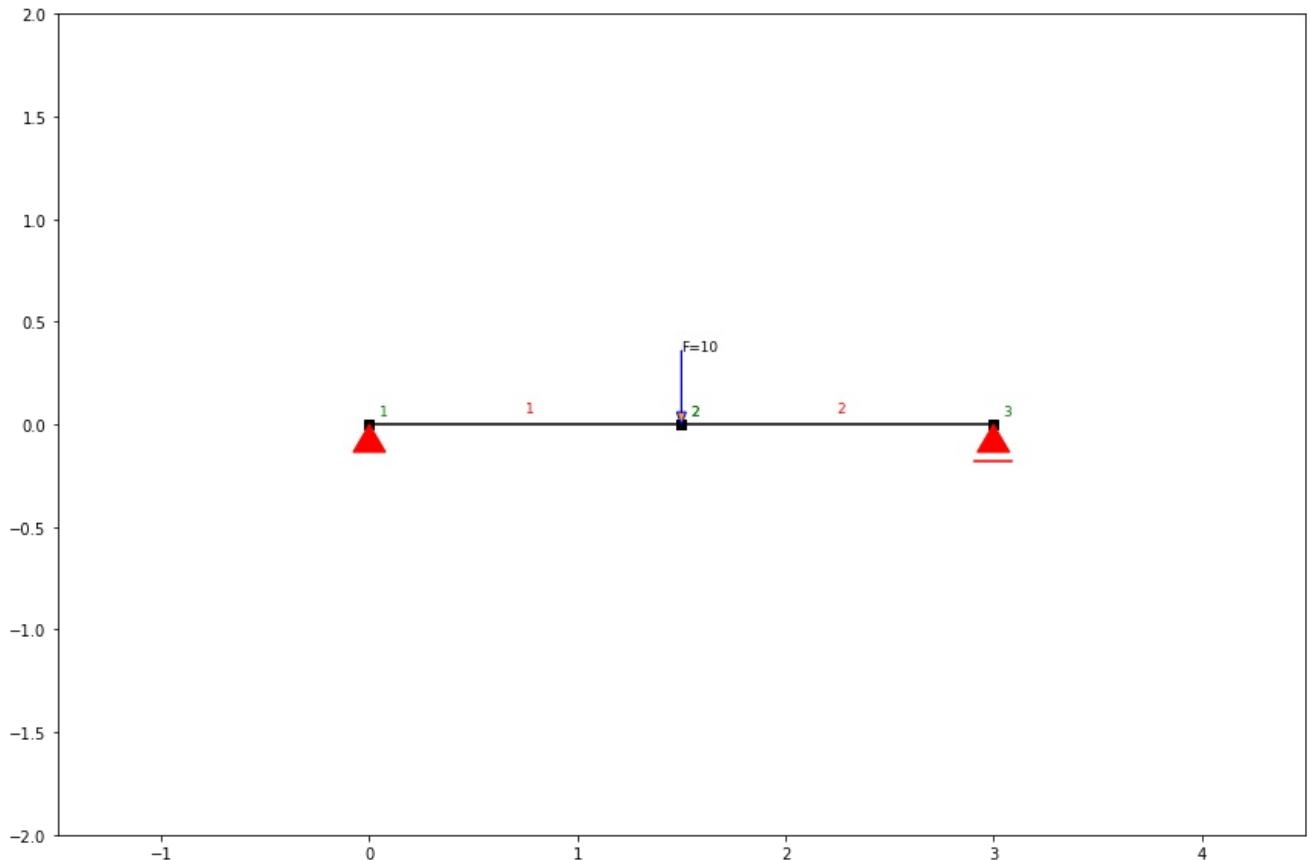
L = 3.0 # Longitud de la barra
P = -10.0 # Carga puntual

Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [L/2.0, 0]])
Añadimos elemento barra 2
ss.add_element(location=[[L/2.0, 0], [L, 0]])

Añadimos apoyo fijo al nodo 1
ss.add_support_hinged(node_id=1)
Añadimos carrito al nodo 3
ss.add_support_roll(node_id=3, direction=2)

Añadimos carga puntual al nodo 2
ss.point_load(2, Fx=P, rotation=-90)

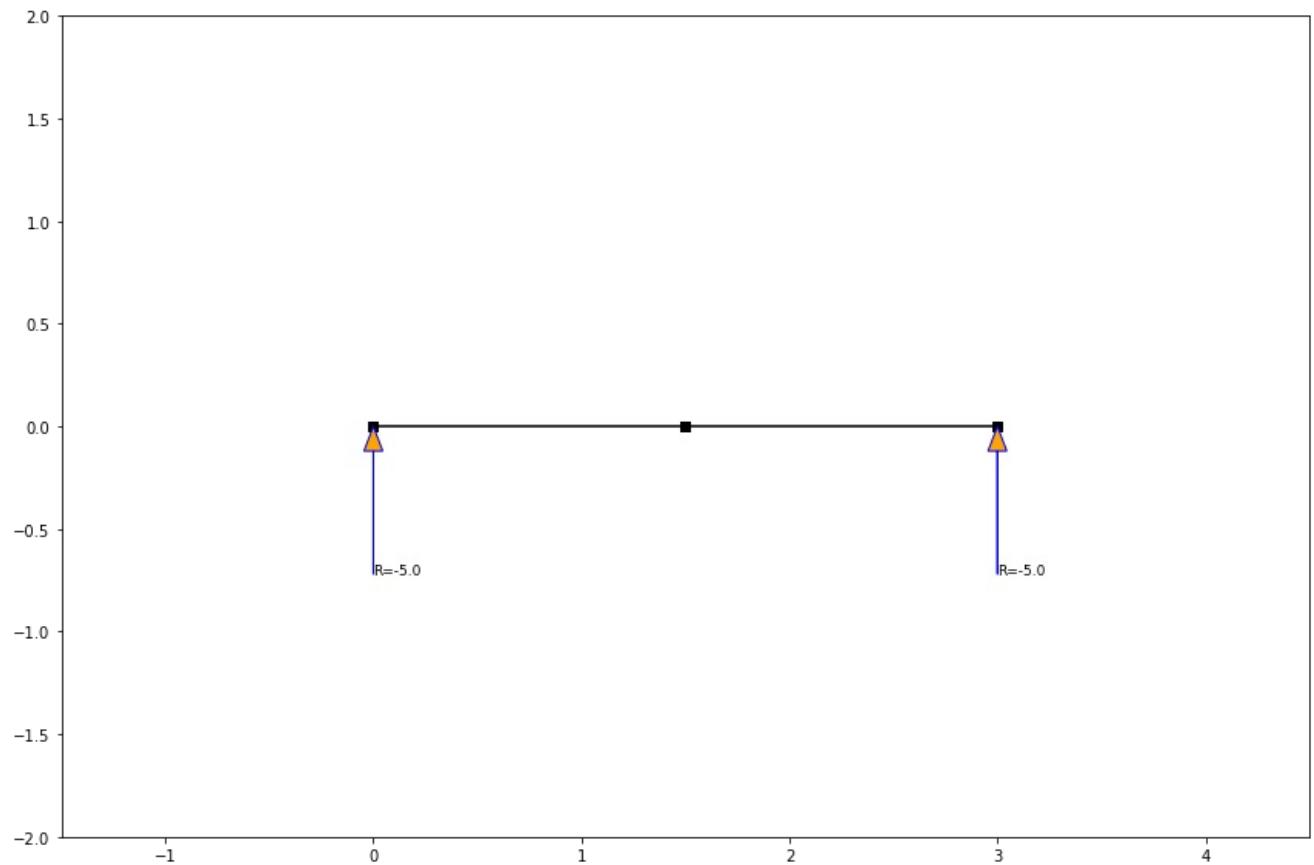
Mostramos estructura generada
ss.show_structure()
```



```
In [3]: # Resolvemos la estructura
ss.solve()
```

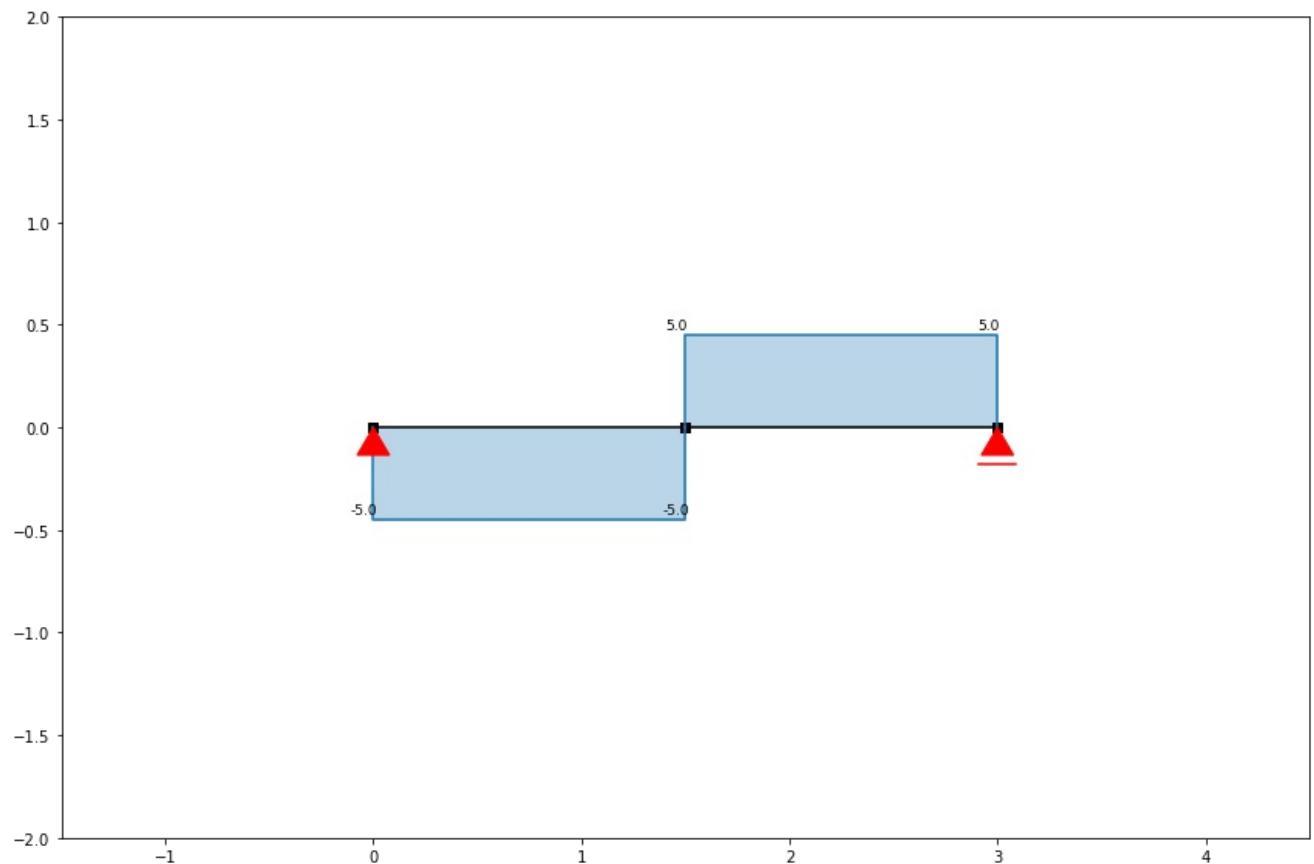
```
Out[3]: array([0.0000000e+00, 0.0000000e+00, -1.1250000e-03, -6.12323400e-20,
 1.1250000e-03, -2.06432094e-22, -6.12323400e-20, 0.0000000e+00,
 1.1250000e-03])
```

```
In [7]: # Mostramos las reacciones
ss.show_reaction_force()
```

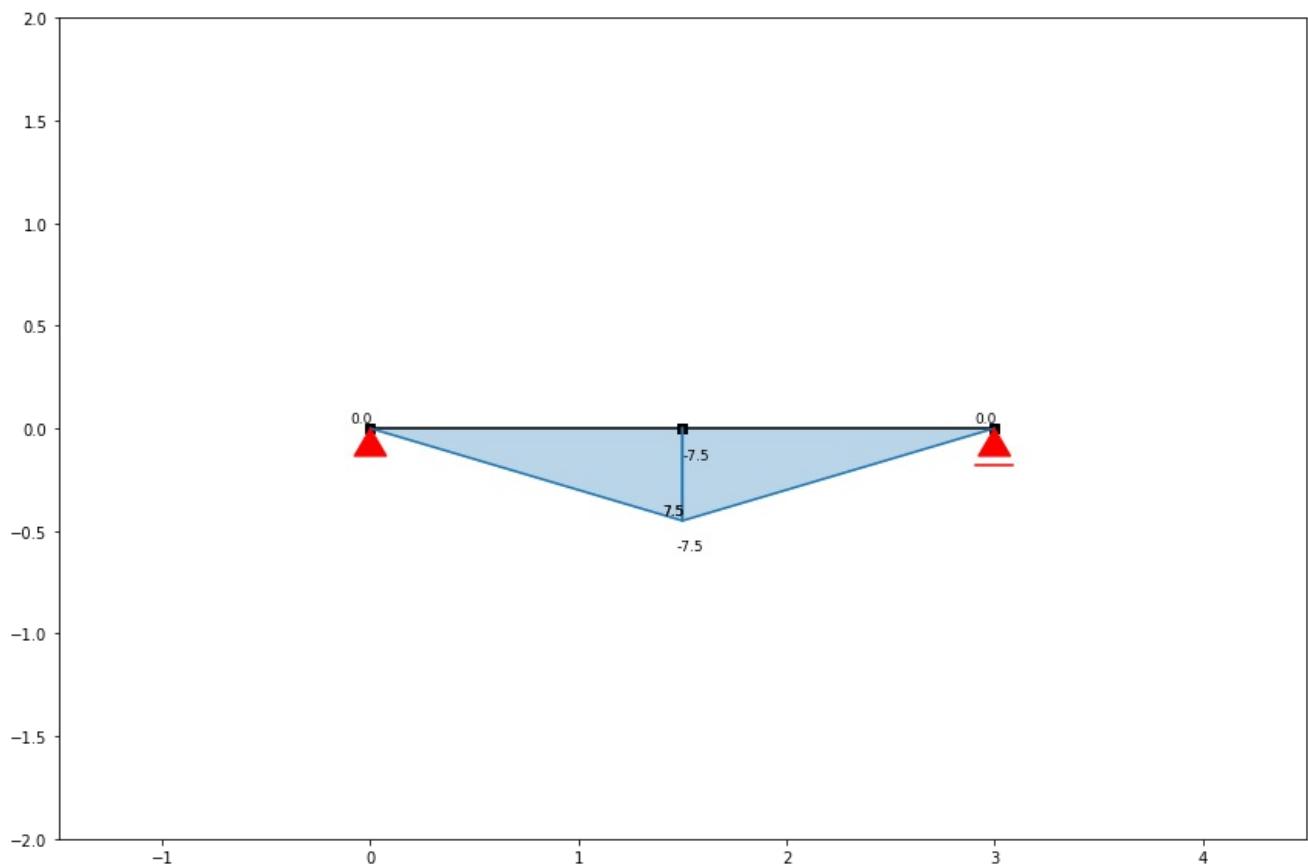


```
In [5]: # Mostramos cortantes
ss.show_shear_force()
```

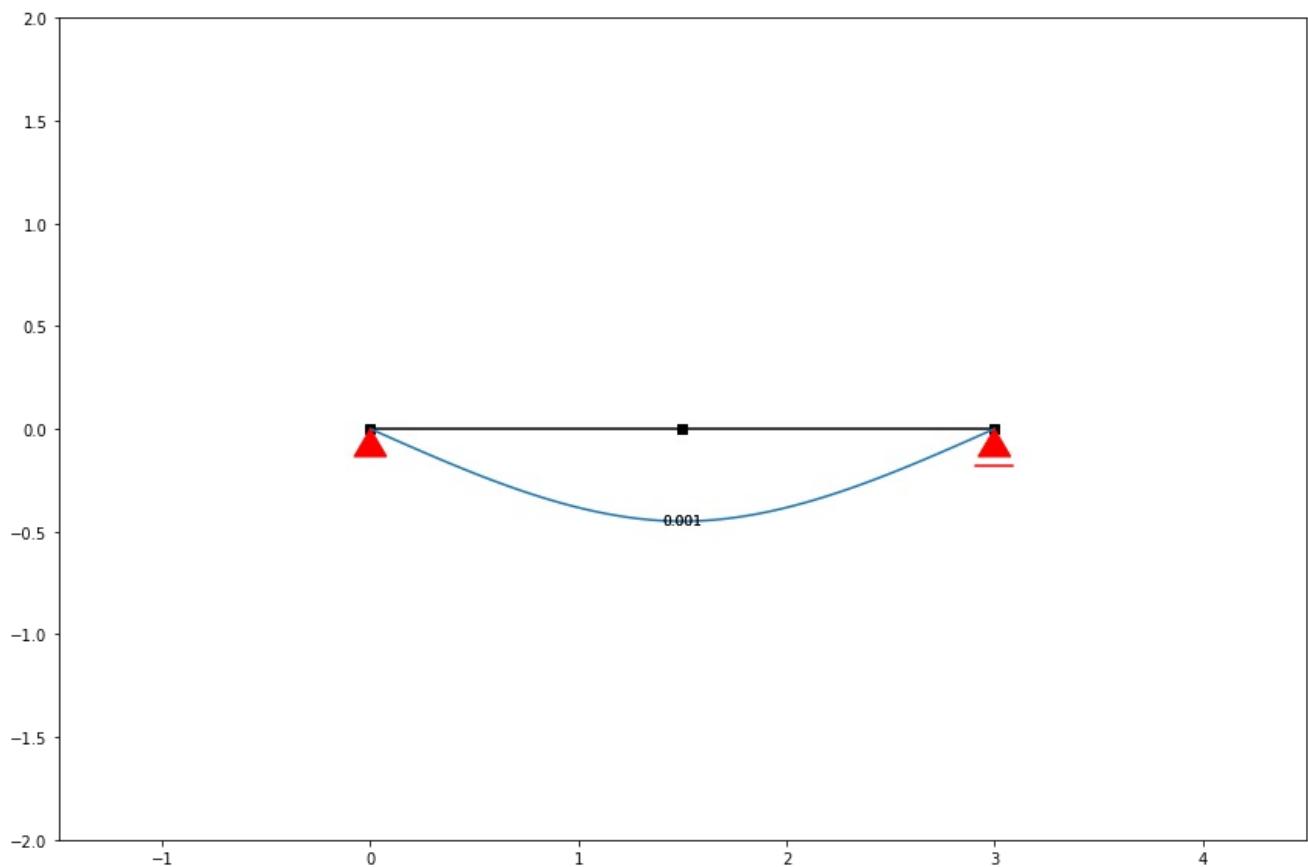
dale  
dale



```
In []: # Mostramos flectores
ss.show_bending_moment()
```



```
In []: ss.show_displacement()
```



```
In []: ss.system_force_vector
ss.loads_point
```

```
Out[]: {2: (-6.123233995736766e-16, 10.0)}
```

```
In []: print('dale')
dale
```

```
In []:
```

# ESTRUCTURAS I - CURSO 2021-2022

Universidad de Granada

## EJERCICIO 1:

Dada la viga simplemente apoyada de la figura. Determine:

1. Reacciones en los apoyos
2. Ley de esfuerzos cortantes
3. Ley de esfuerzos flectores



Lo primero que debemos hacer es conectarnos a un servidor de Google. Para ello, debemos clickar sobre la pestaña "Conectar" en la esquina superior derecha. Esto nos permitirá acceder a un ordenador que Google nos cederá de forma gratuita. De este modo, no será nuestro propio ordenador el que realice los cálculos, sino el ordenador al que se nos ha dado acceso. Una vez hayamos clickado en "Conectar", deberemos esperar unos segundos hasta que se nos habilite el acceso. Cuando esto ocurra, aparecerá un tick en verde, y se nos indicarán los valores de memoria RAM y de disco que se nos ha sido asignada.

A continuación, debemos cargar el paquete de código de "anastruct". Para ello, en la celda inferior tenemos el código necesario para acceder a la última versión del mismo. Cada celda se ejecuta clickando sobre el botón (►) en la parte izquierda de la misma. Alternativamente, podemos teclear la combinación shift (↑) e intro.

Fíjate que en las siguientes líneas de código, se definen muchos comentarios precedidos del símbolo (#). Python va a entender que el texto que sigue es un comentario, y por lo tanto no hará nada. Esto nos sirve para hacer anotaciones para posteriormente recordar los pasos seguidos.

```
In [19]: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-sxubhu5u
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-sxubhu5u
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.4.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages
(from matplotlib>=3.0->anastruct==0.0.0)
(2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(0.10.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0)
(1.15.0)
```

```
In [28]: # Cargamos el software anastruct
from anastruct import SystemElements

Cargamos el objeto de nuestra estructura
ss = SystemElements()
```

```
In [29]: # Vamos a definir ahora algunas variables
L = 3.0 # Longitud de la barra [m]
P = -1000.0 # Carga puntual [N]
a = 0.75 # Distancia a la que se encuentra la carga aplicada [m]
```

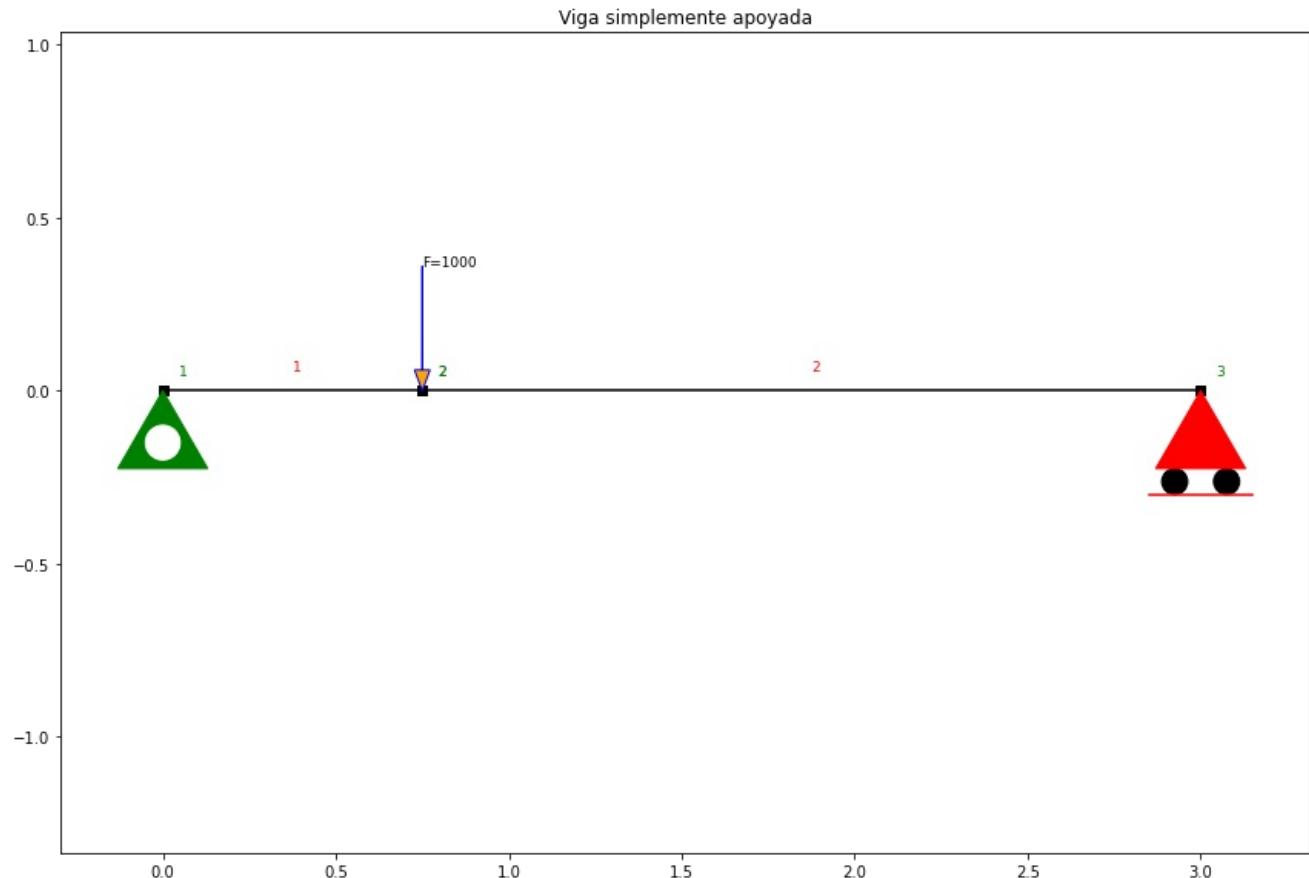
```
In [30]: # Construcción de la estructura
```

```
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [a, 0]]);
Añadimos elemento barra 2
ss.add_element(location=[[a, 0], [L, 0]]);
```

```
In [31]: # Añadimos apoyo fijo al nodo 1
ss.add_support_hinged(node_id=1)
Añadimos carrito al nodo 3
ss.add_support_roll(node_id=3, direction=2)
```

```
In [32]: # Añadimos carga puntual al nodo 2
ss.point_load(2, Fx=0, Fy=P)
```

```
In [33]: # Mostramos estructura generada
ss.show_structure(title='Viga simplemente apoyada')
```

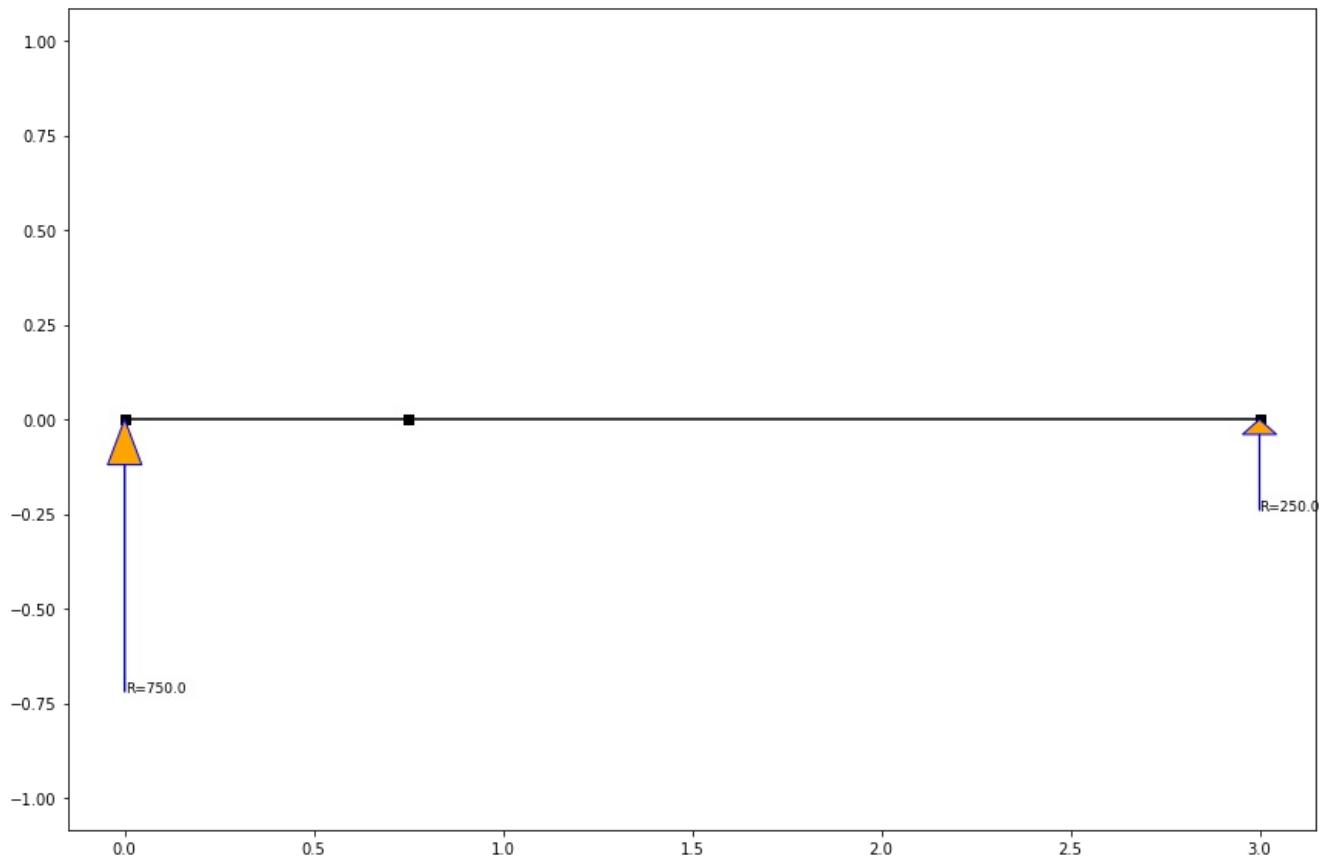


```
In [35]: # Resolvemos la estructura
ss.solve();
```

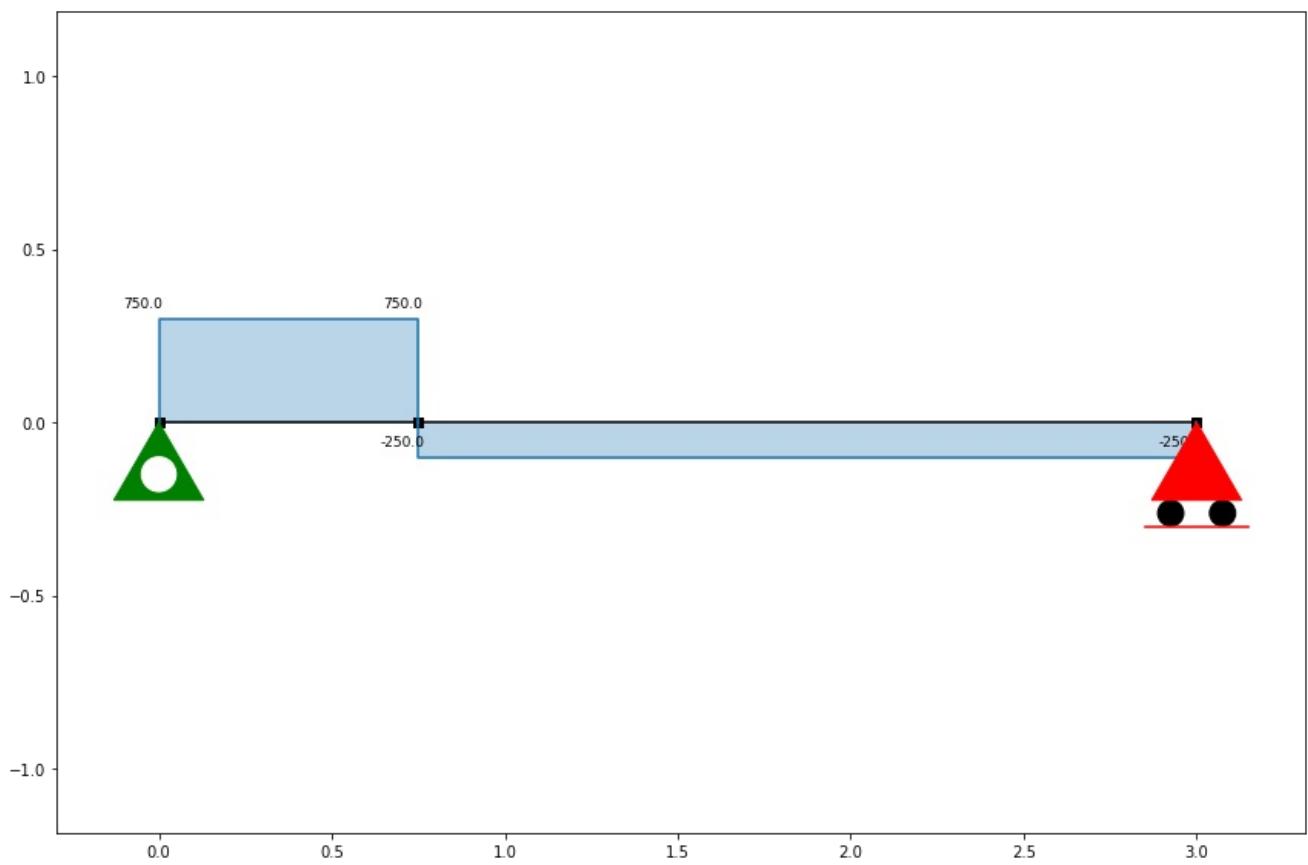
```
In [36]: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

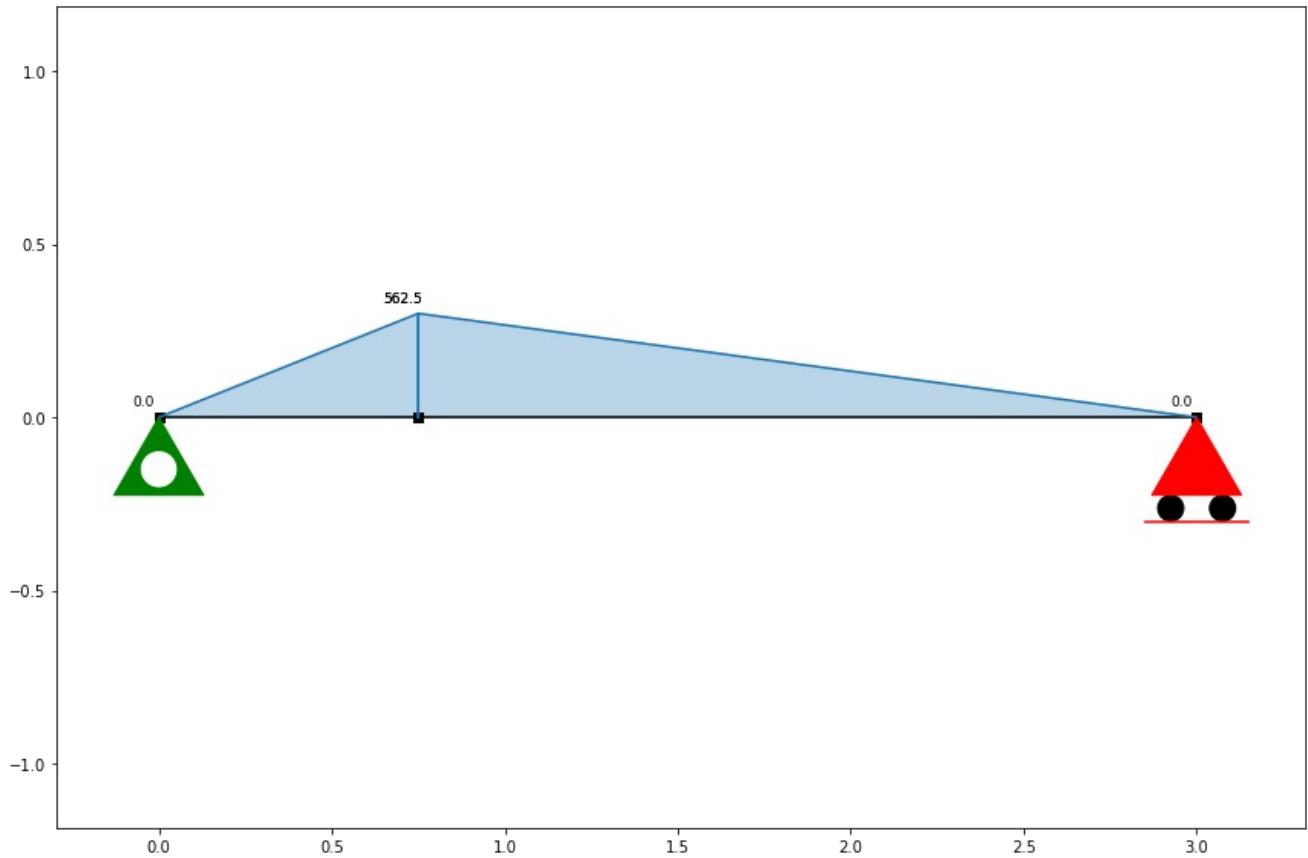
*Nodo: 1
Reacción Fy: 750.0
*Nodo: 3
Reacción Fy: 250.00000000000023
```



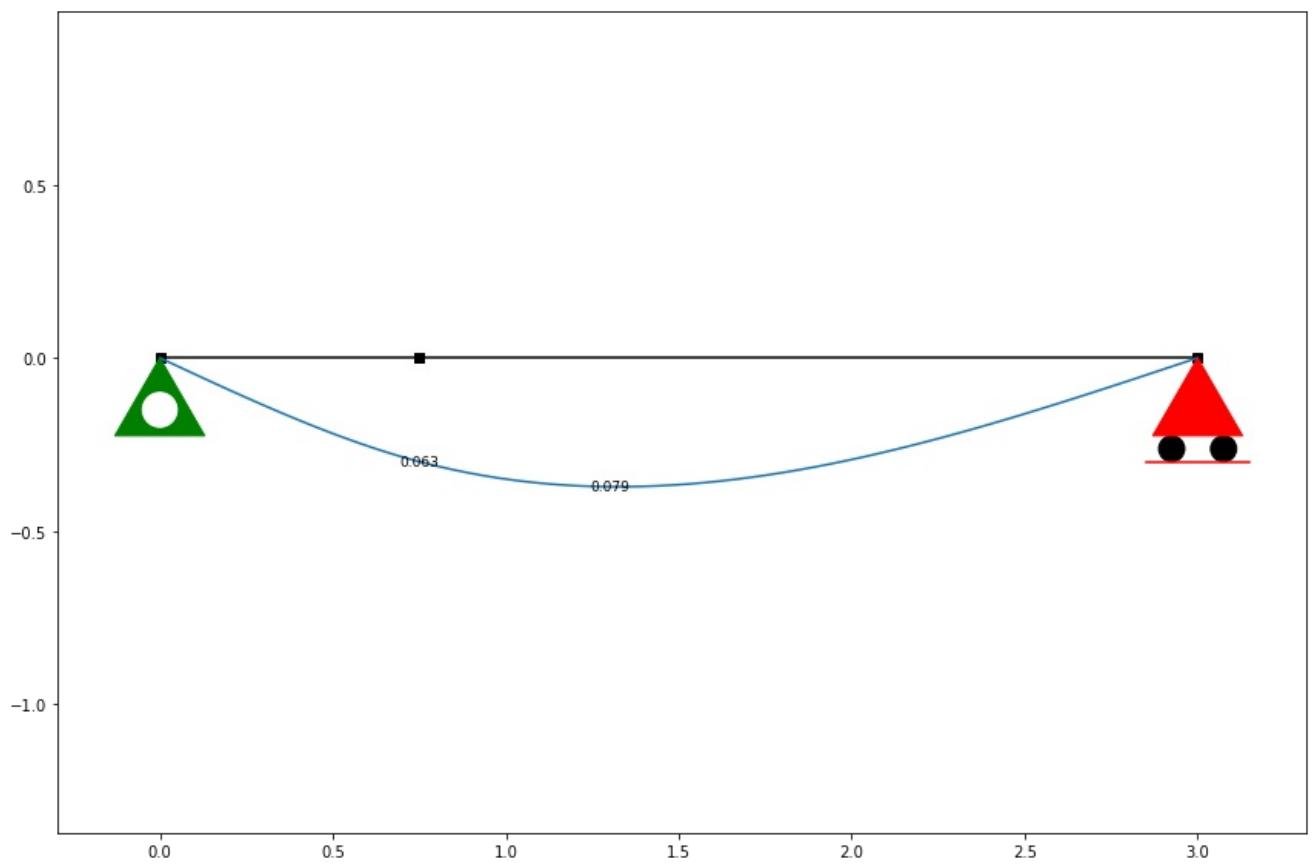
```
In [37]: # Mostramos cortantes
ss.show_shear_force()
```



```
In [38]: # Mostramos flectores
ss.show_bending_moment()
```



```
In [39]: # Mostramos deformada
ss.show_displacement()
```



In [ ]:

```
In [1]: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-qg9xbbj6
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-qg9xbbj6
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.4.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!!=2.1.2,!!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages
(from matplotlib>=3.0->anastruct==0.0.0)
(2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(1.3.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0)
(1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=58944 sha256=702b2d629e7839347852
a1e568d124c03dea415e2ba567c2aa403a0578182688
 Stored in directory: /tmp/pip-ephem-wheel-cache-sa5z_m1f/wheels/23/97/1a/d460d2d29ccd27f0842a8e862290c78834c5
5be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

```
In [2]: from anastruct import SystemElements

ss = SystemElements()

L = 3.0 # Longitud de la barra
M = 200.0 # Momento puntual

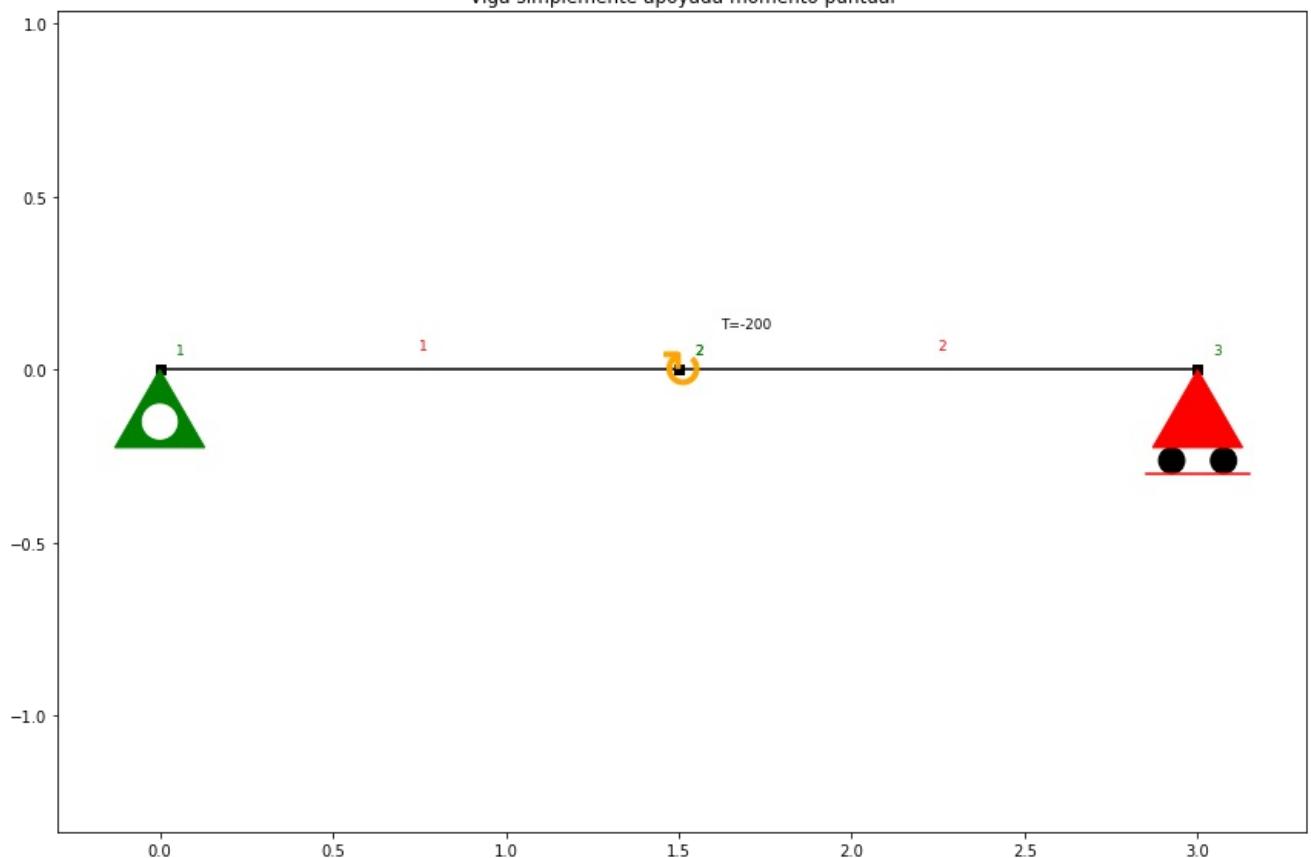
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [L/2.0, 0]])
Añadimos elemento barra 2
ss.add_element(location=[[L/2.0, 0], [L, 0]])

Añadimos apoyo fijo al nodo 1
ss.add_support_hinged(node_id=1)
Añadimos carrito al nodo 3
ss.add_support_roll(node_id=3, direction=2)

Añadimos momento puntual al nodo 2
ss.moment_load(2, Ty=M)

Mostramos estructura generada
ss.show_structure(title='Viga simplemente apoyada momento puntual')
```

Viga simplemente apoyada momento puntual



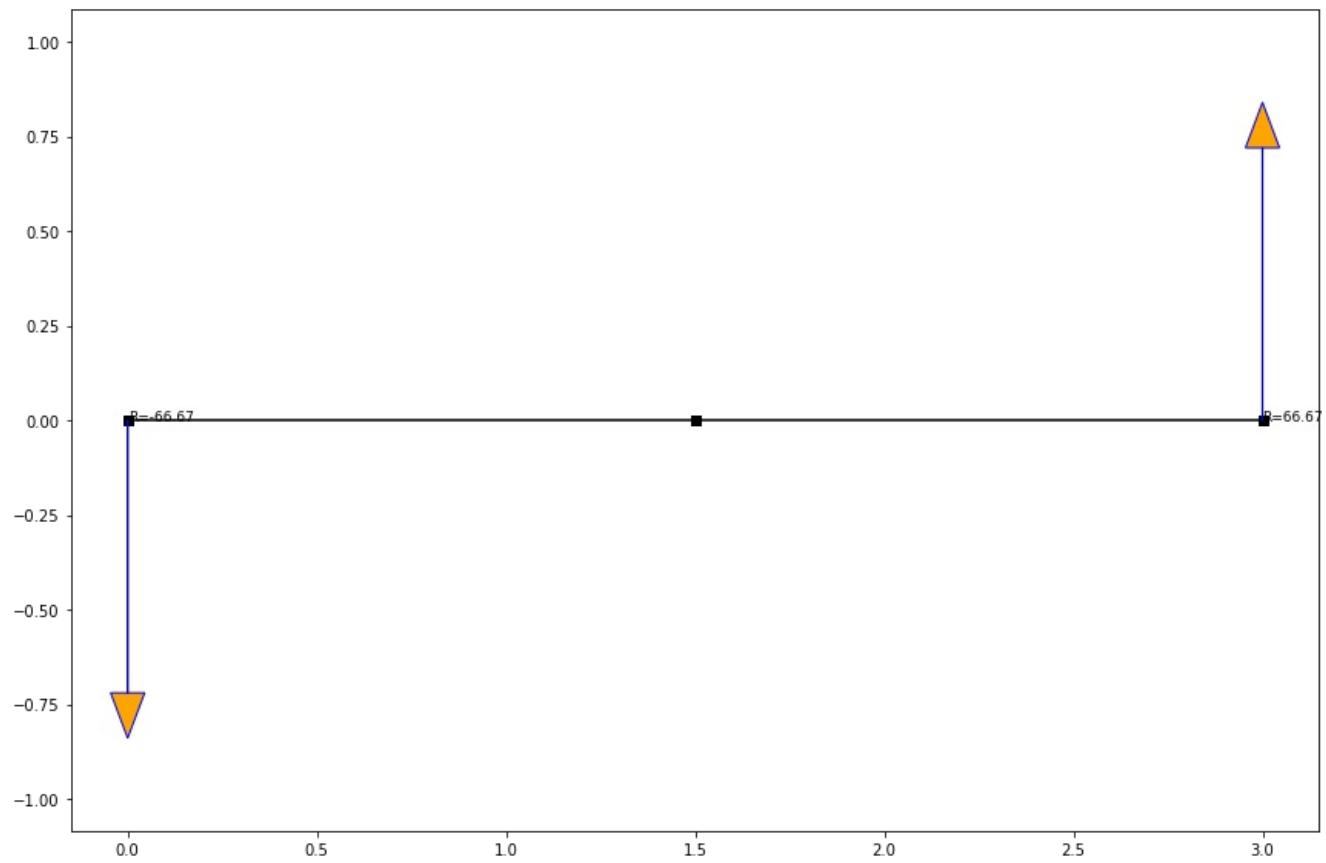
```
In [3]: # Resolvemos la estructura
ss.solve()
```

```
Out[3]: array([-0.00000000e+00, 0.00000000e+00, -5.00000000e-03, -0.00000000e+00,
 1.04782849e-19, 1.00000000e-02, -0.00000000e+00, 0.00000000e+00,
 -5.00000000e-03])
```

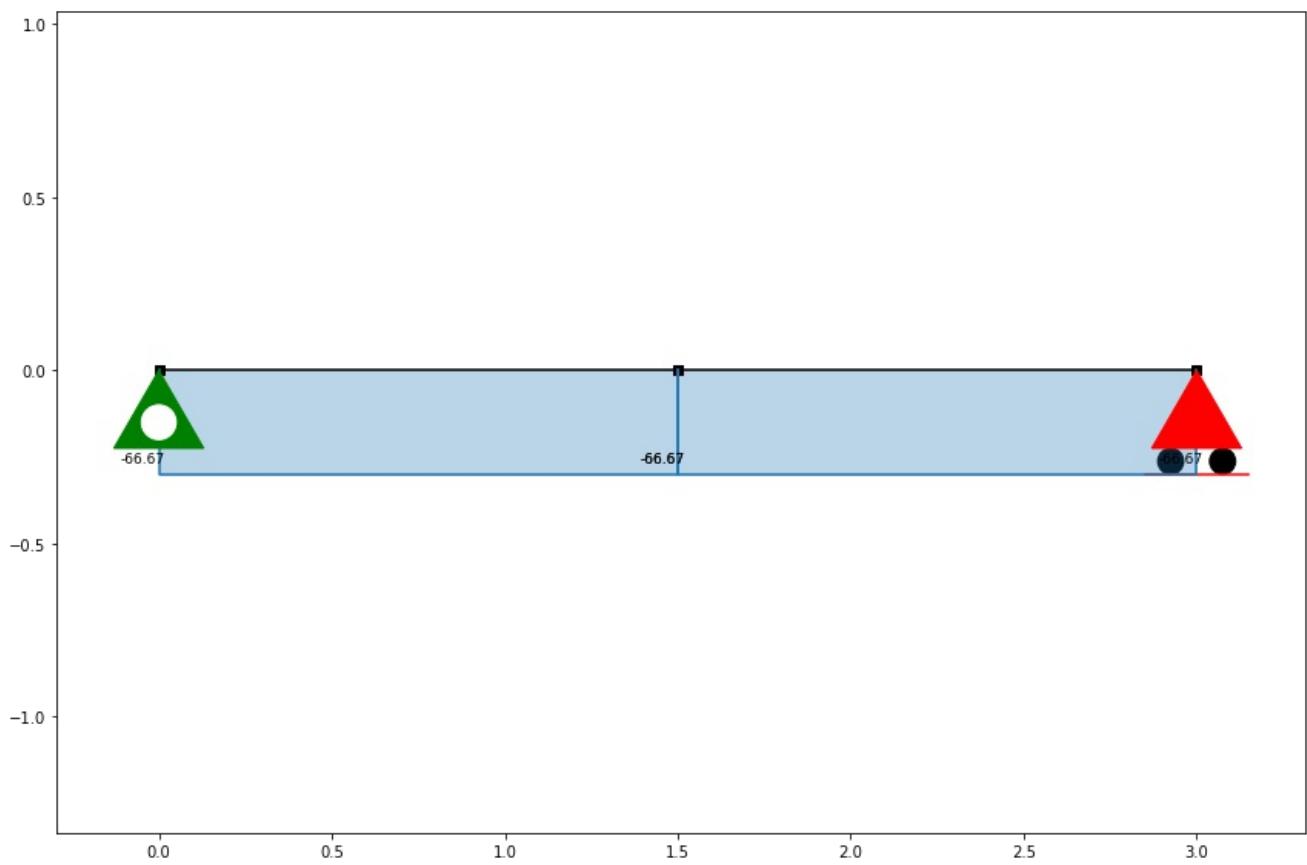
```
In [4]: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

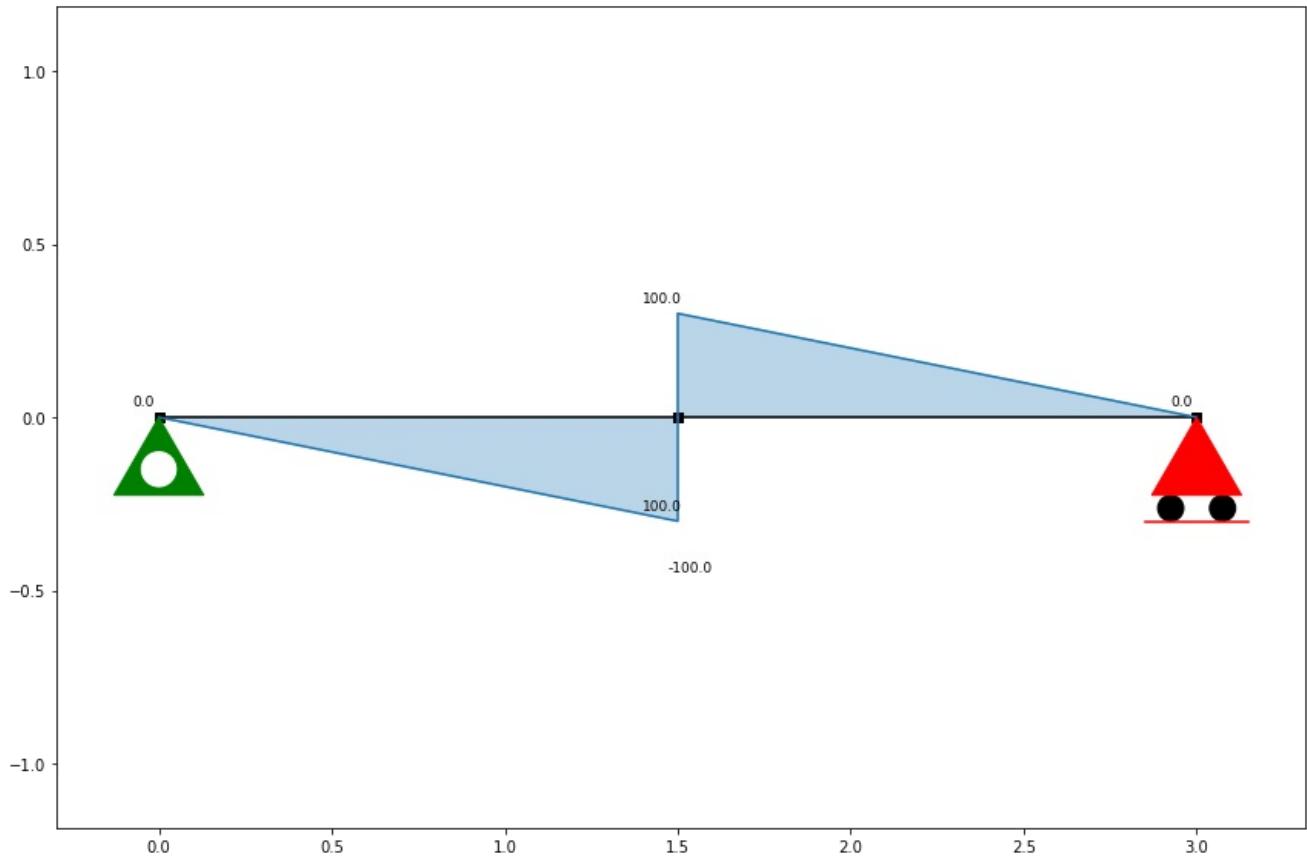
0
*Nodo: 1
Reacción Fy: -66.66666666666666
0
*Nodo: 3
Reacción Fy: 66.66666666666666
```



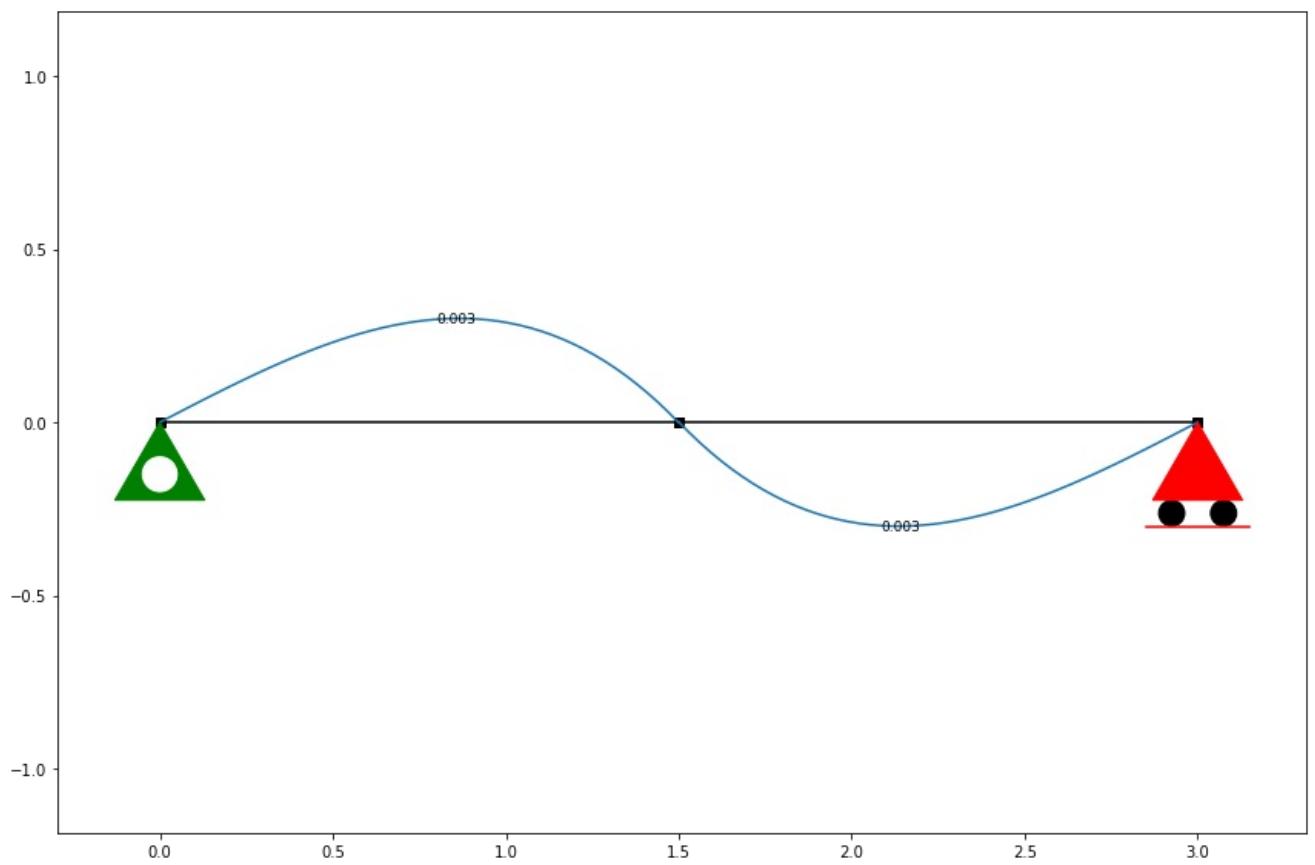
```
In [5]: # Mostramos cortantes
ss.show_shear_force()
```



```
In [6]: # Mostramos flectores
ss.show_bending_moment()
```



```
In [7]: ss.show_displacement()
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [2]:

```
Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-coebcpio
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-coebcpio
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.4.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (1.3.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.8.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0) (1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=58944 sha256=5659e922ae6d120a456c153cb14ddc9eb3cf89b7a52db9b12f0cb8ccca14d5cb
 Stored in directory: /tmp/pip-ephem-wheel-cache-cm85fegf/wheels/23/97/1a/d460d2d29ccd27f0842a8e862290c78834c55be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

In [3]:

```
from anastruct import SystemElements

ss = SystemElements()

L = 3.0 # Longitud de la barra
q = 200.0 # Carga puntual

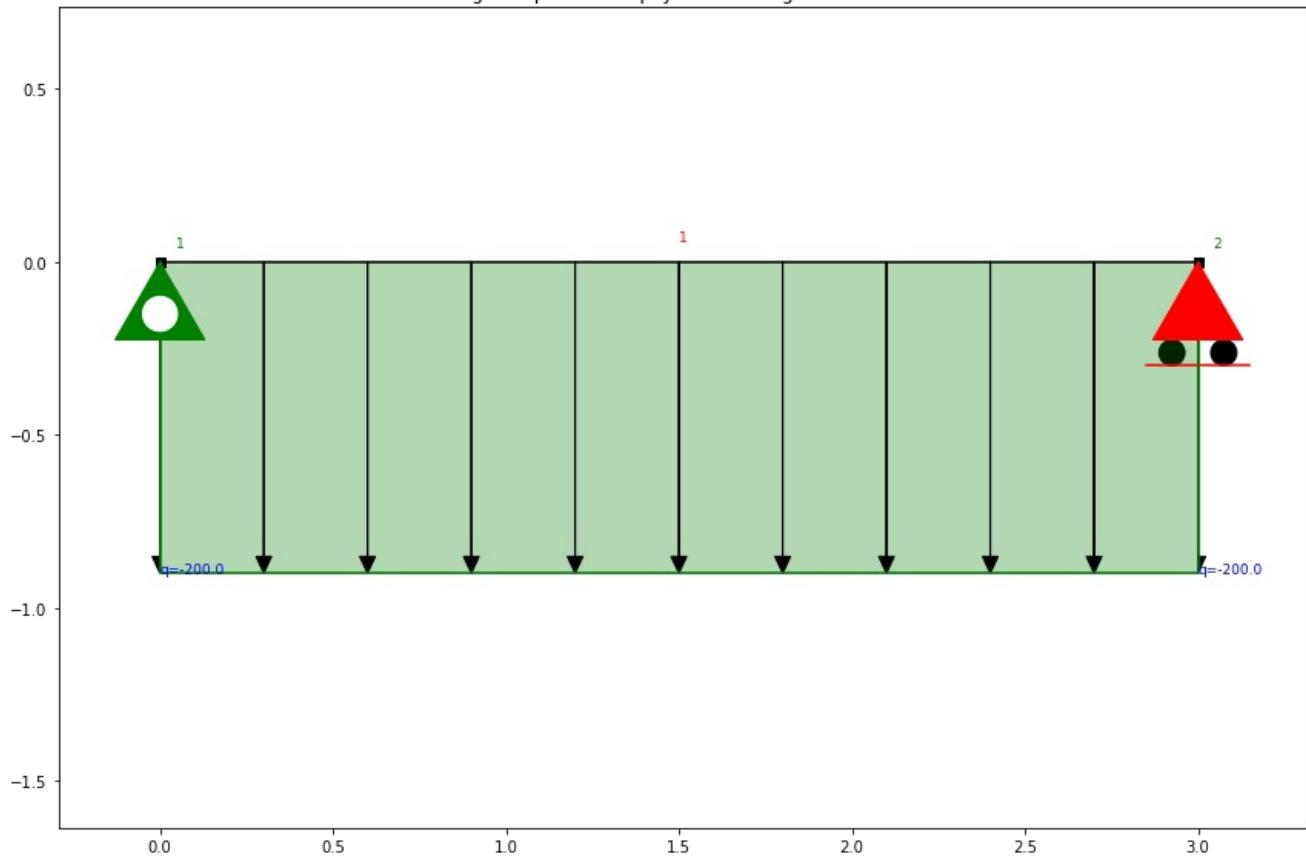
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [L, 0]])

Añadimos apoyo fijo al nodo 1
ss.add_support_hinged(node_id=1)
Añadimos carrito al nodo 3
ss.add_support_roll(node_id=2, direction=2)

Añadimos carga distribuida
ss.q_load(element_id=1, q=q)

Mostramos estructura generada
ss.show_structure(title='Viga simplemente apoyada con carga distribuida')
```

Viga simplemente apoyada con carga distribuida



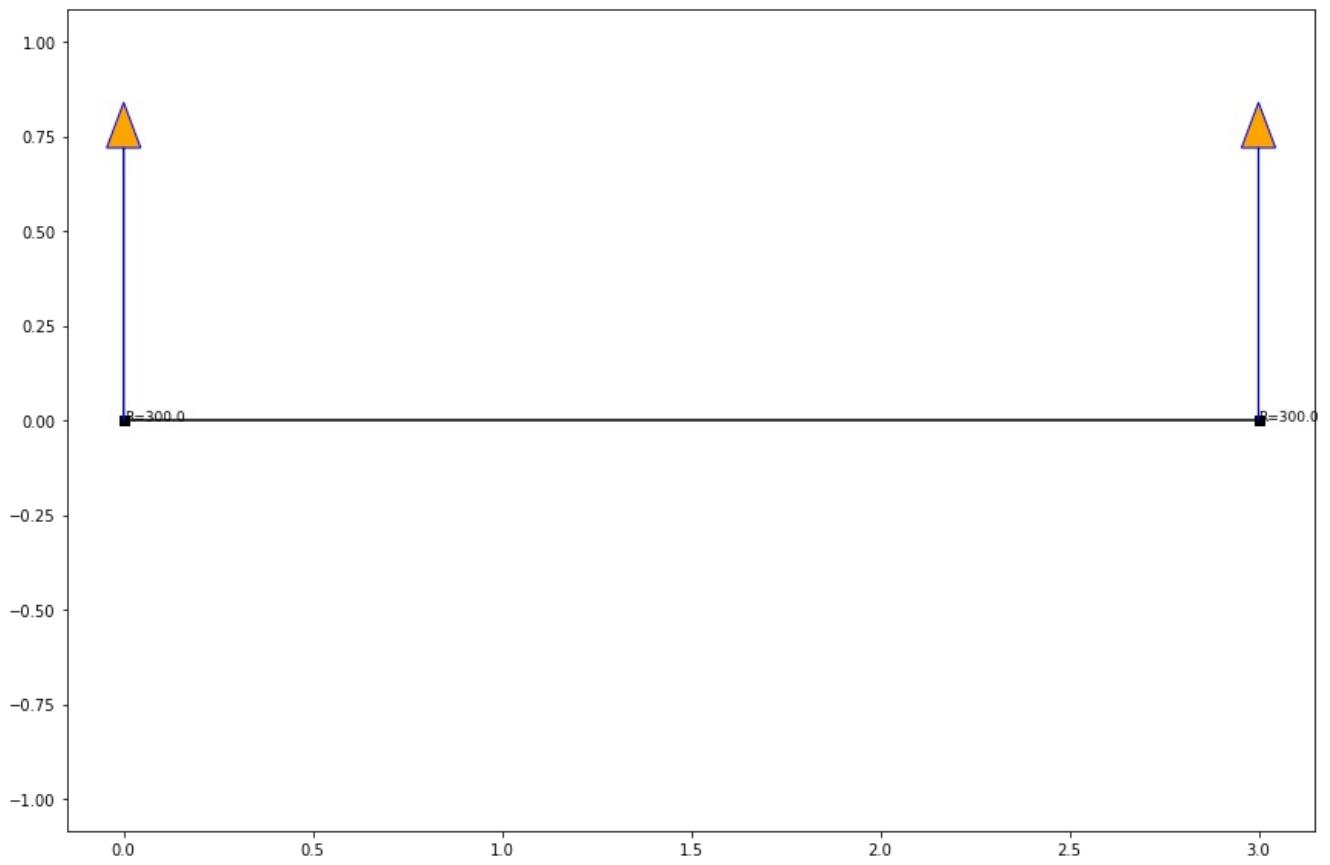
```
In [4]: # Resolvemos la estructura
ss.solve()
```

```
Out[4]: array([-0.00000000e+00, 0.00000000e+00, 4.49999775e-02, -3.67394040e-18,
 0.00000000e+00, -4.49999775e-02])
```

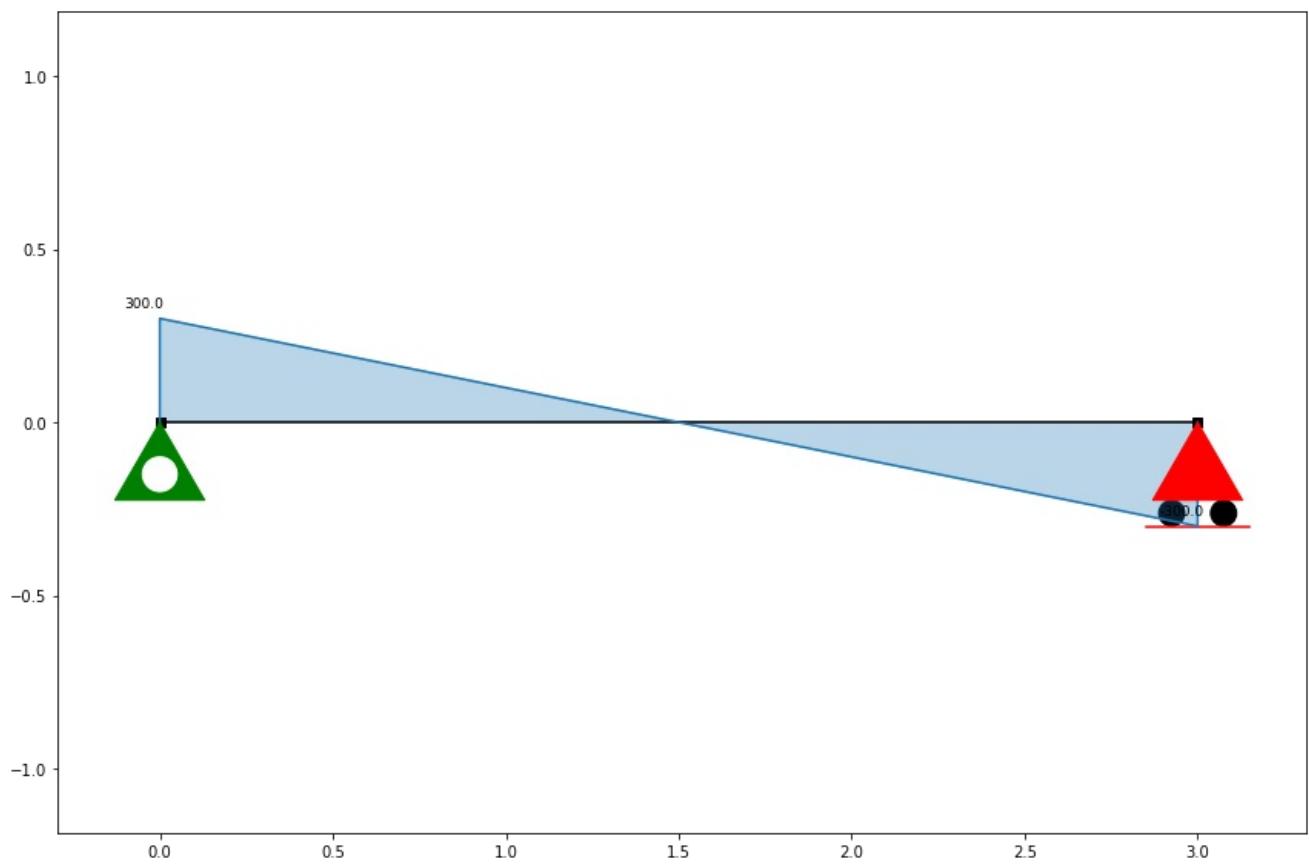
```
In [5]: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

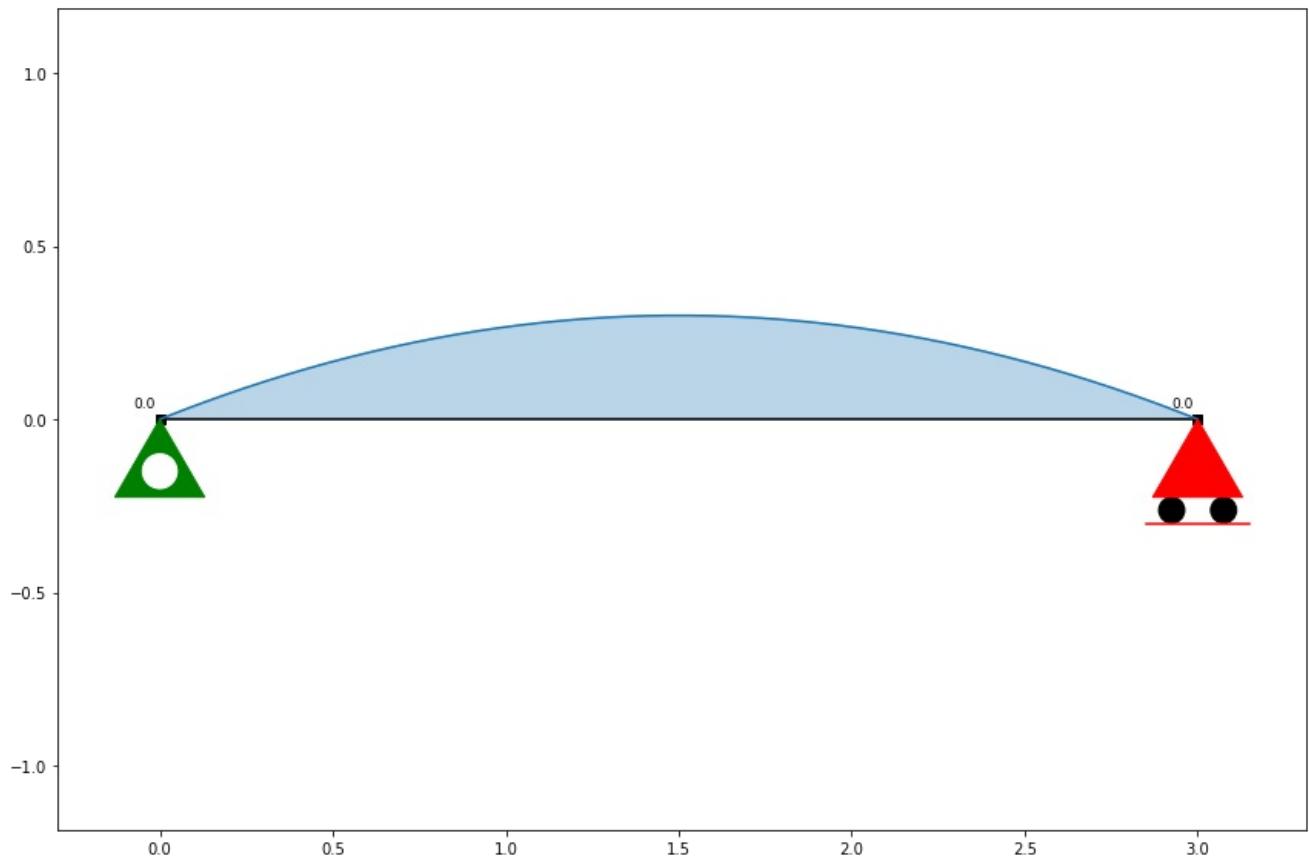
0
*Nodo: 1
Reacción Fy: 300.0000000000006
0
*Nodo: 2
Reacción Fy: 299.9999999999994
```



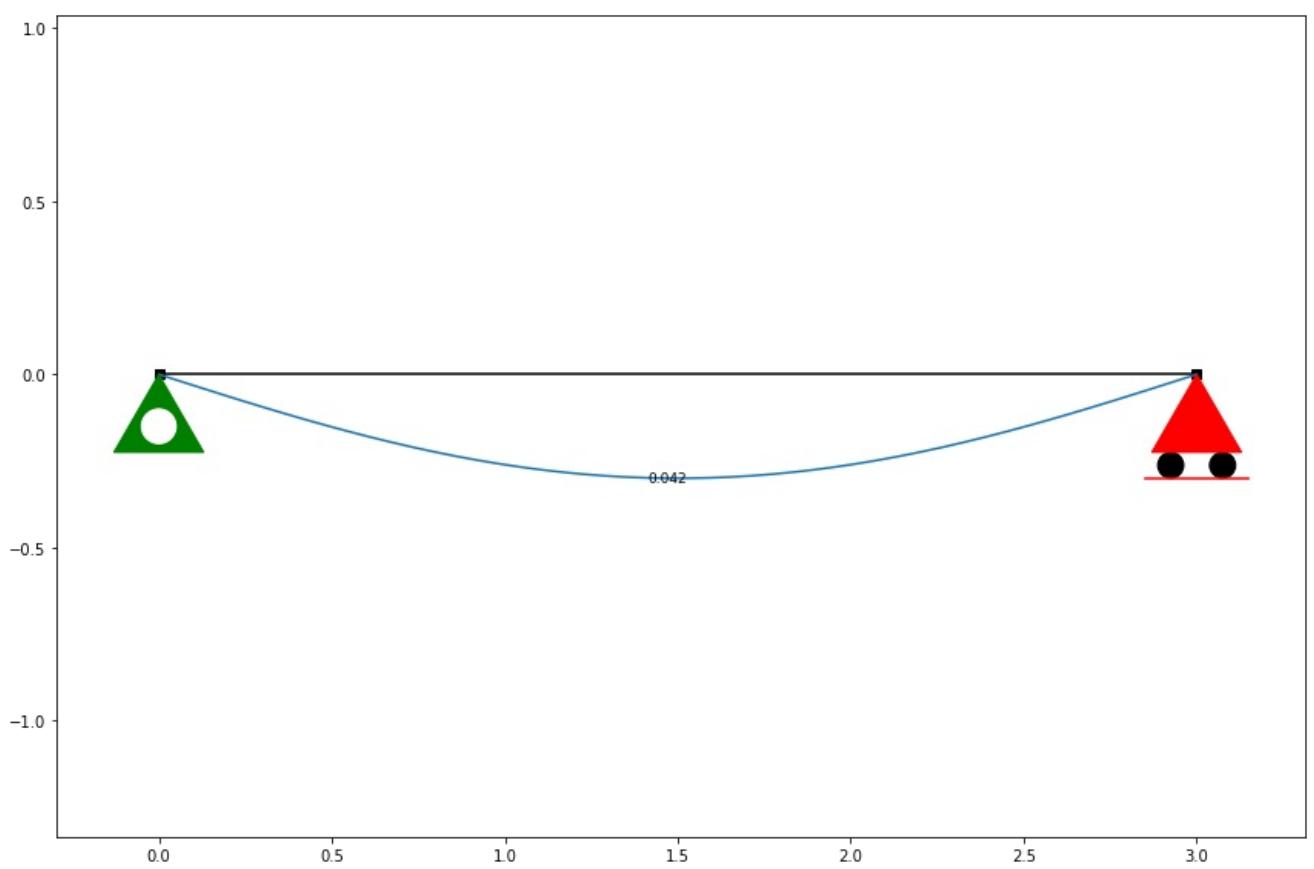
```
In [6]: # Mostramos cortantes
ss.show_shear_force()
```



```
In [7]: # Mostramos flectores
ss.show_bending_moment()
```



```
In [8]: ss.show_displacement()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In []: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-awidc9ix
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-awidc9ix
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.4.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!!=2.1.2,!!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (1.3.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0) (1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=58944 sha256=388ba0f4ecc171a9893b1f2931f5e1ff606534759d95a1a3bd7004a4d647c999
 Stored in directory: /tmp/pip-ephem-wheel-cache-v4loftq8/wheels/23/97/1a/d460d2d29ccd27f0842a8e862290c78834c55be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

```
In []: import numpy as np
from anastruct import SystemElements

ss = SystemElements()

L = 3.0 # Longitud de la barra
q = 200.0 # Carga puntual
ang_giro = 0.

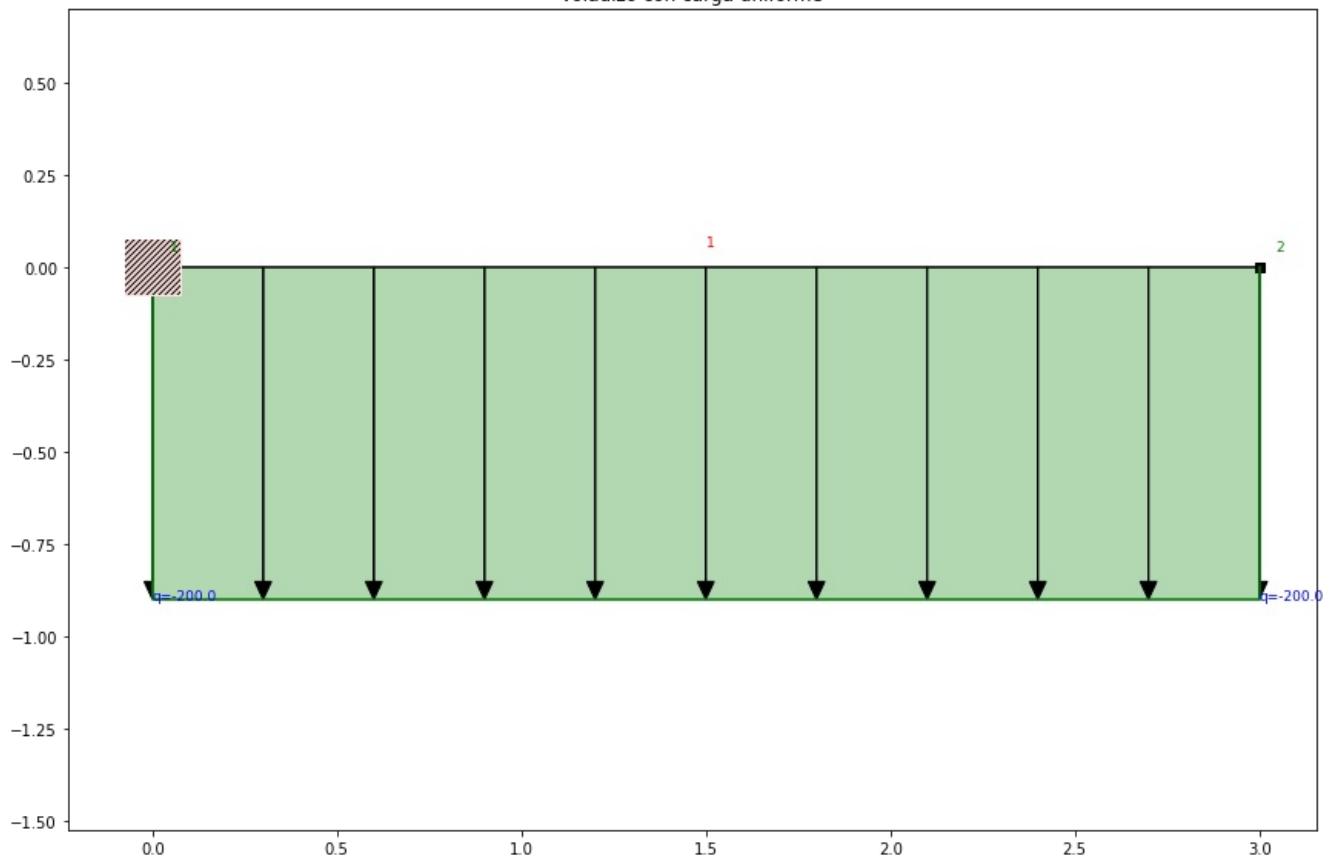
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [L*np.cos(ang_giro*np.pi/180.), L*np.sin(ang_giro*np.pi/180.)]])

Añadimos empotramiento al nodo 1
ss.add_support_fixed(node_id=1)

Añadimos carga uniformemente distribuida
ss.q_load(element_id=1, q=q)

Mostramos estructura generada
ss.show_structure(title='Voladizo con carga uniforme')
```

### Voladizo con carga uniforme



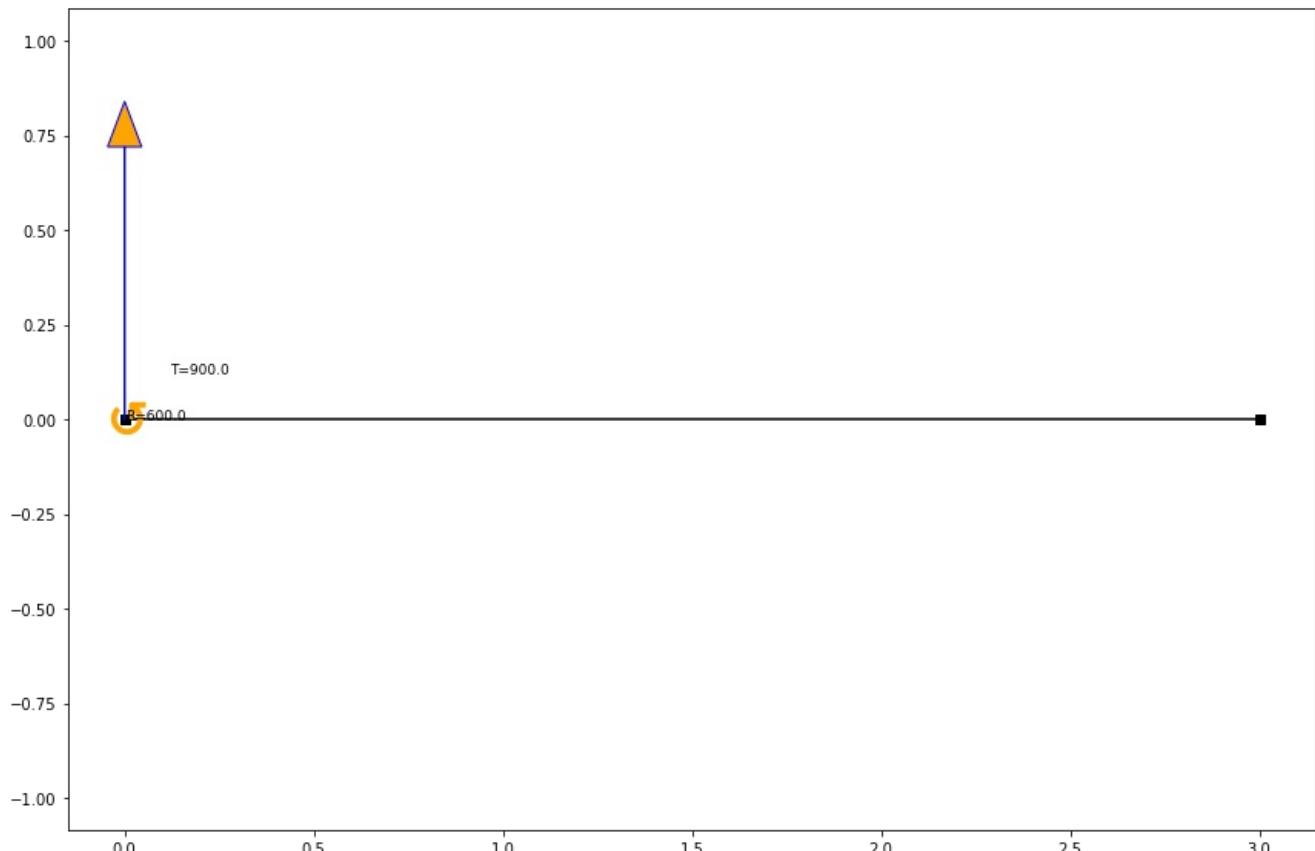
```
In []: # Resolvemos la estructura
ss.solve()
```

```
Out[]: array([-0.00000000e+00, 0.00000000e+00, 0.00000000e+00, -3.67394040e-18,
 -4.05000067e-01, 1.80000045e-01])
```

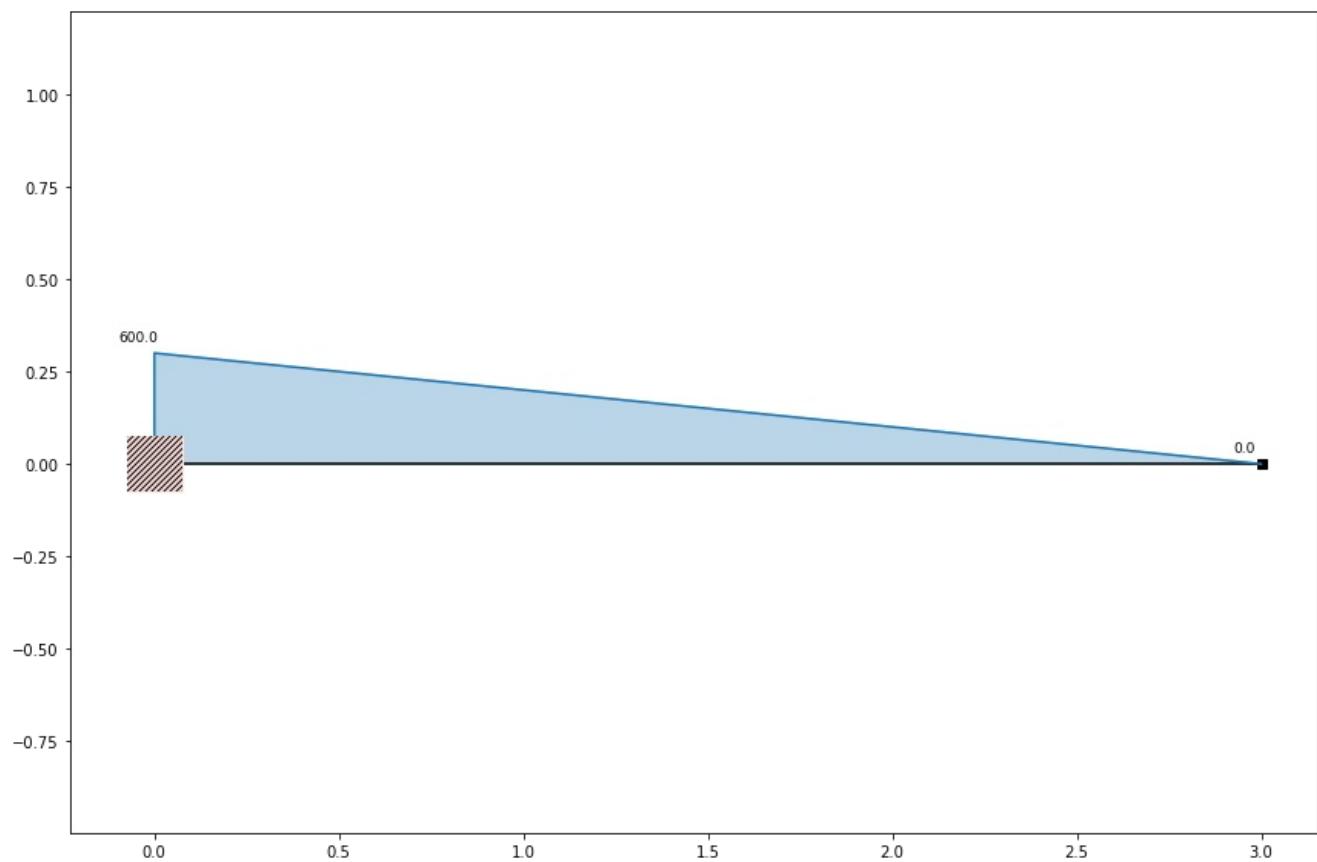
```
In []: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

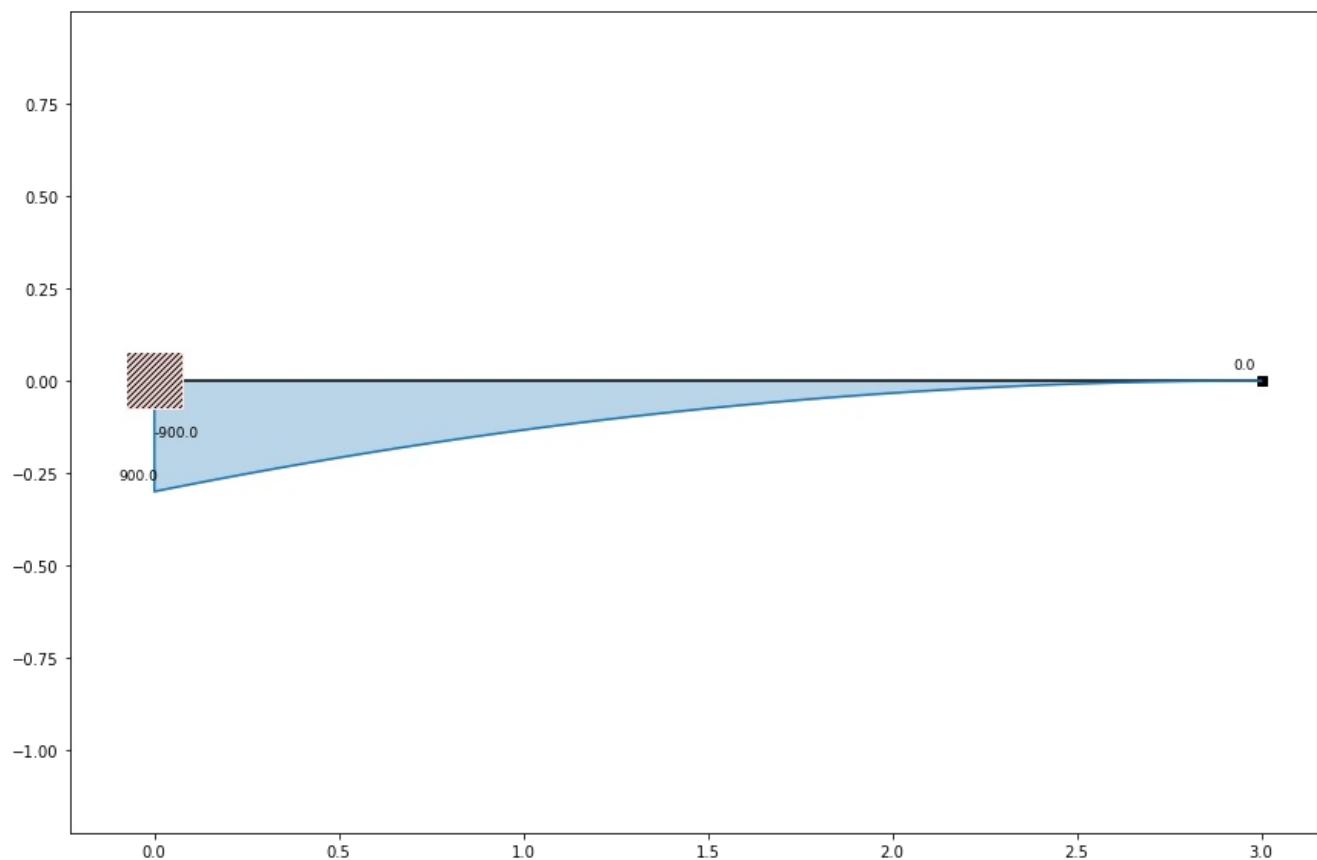
0
*Nodo: 1
Reacción Fy: 600.0
*Nodo: 1
Momento Mz: 900.000000000001
```



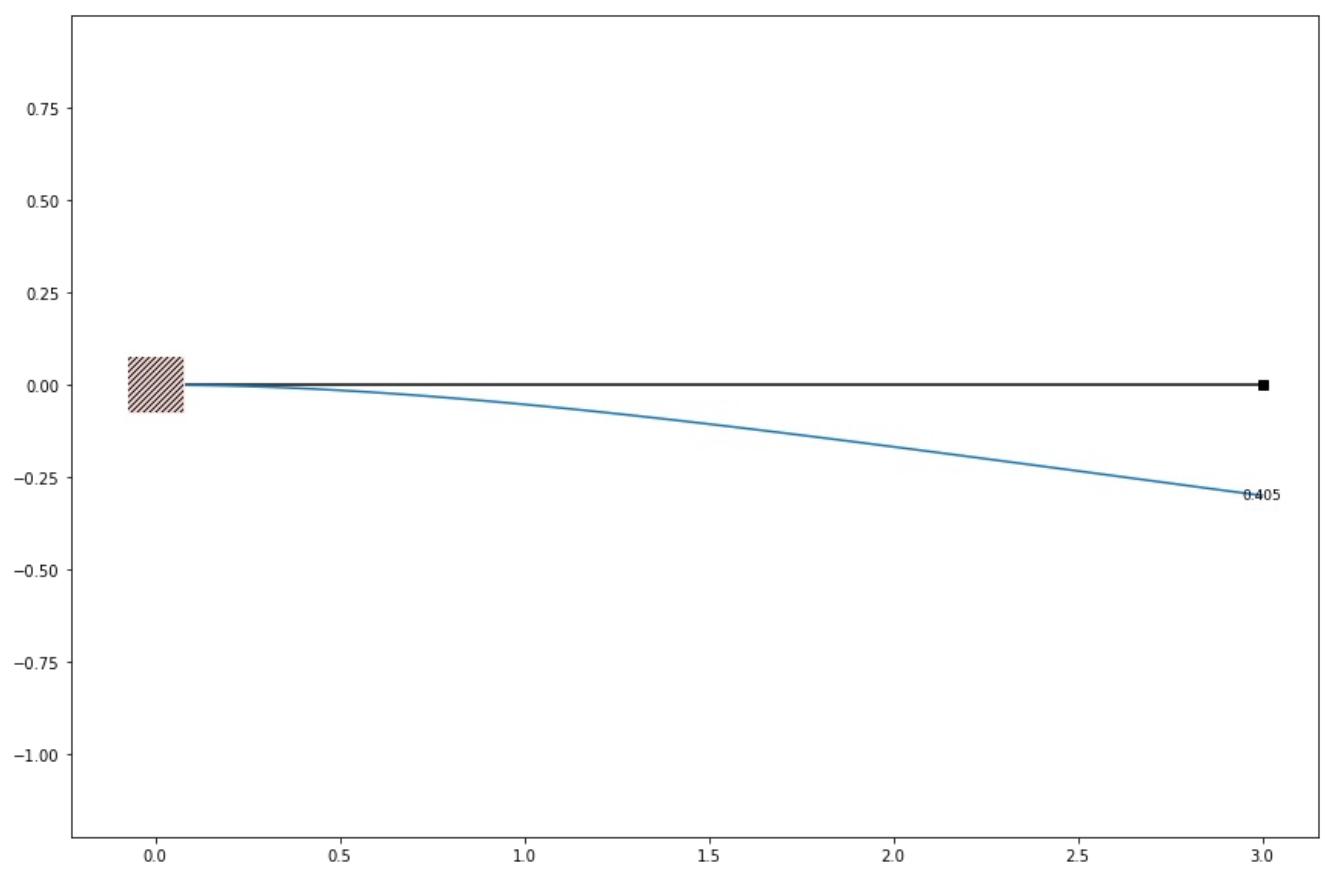
```
In []: # Mostramos cortantes
ss.show_shear_force()
```



```
In []: # Mostramos flectores
ss.show_bending_moment()
```



```
In []: ss.show_displacement()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In [1]: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-ntad08jt
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-ntad08jt
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.4.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.4.7)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0) (1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=58944 sha256=daaf0aad9d621b801c6547c06f5b722be5c279abe5d7d696ed43aa9e15da94bd
 Stored in directory: /tmp/pip-ephem-wheel-cache-eqy2lx8s/wheels/23/97/1a/d460d2d29cccd27f0842a8e862290c78834c55be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

```
In [2]: import numpy as np
from anastruct import SystemElements

ss = SystemElements()

L = 3.0 # Longitud de la barra
q = 200.0 # Carga puntual
ang_giro = 0.

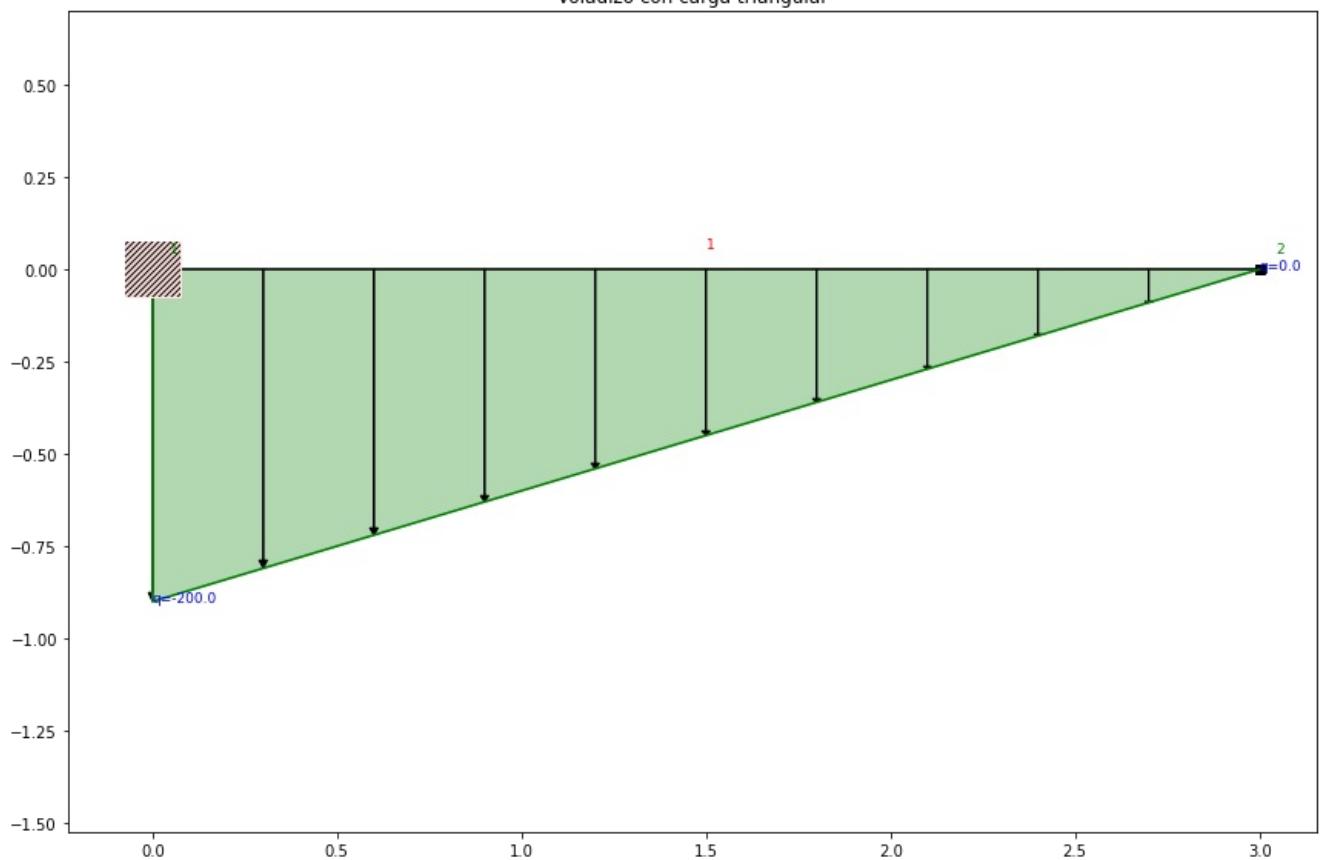
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [L*np.cos(ang_giro*np.pi/180.), L*np.sin(ang_giro*np.pi/180.)]])

Añadimos empotramiento al nodo 1
ss.add_support_fixed(node_id=1)

Añadimos carga puntual al nodo 2
ss.q_load(element_id=1, q=(q,0), q_perp=(0,0))

Mostramos estructura generada
ss.show_structure(title='Voladizo con carga triangular')
```

### Voladizo con carga triangular



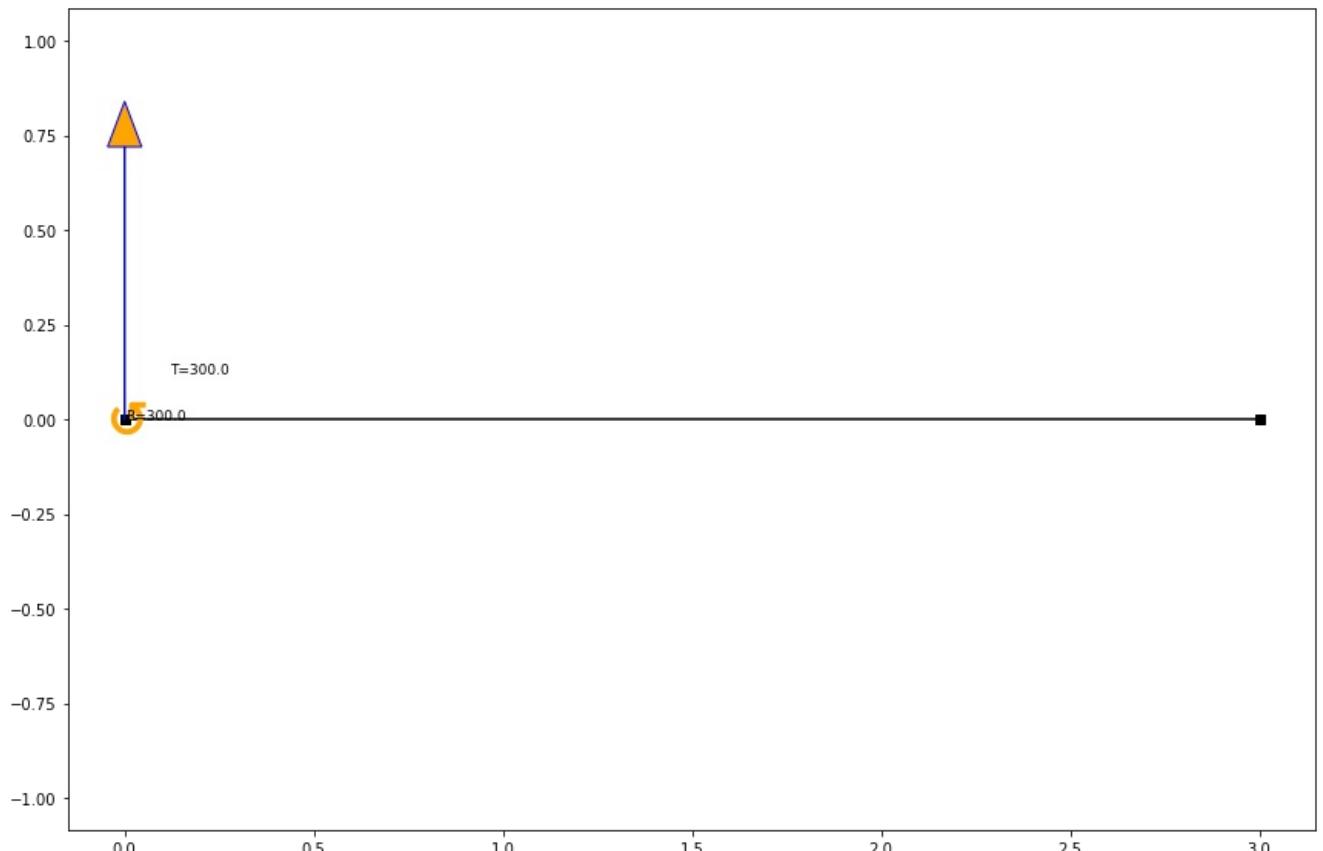
```
In [3]: # Resolvemos la estructura
ss.solve()
```

```
Out[3]: array([-0.00000000e+00, 0.00000000e+00, 0.00000000e+00, -1.22464680e-18,
 -1.08000040e-01, 4.50000225e-02])
```

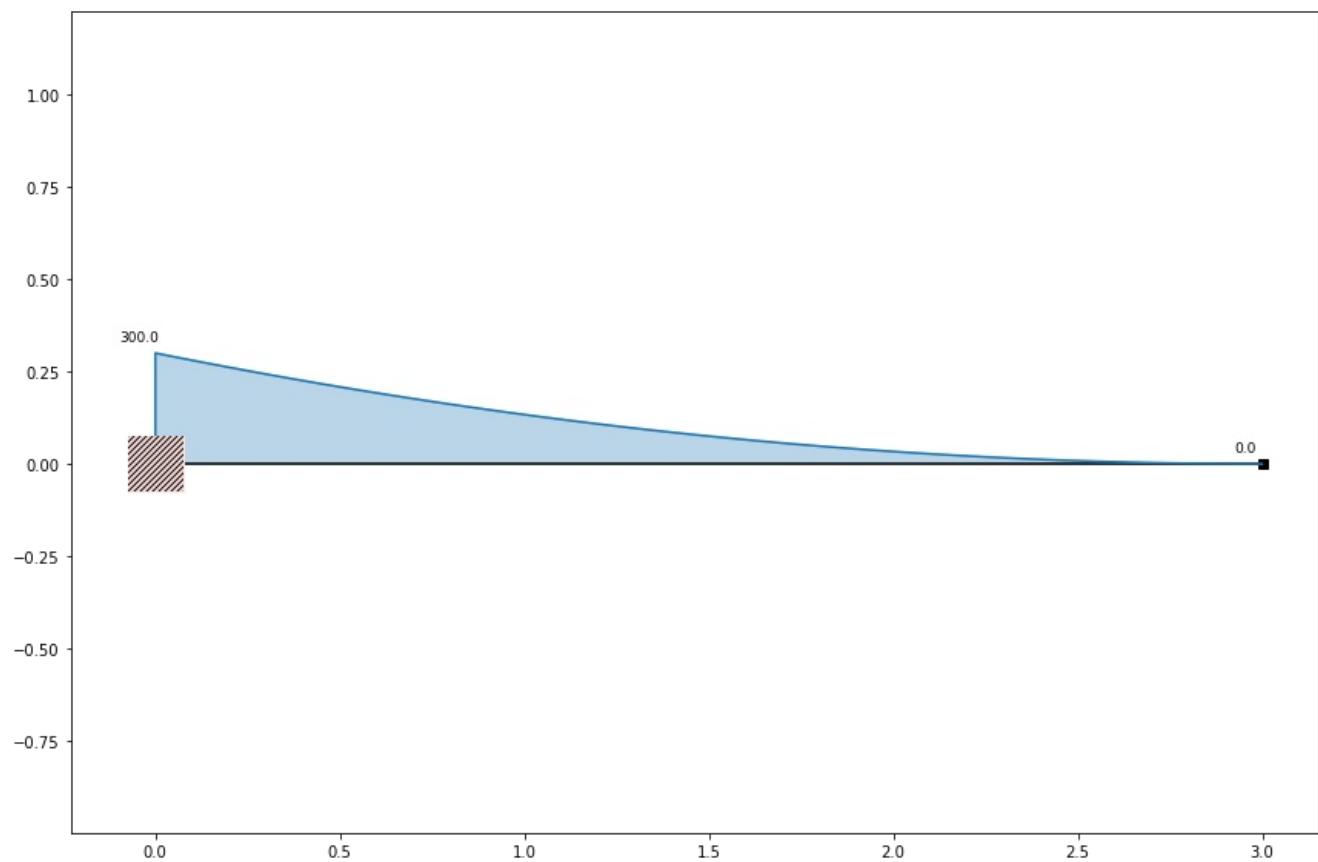
```
In [4]: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

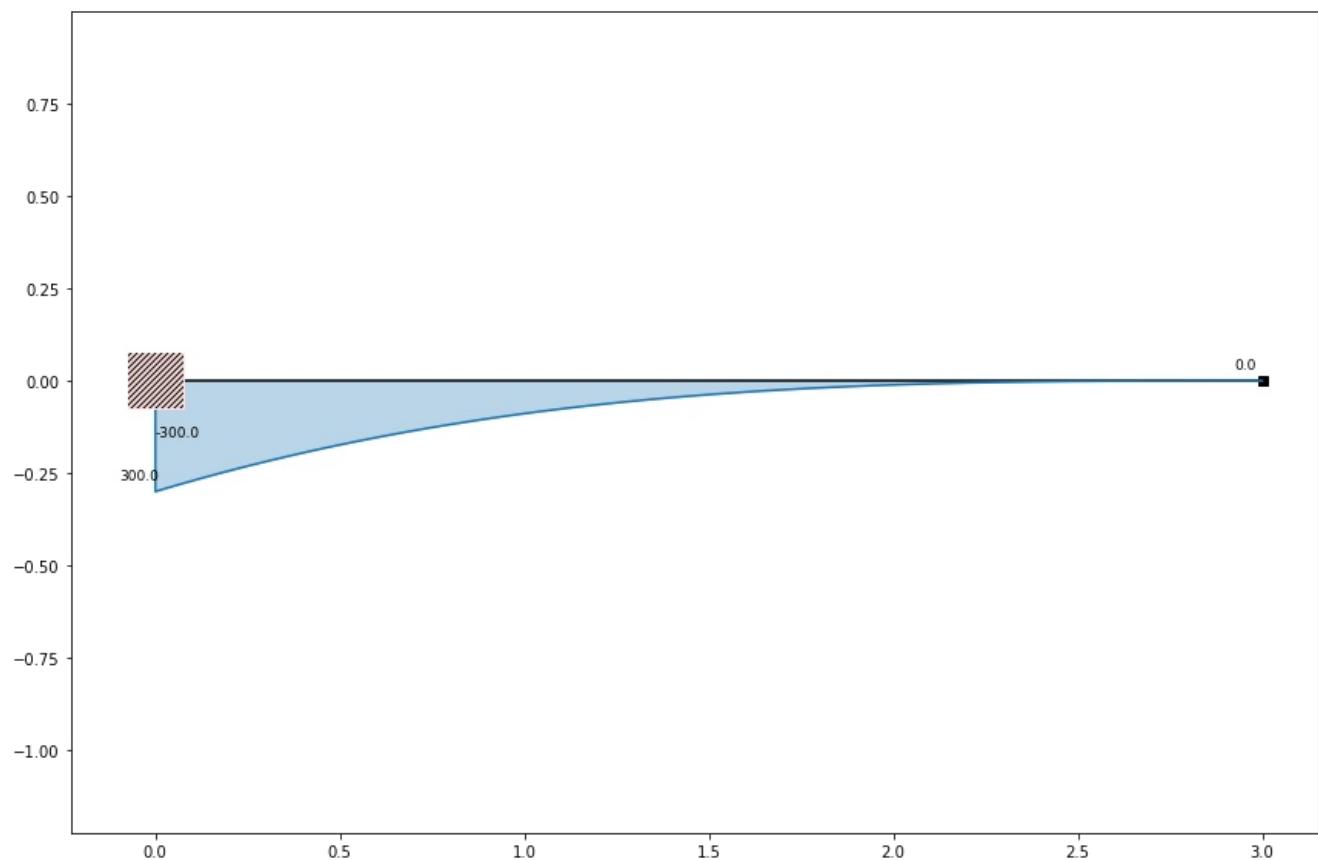
0
*Nodo: 1
Reacción Fy: 300.0000000000006
*Nodo: 1
Momento Mz: 300.0000000000002
```



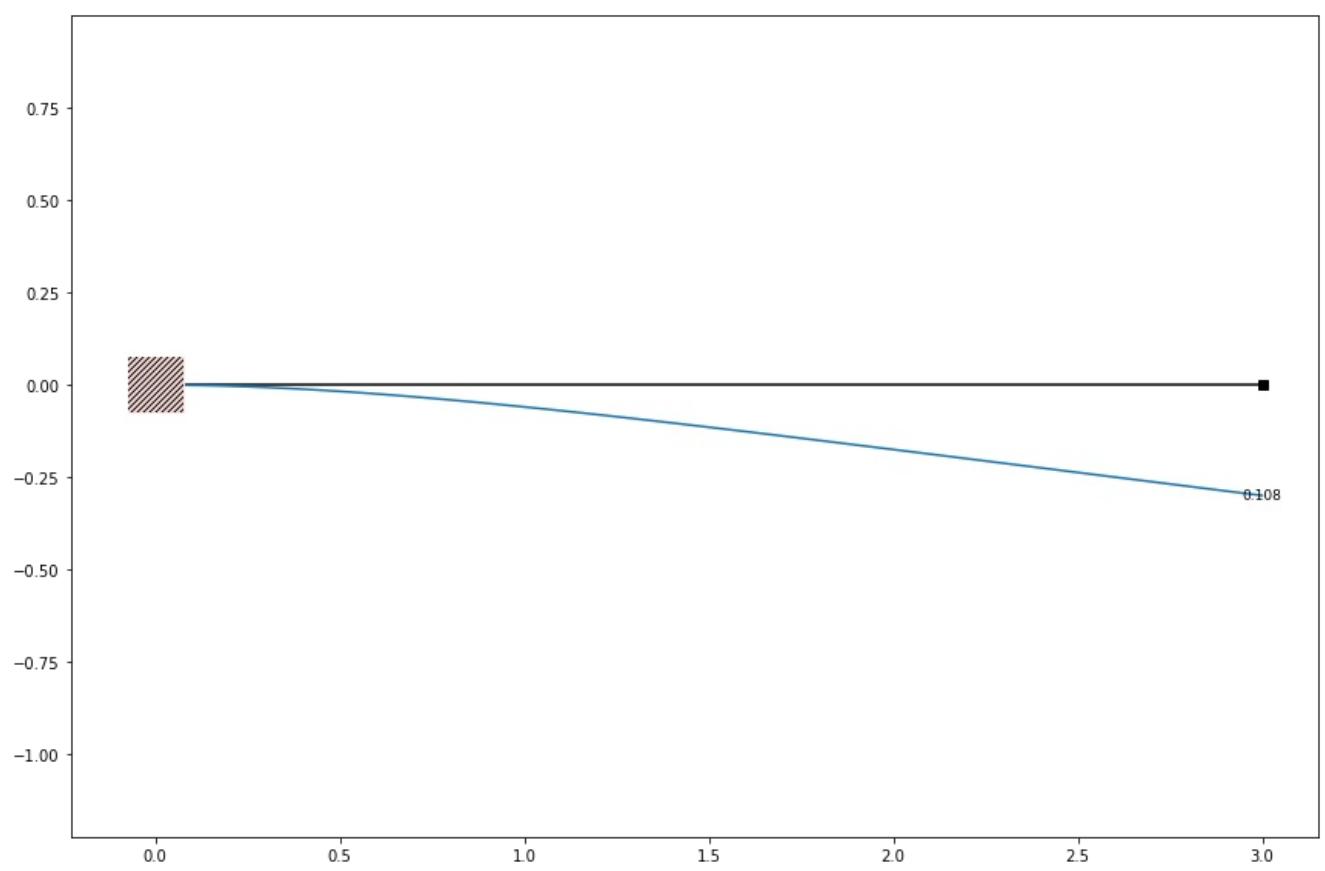
```
In [5]: # Mostramos cortantes
ss.show_shear_force()
```



```
In [6]: # Mostramos flectores
ss.show_bending_moment()
```



```
In [7]: ss.show_displacement()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In [1]: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-r72n30h4
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-r72n30h4
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.4.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (1.3.2)
Requirement already satisfied: pyparsing!=2.0.4,!!=2.1.2,!!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.4.7)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0) (1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=58944 sha256=58b69e9eba6fa345d247a4fcfaaa6fa70d8f973e423c201d2940caffc0d33fd7
 Stored in directory: /tmp/pip-ephem-wheel-cache-9wkjf5cb/wheels/23/97/1a/d460d2d29ccd27f0842a8e862290c78834c55be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

```
In [3]: from anastruct import SystemElements
ss = SystemElements()

L = 3.0 # Longitud de la barra
P = -100.0 # Carga puntual

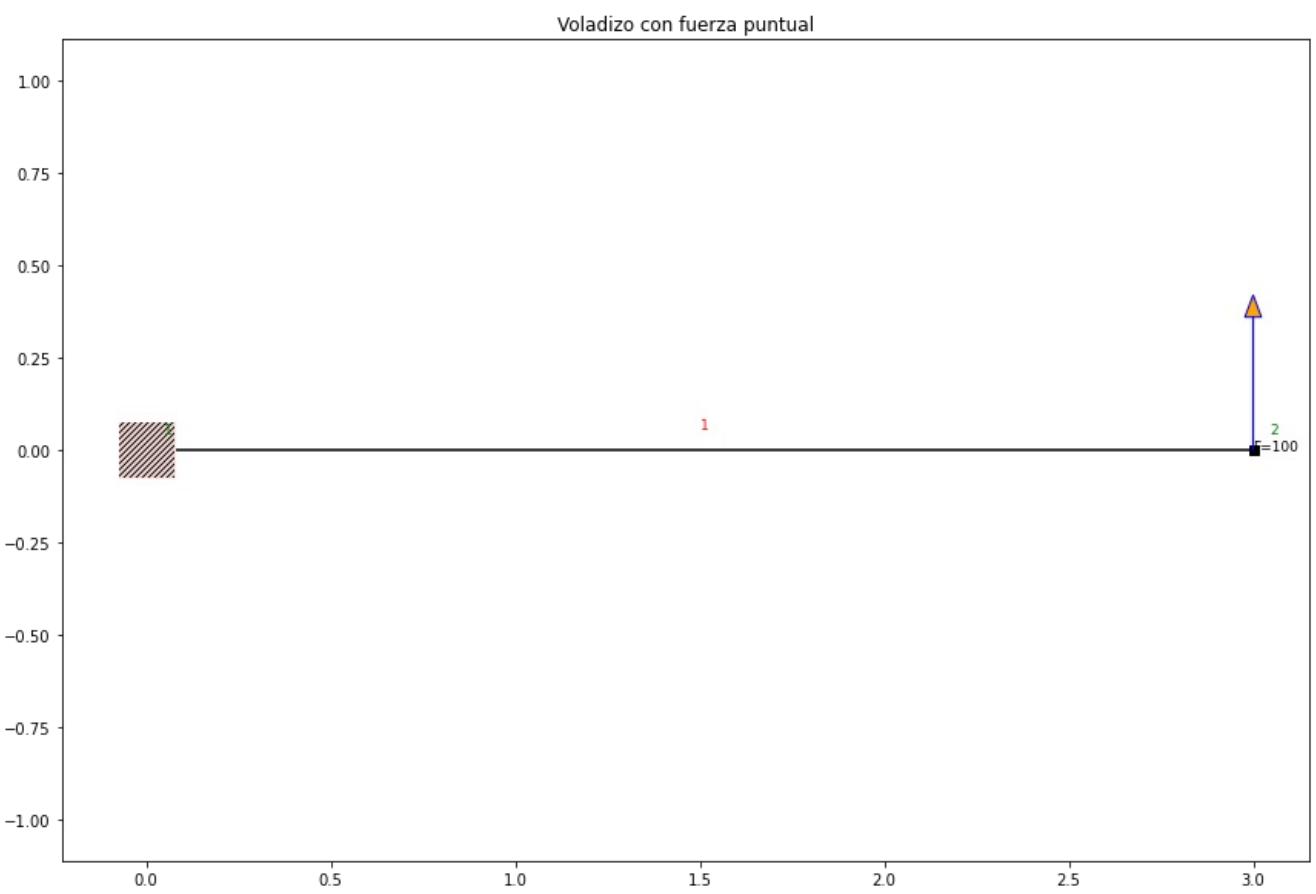
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [L, 0]])

Añadimos empotramiento al nodo 1
ss.add_support_fixed(node_id=1)

Añadimos carga puntual al nodo 2
ss.point_load(2, Fx=0, Fy=-P)

Mostramos estructura generada
ss.show_structure(title='Voladizo con fuerza puntual')
```

-0.0



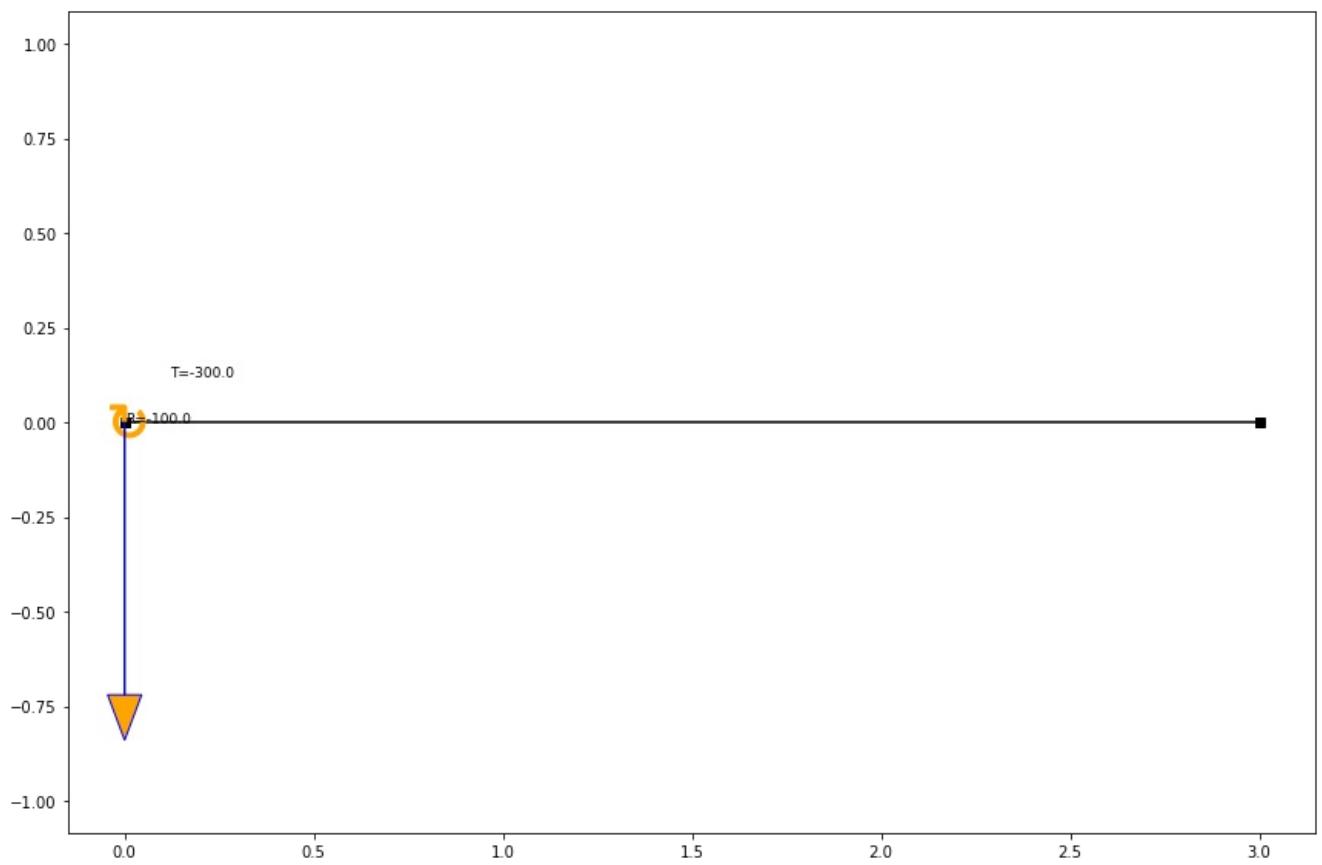
```
In [4]: # Resolvemos la estructura
ss.solve()
```

```
Out[4]: array([-0. , 0. , 0. , -0. , 0.18, -0.09])
```

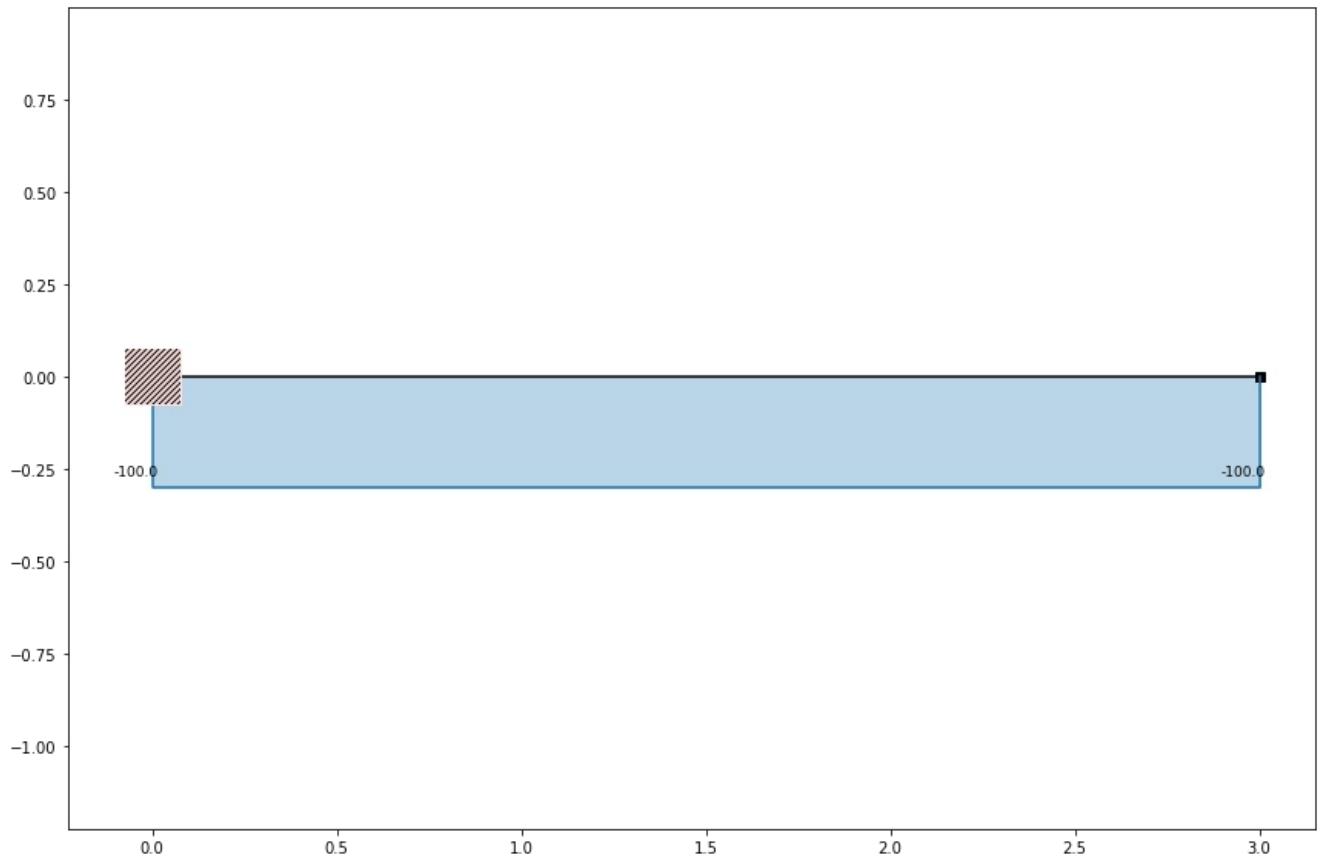
```
In [5]: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

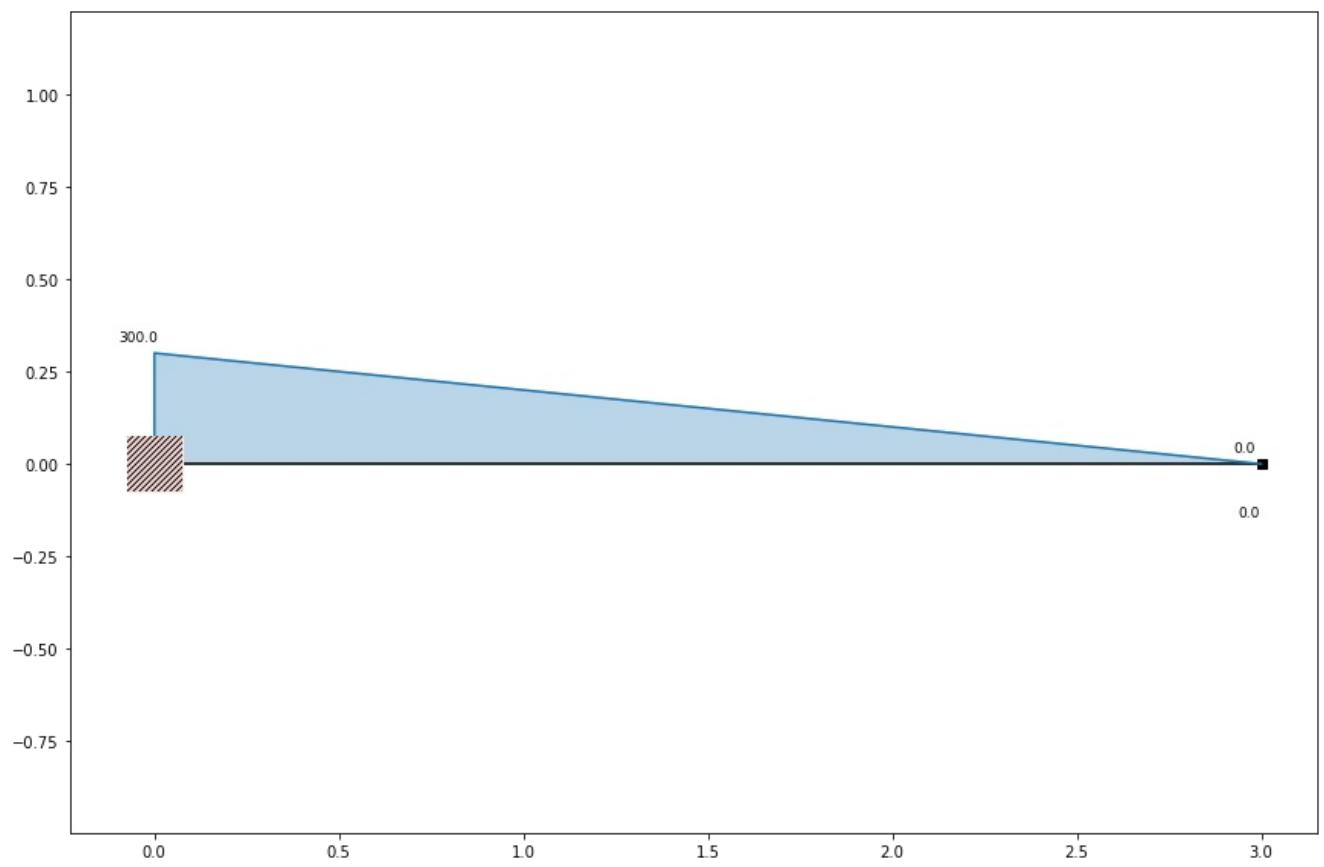
0
*Nodo: 1
Reacción Fy: -99.9999999999999
*Nodo: 1
Momento Mz: -300.0000000000006
```



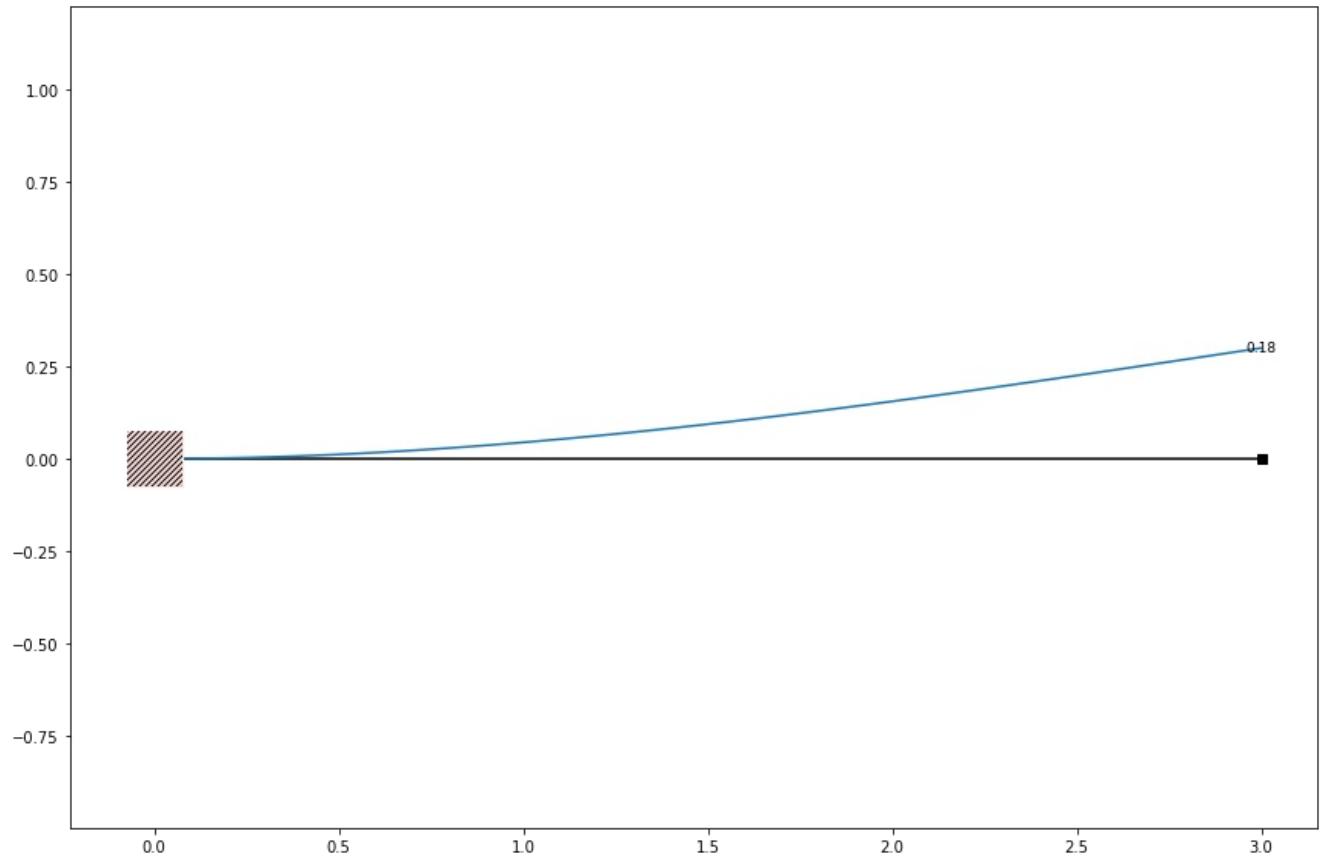
```
In [6]: # Mostramos cortantes
ss.show_shear_force()
```



```
In [7]: # Mostramos flectores
ss.show_bending_moment()
```



```
In [8]: # Mostramos deformada
ss.show_displacement()
```



```
In []:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In [1]: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-h76hh1y2
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-h76hh1y2
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.4.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (1.3.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (0.10.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0) (1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=58944 sha256=24b8eebcf2c76cec7305c94ca54a81fb4fdd214b9994e92d7cff7c9124217ac5
 Stored in directory: /tmp/pip-ephem-wheel-cache-bb67z1vn/wheels/23/97/1a/d460d2d29ccd27f0842a8e862290c78834c55be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

```
In [2]: from anastruct import SystemElements

ss = SystemElements()

L = 3.0 # Longitud de la barra
P = -100.0 # Carga puntual
a = 2.5 # Distancia a la que se encuentra la carga aplicada

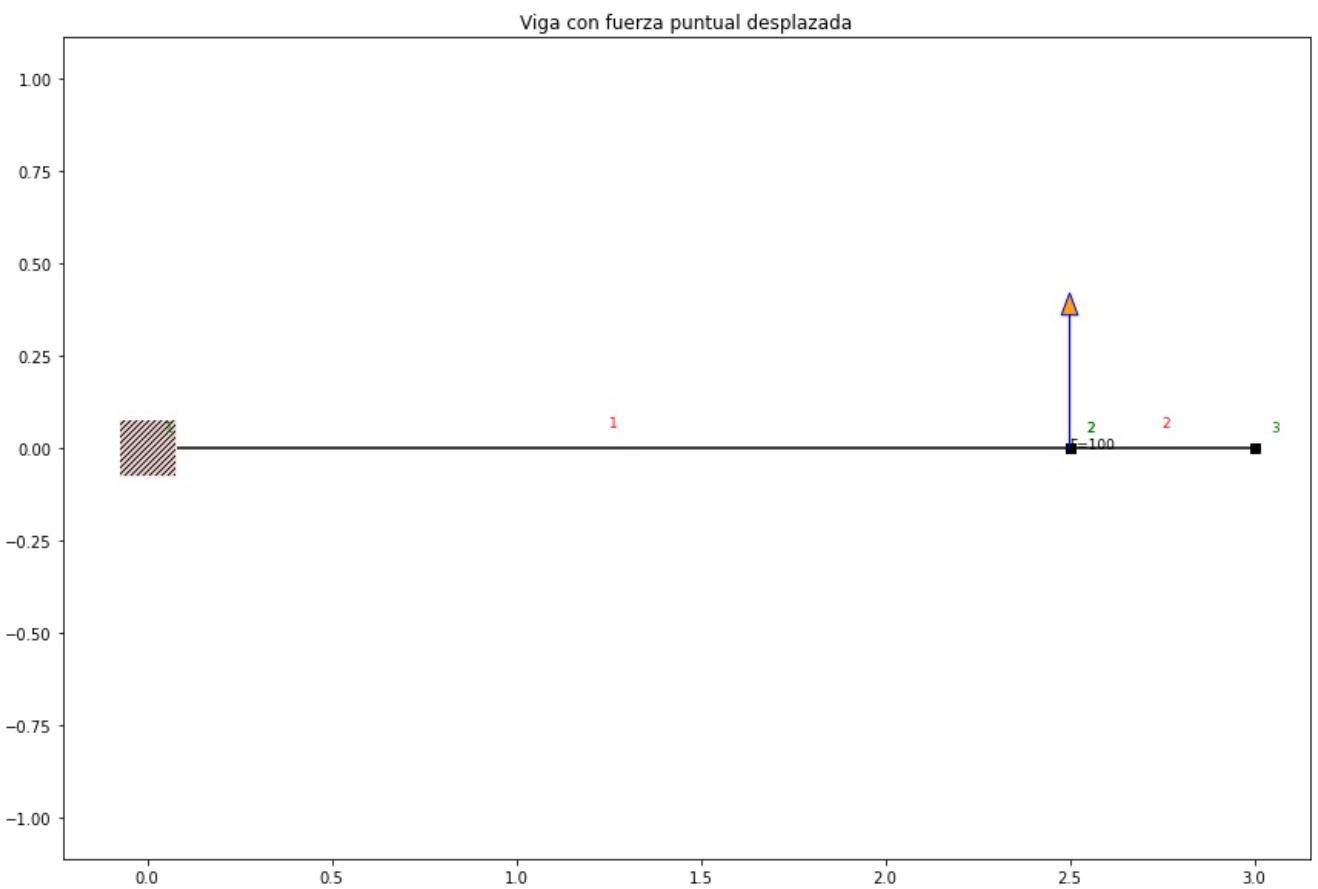
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [a, 0]])
Añadimos elemento barra 2
ss.add_element(location=[[a, 0], [L, 0]])

Añadimos empotramiento al nodo 1
ss.add_support_fixed(node_id=1)

Añadimos carga puntual al nodo 2
ss.point_load(2, Fx=0, Fy=-P)

Mostramos estructura generada
ss.show_structure(title='Viga con fuerza puntual desplazada')

-0.0
```



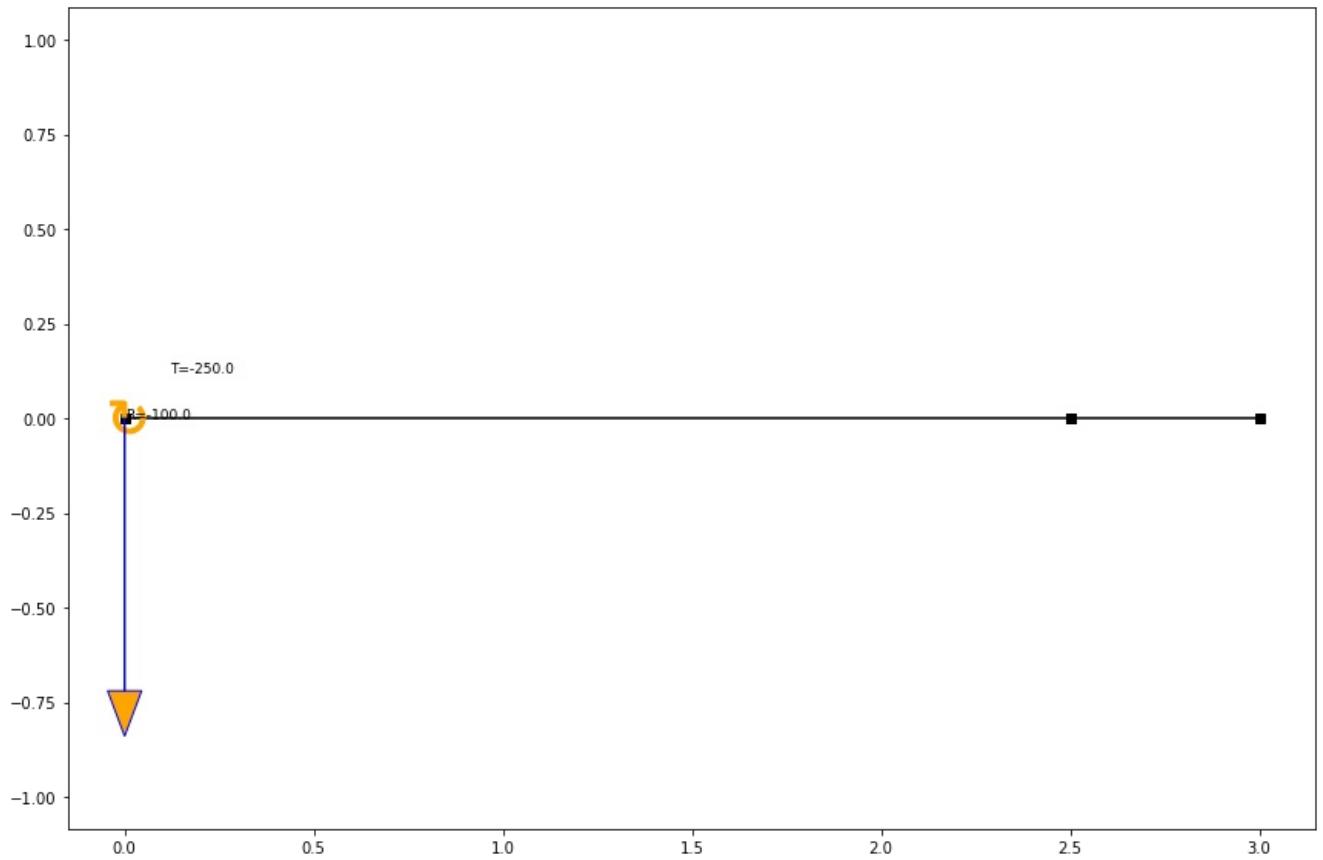
```
In [3]: # Resolvemos la estructura
ss.solve()
```

```
Out[3]: array([-0. , 0. , 0. , -0. ,
 -0.0625 , -0. , 0.13541667, -0.0625])
```

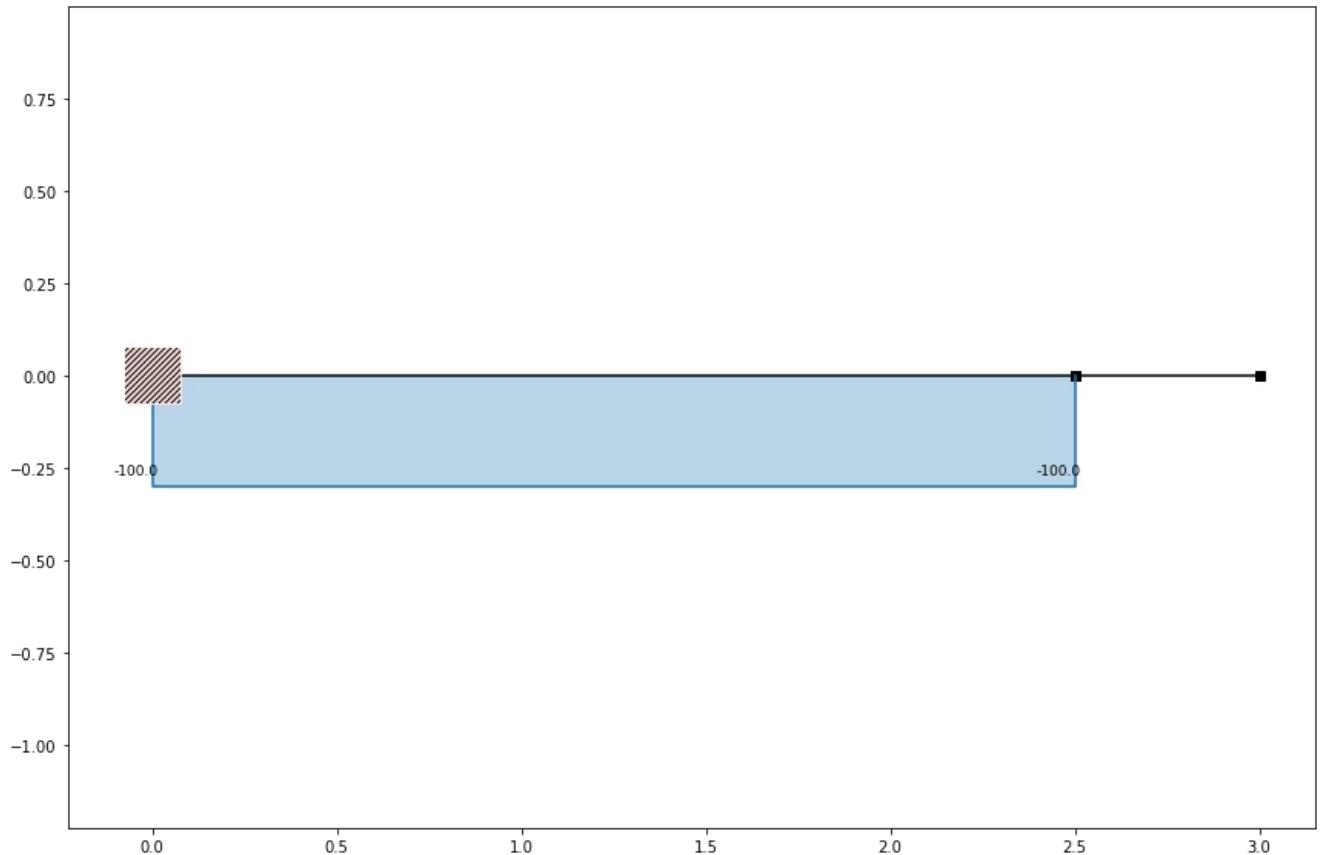
```
In [4]: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

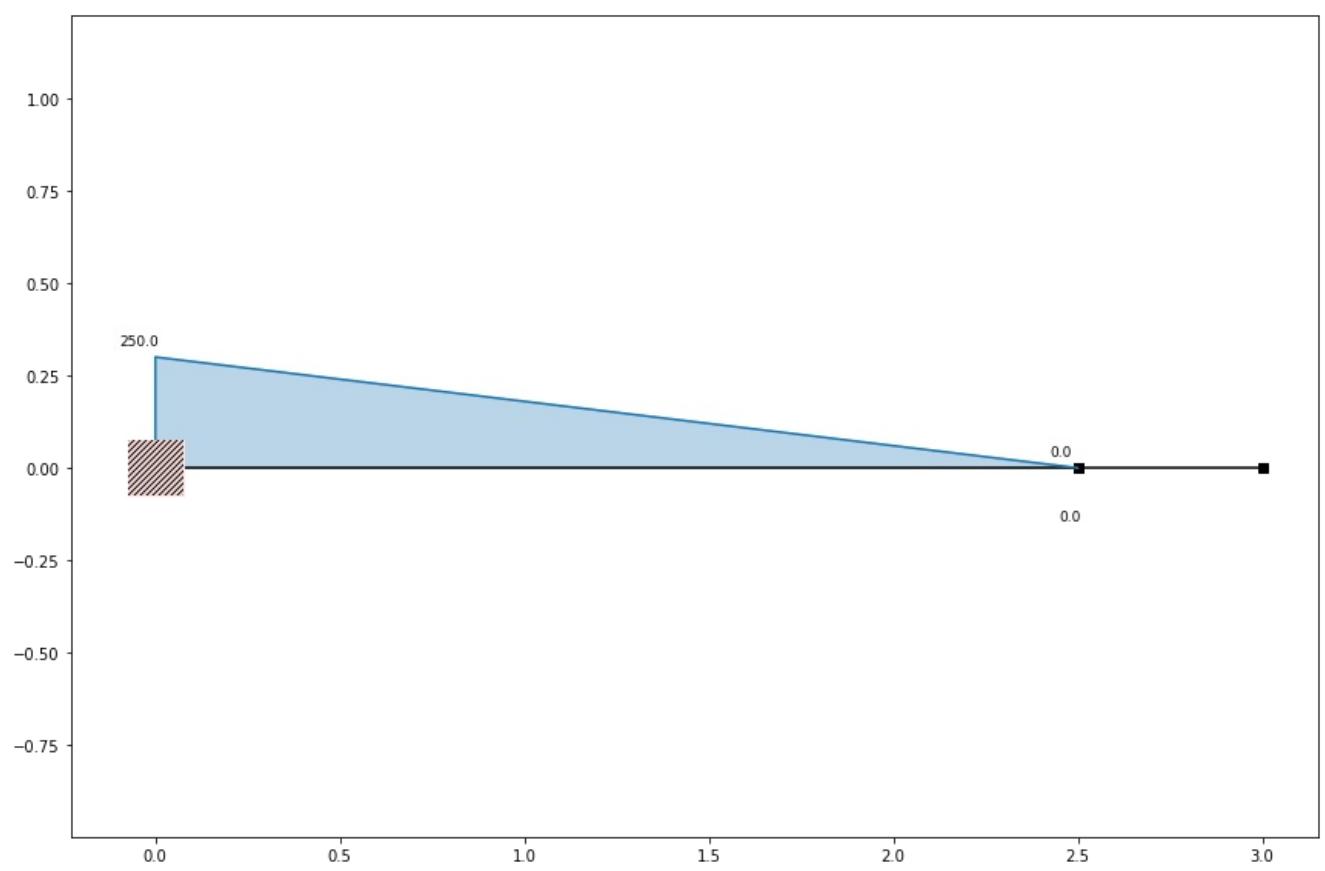
0
*Nodo: 1
Reacción Fy: -100.000000000046
*Nodo: 1
Momento Mz: -250.0000000001288
```



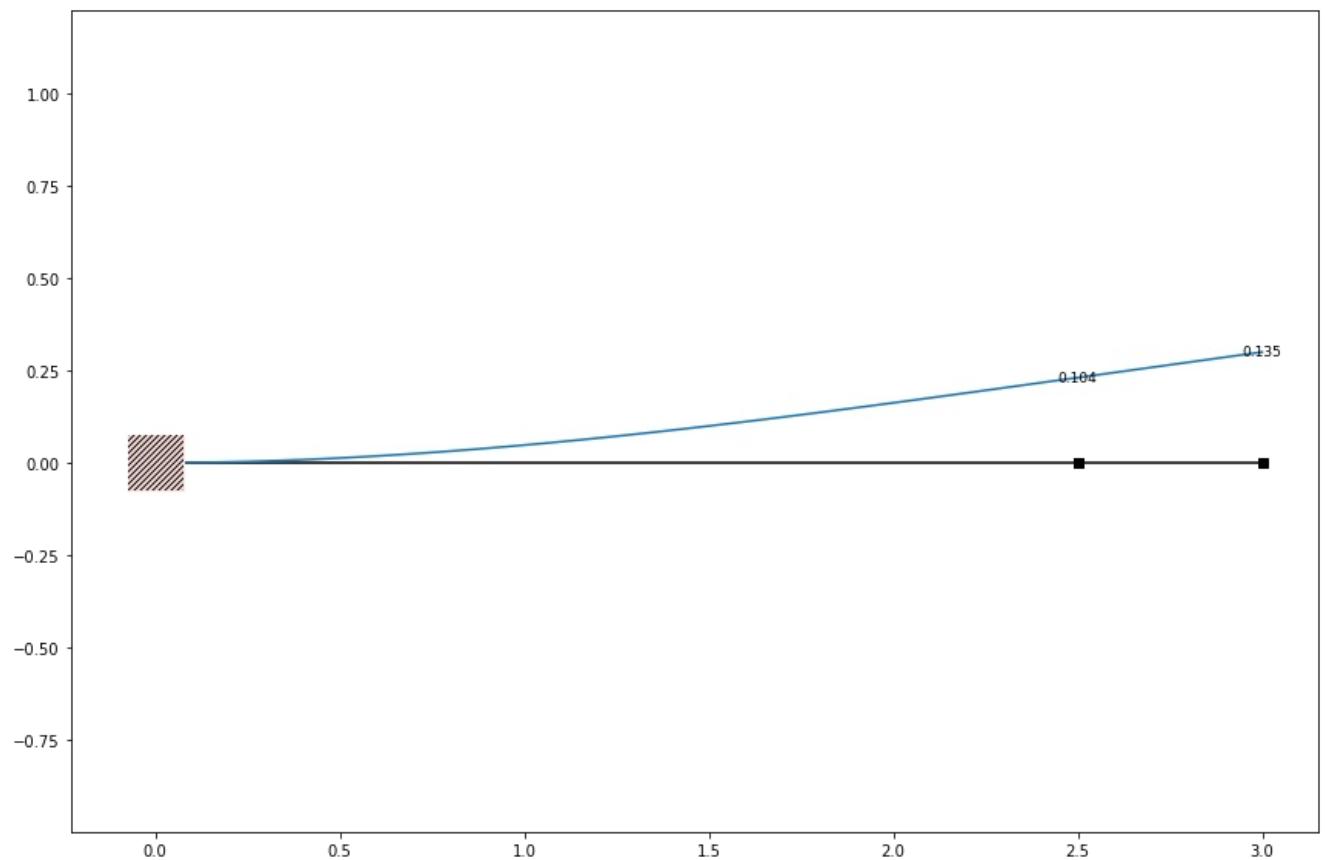
```
In [5]: # Mostramos cortantes
ss.show_shear_force()
```



```
In [6]: # Mostramos flectores
ss.show_bending_moment()
```



In [7]: `ss.show_displacement()`



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In [1]: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-ftizzmiq
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-ftizzmiq
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0) (1.4.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (0.10.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0) (1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=58944 sha256=7c9cd6e5e68df4970261b683e98a12ff781c74f5644977827db9dfac1ae47947
 Stored in directory: /tmp/pip-ephem-wheel-cache-dddfb6ou/wheels/23/97/1a/d460d2d29ccd27f0842a8e862290c78834c55be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

```
In [2]: import numpy as np
from anastruct import SystemElements

ss = SystemElements()

L = 3.0 # Longitud de la barra
M = -20.0 # Momento puntual

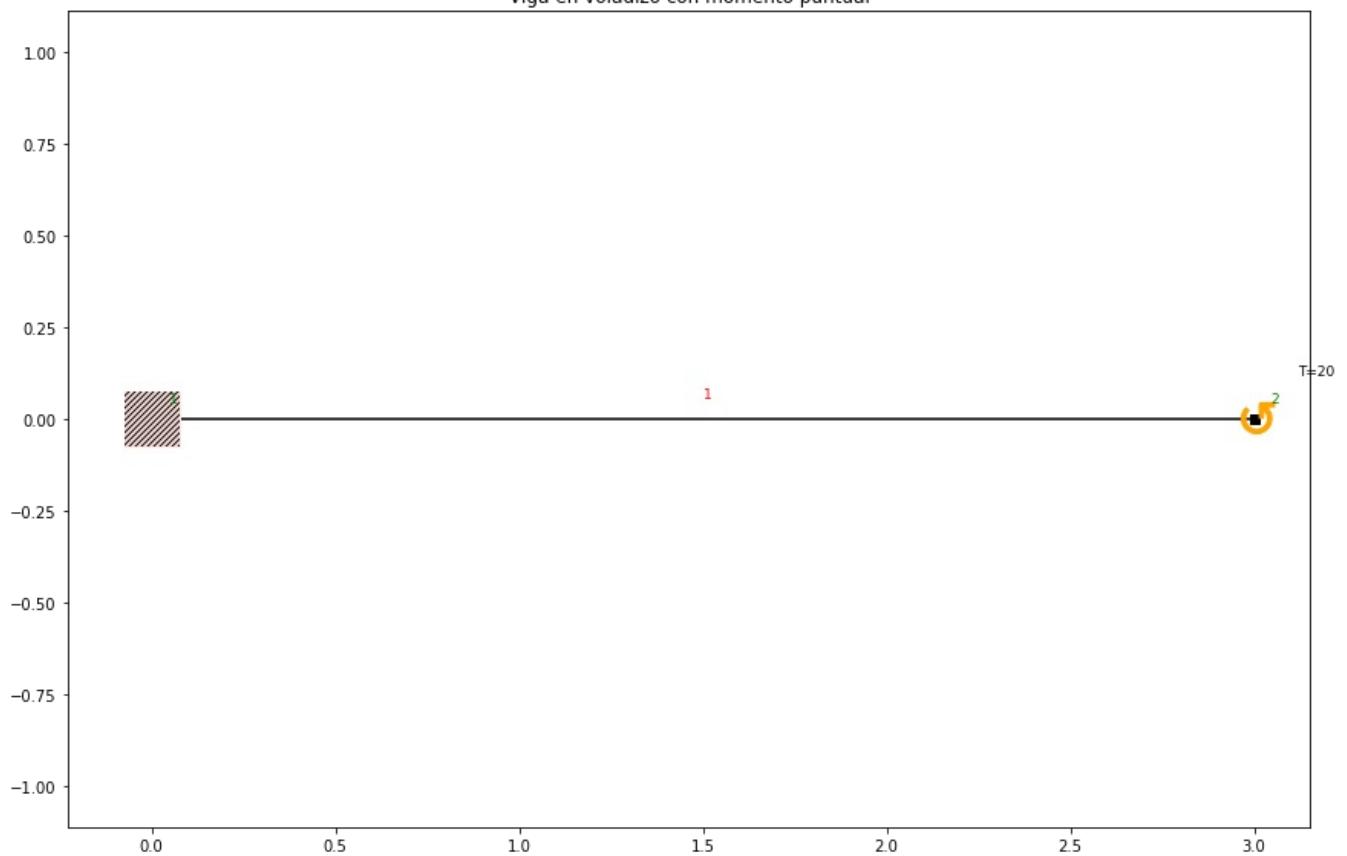
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [L, 0]])

Añadimos empotramiento al nodo 1
ss.add_support_fixed(node_id=1)

Añadimos momento puntual al nodo 2
ss.moment_load(2, Ty=M)

Mostramos estructura generada
ss.show_structure(title='Viga en voladizo con momento puntual')
```

Viga en voladizo con momento puntual



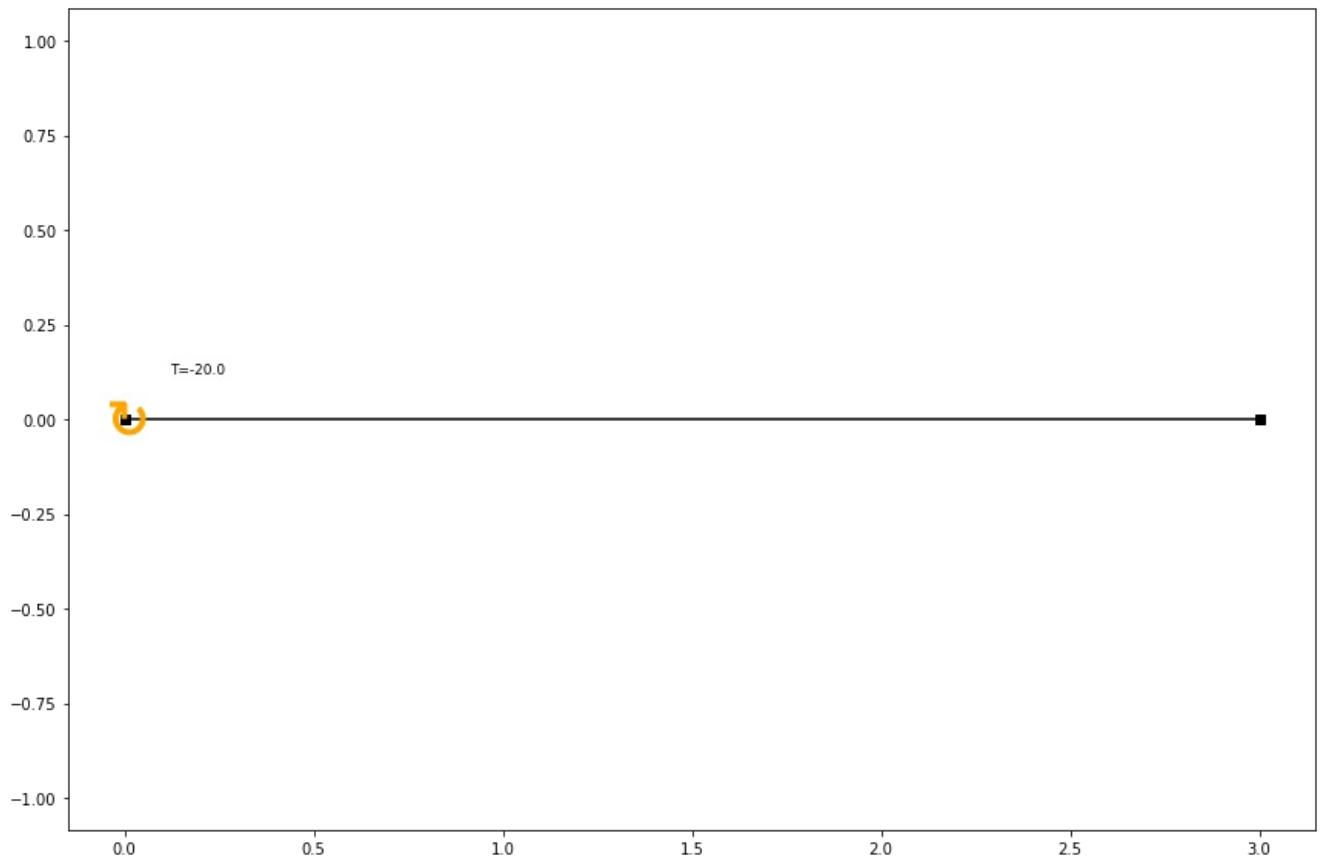
```
In [3]: # Resolvemos la estructura
ss.solve()
```

```
Out[3]: array([-0. , 0. , 0. , -0. , 0.018, -0.012])
```

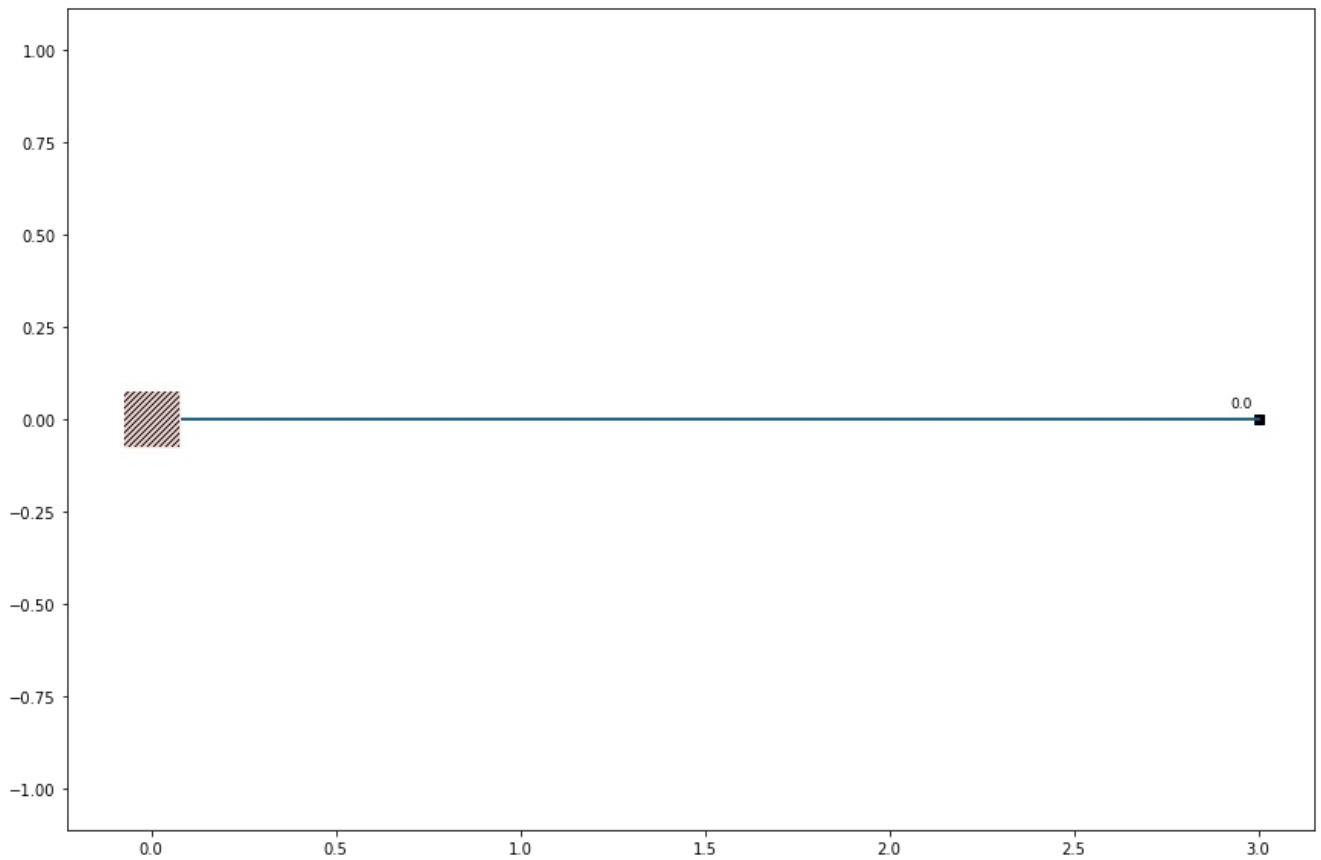
```
In [4]: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

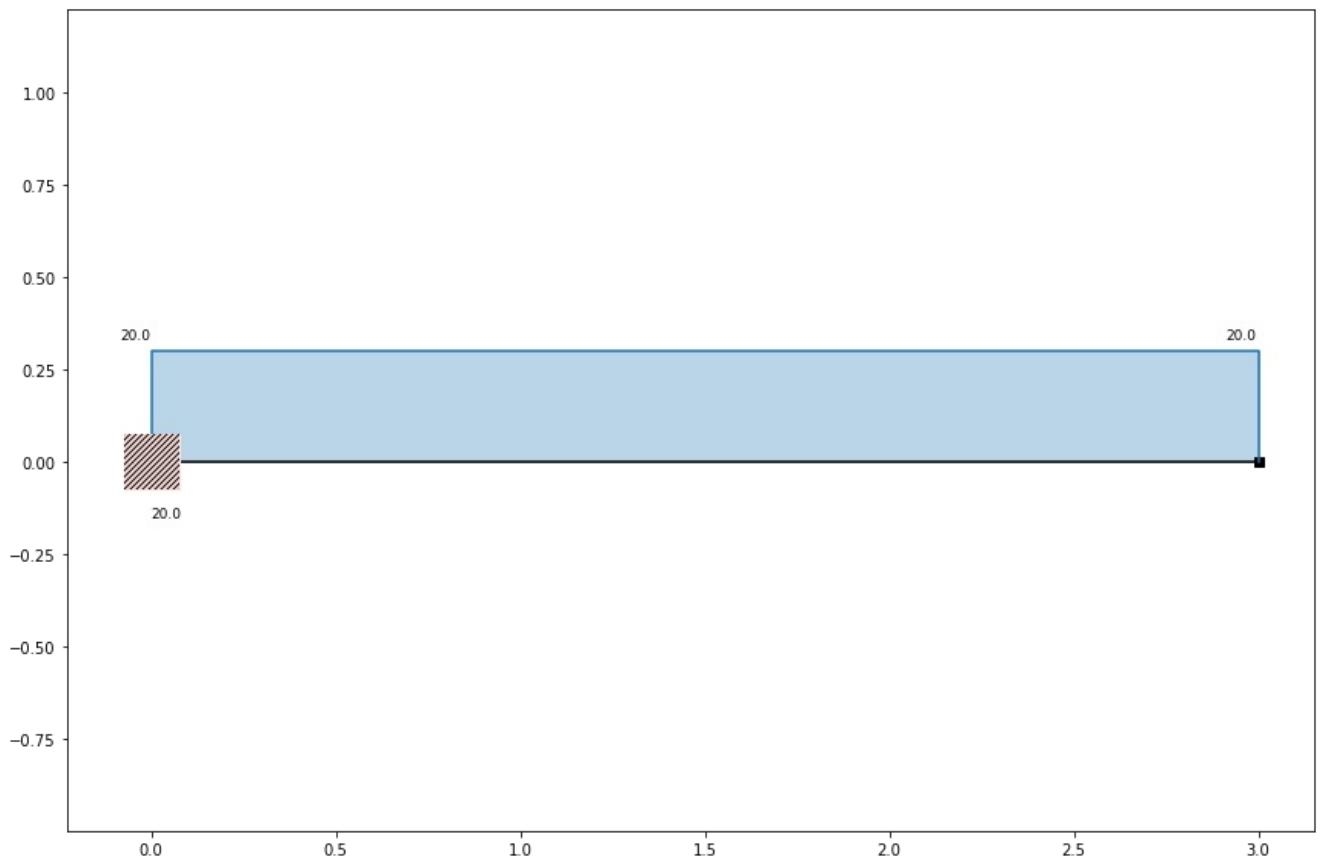
*Nodo: 1
Memento Mz: -20.00000000000004
```



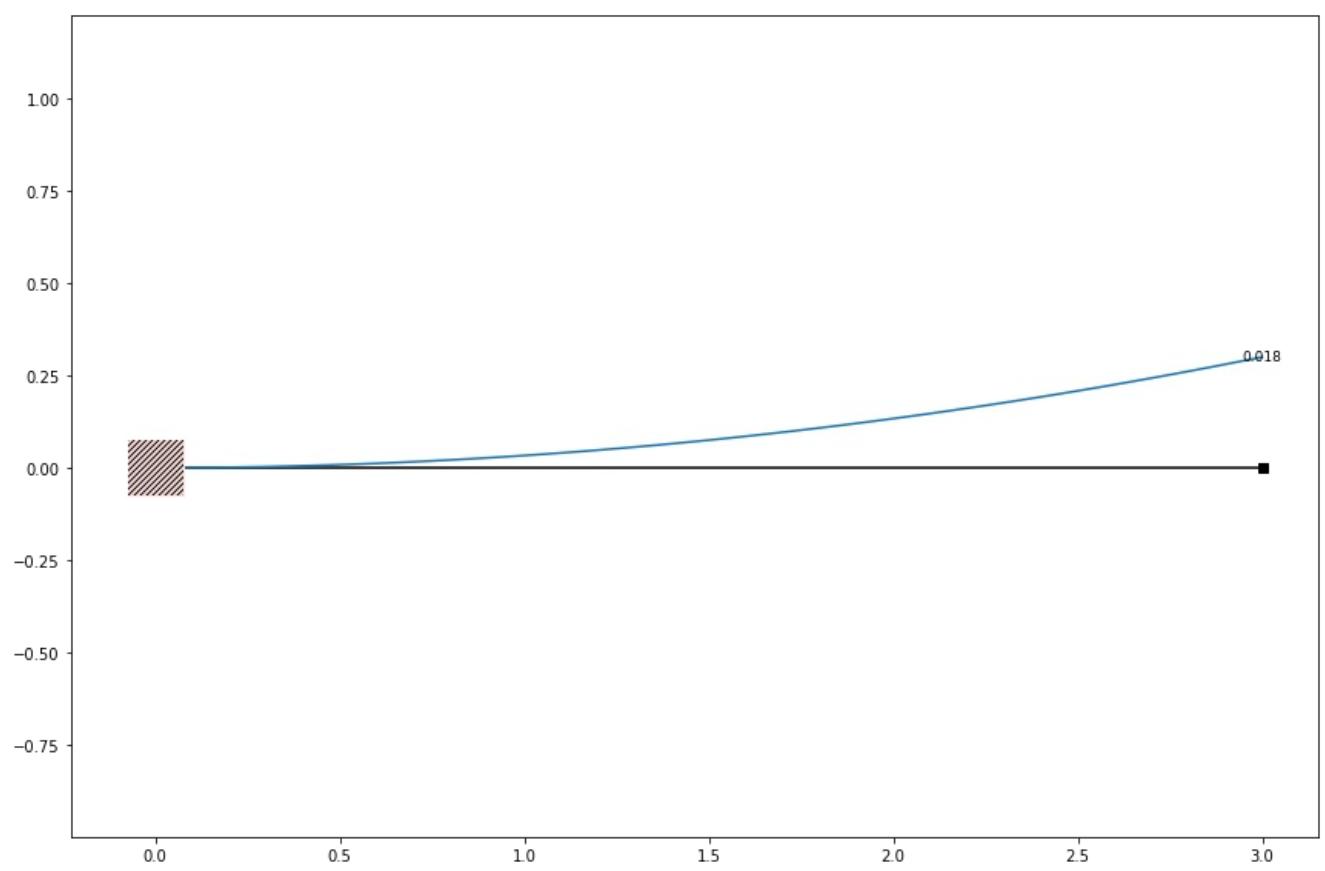
```
In [5]: # Mostramos cortantes
ss.show_shear_force()
```



```
In [6]: # Mostramos flectores
ss.show_bending_moment()
```



```
In [7]: ss.show_displacement()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
In []: # Ejemplo portico
```

```
Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-swbrixgy
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-swbrixgy
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.21.6)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.7.3)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages
(from matplotlib>=3.0->anastruct==0.0.0) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(1.4.4)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0)
(0.11.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib>=3.0->anastruct==0.0.0)
(4.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib>=3.0->anastruct==0.0.0)
(1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=59105 sha256=5b5e2c44ae909bd6170cc1c8968bd38e501f0a33803b1e8df37421b8e43b0d7c
 Stored in directory: /tmp/pip-ephem-wheel-cache-qxw967l6/wheels/23/97/1a/d460d2d29ccd27f0842a8e862290c78834c55be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

```
In []: # Cargamos el software anastruct
from anastruct import SystemElements
```

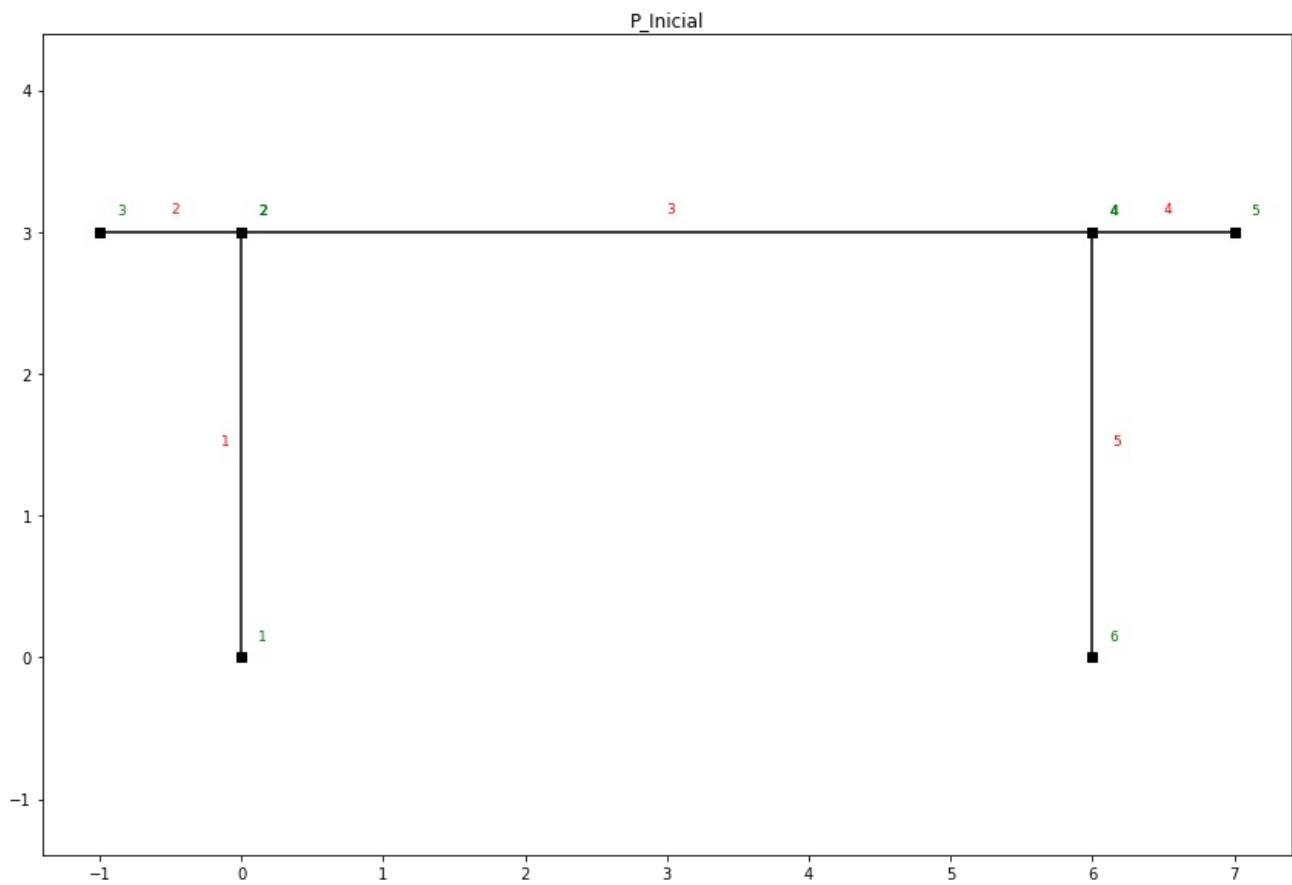
```
Cargamos el objeto de nuestra estructura
ss = SystemElements()
```

```
In []: # Variables
H = 3.0 # Altura del portico
L = 6.0 # Vano central
P1 = -5.0 # Carga 1
P2 = 10.0 # Carga 2
```

```
In []: # Construcción de la estructura

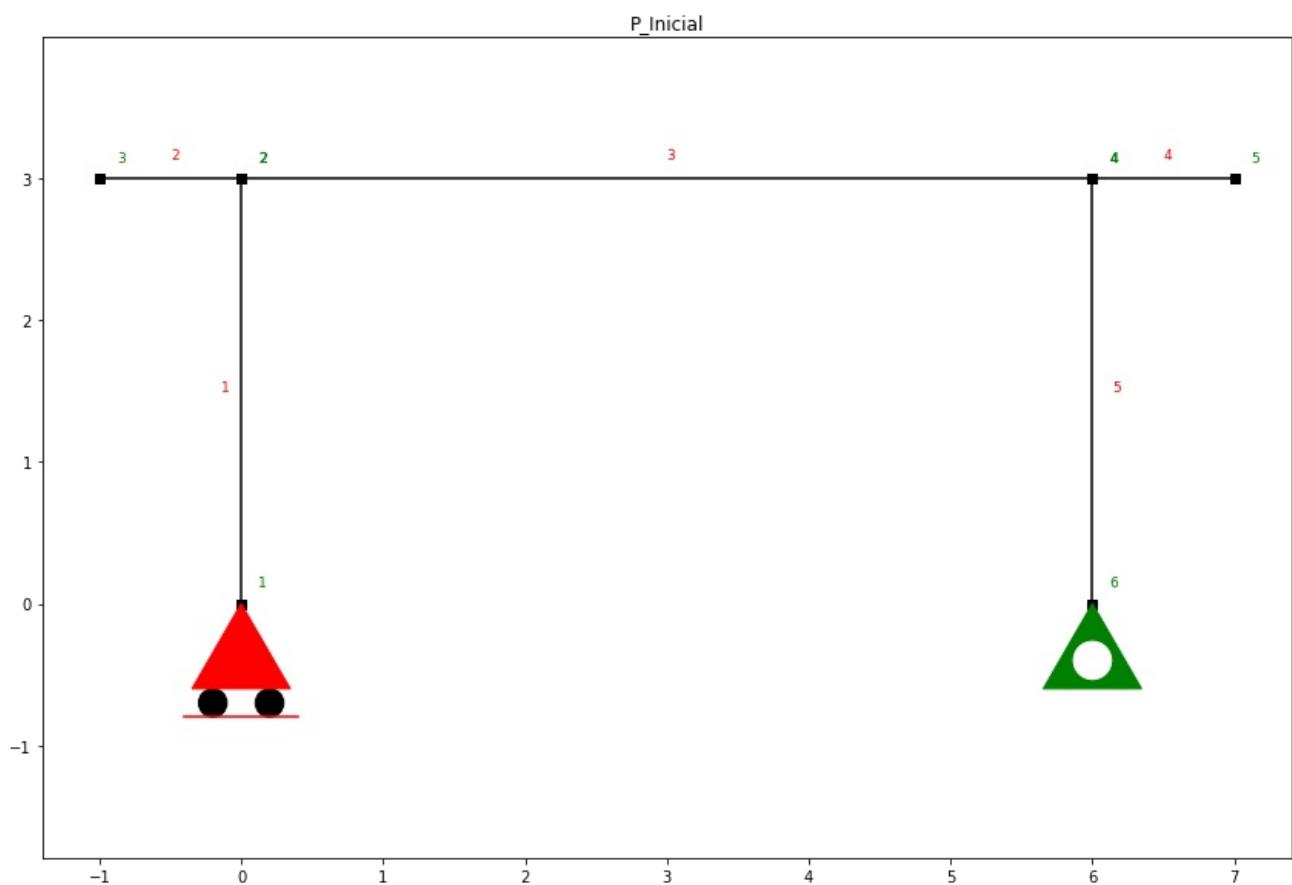
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [0, H]]);
Añadimos elemento barra 2
ss.add_element(location=[[-1, H], [0, H]]);
Añadimos elemento barra 3
ss.add_element(location=[[0, H], [L, H]]);
Añadimos elemento barra 4
ss.add_element(location=[[L, H], [L+1.0, H]]);
Añadimos elemento barra 5
ss.add_element(location=[[L, H], [L, 0]]);
```

```
In []: # Mostramos estructura generada
ss.show_structure(title='P_Inicial')
```



```
In []: # Añadimos apoyo fijo al nodo 1
ss.add_support_hinged(node_id=6)
Añadimos carrito al nodo 3
ss.add_support_roll(node_id=1, direction=2)
```

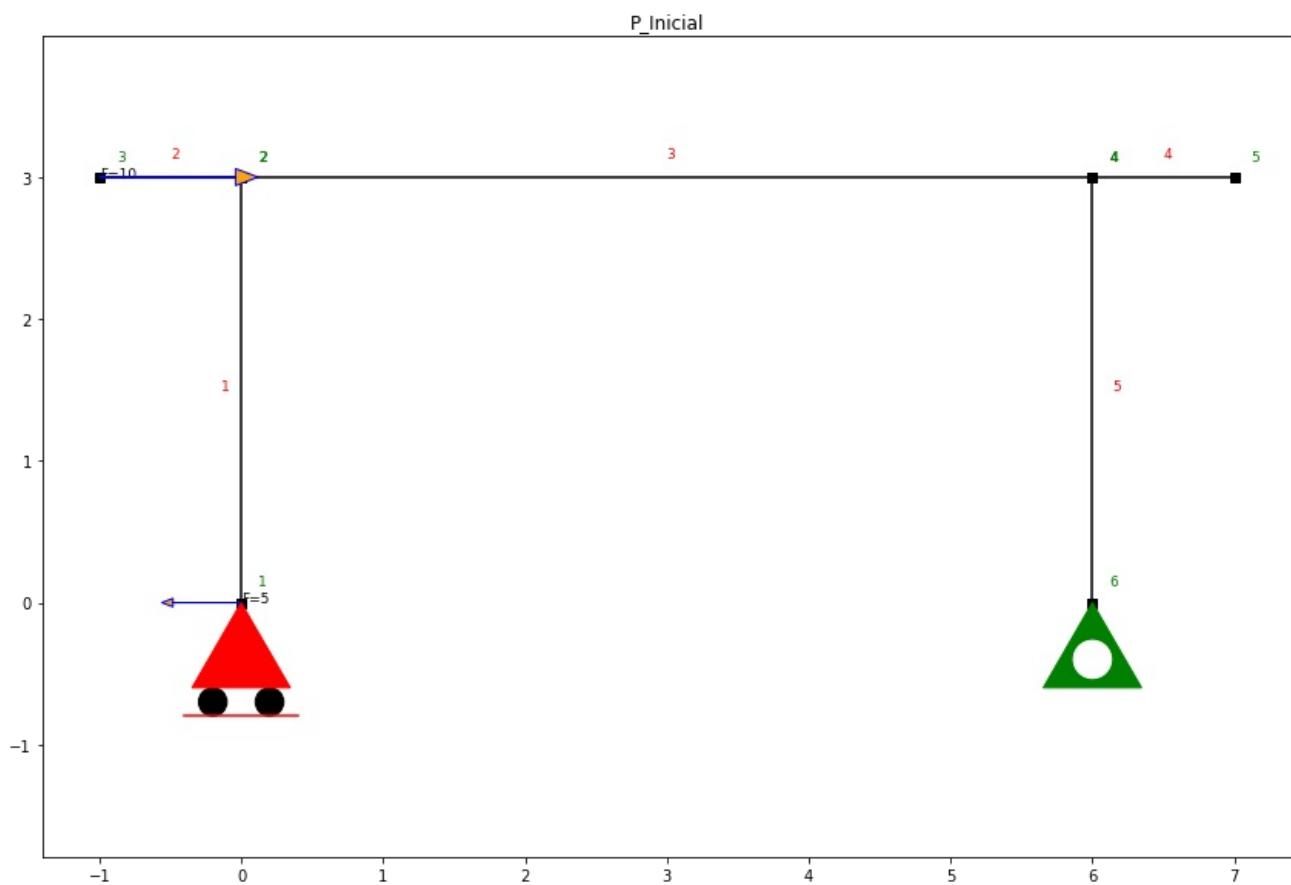
```
In []: # Mostramos estructura generada
ss.show_structure(title='P_Inicial')
```



```
In []: # Añadimos carga puntual al nodo 2
ss.point_load(1, Fx=P1, Fy=0.0)
Añadimos carga puntual al nodo 2
ss.point_load(3, Fx=P2, Fy=0.0)
```

```
In []: # Mostramos estructura generada
```

```
ss.show_structure(title='P_Inicial')
```

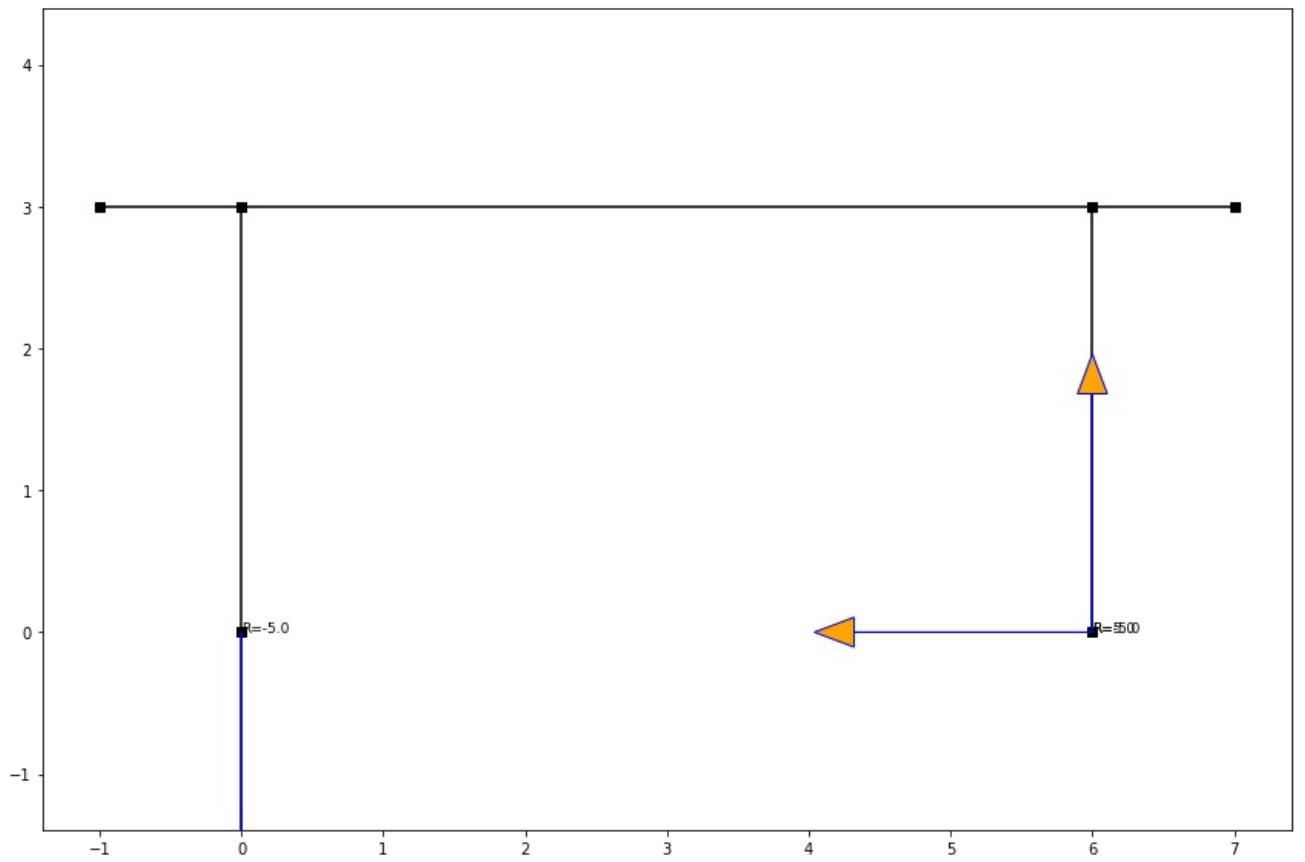


```
In []: # Resolvemos la estructura
ss.solve();
```

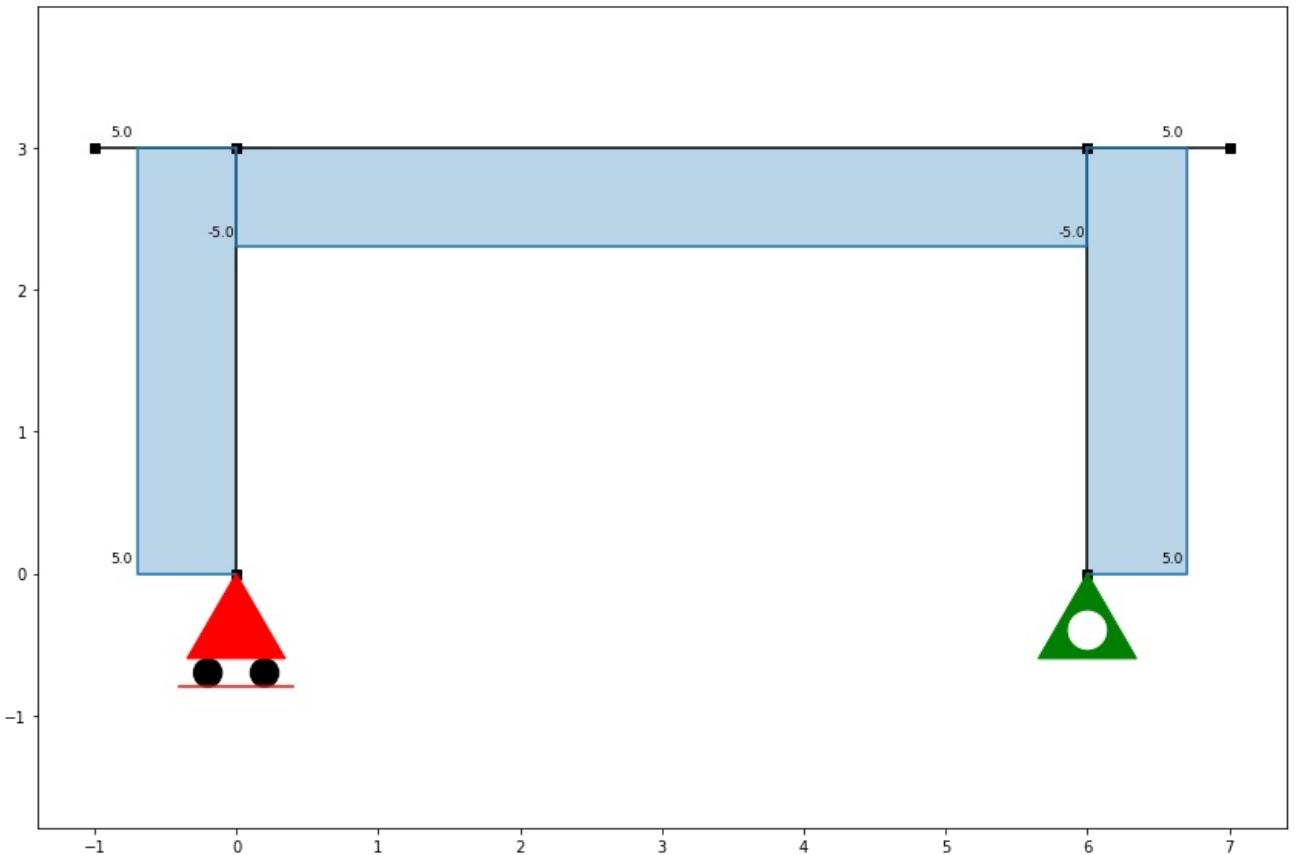
```
In []: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

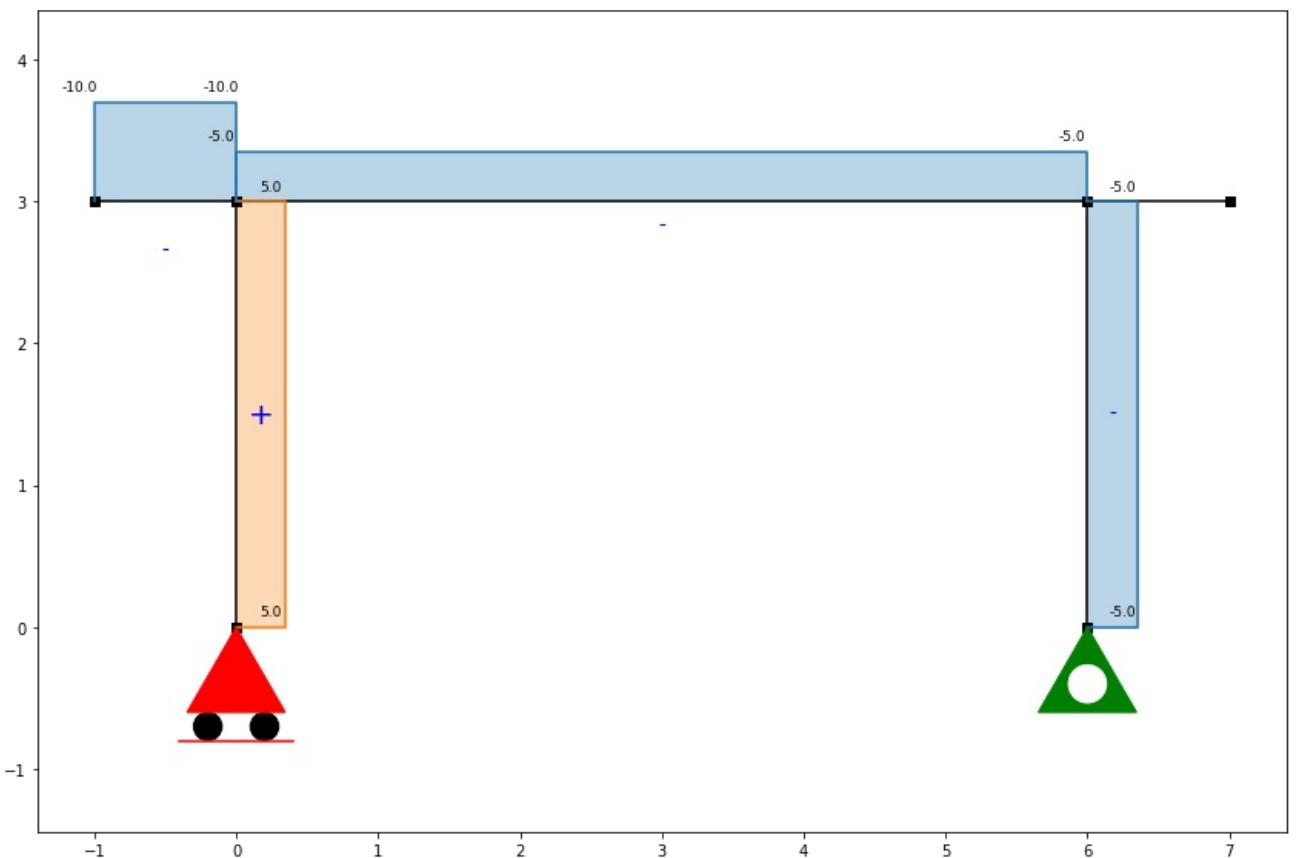
*Nodo: 6
Reacción Fx: -5.0
*Nodo: 6
Reacción Fy: 5.0
*Nodo: 1
Reacción Fy: -5.0
```



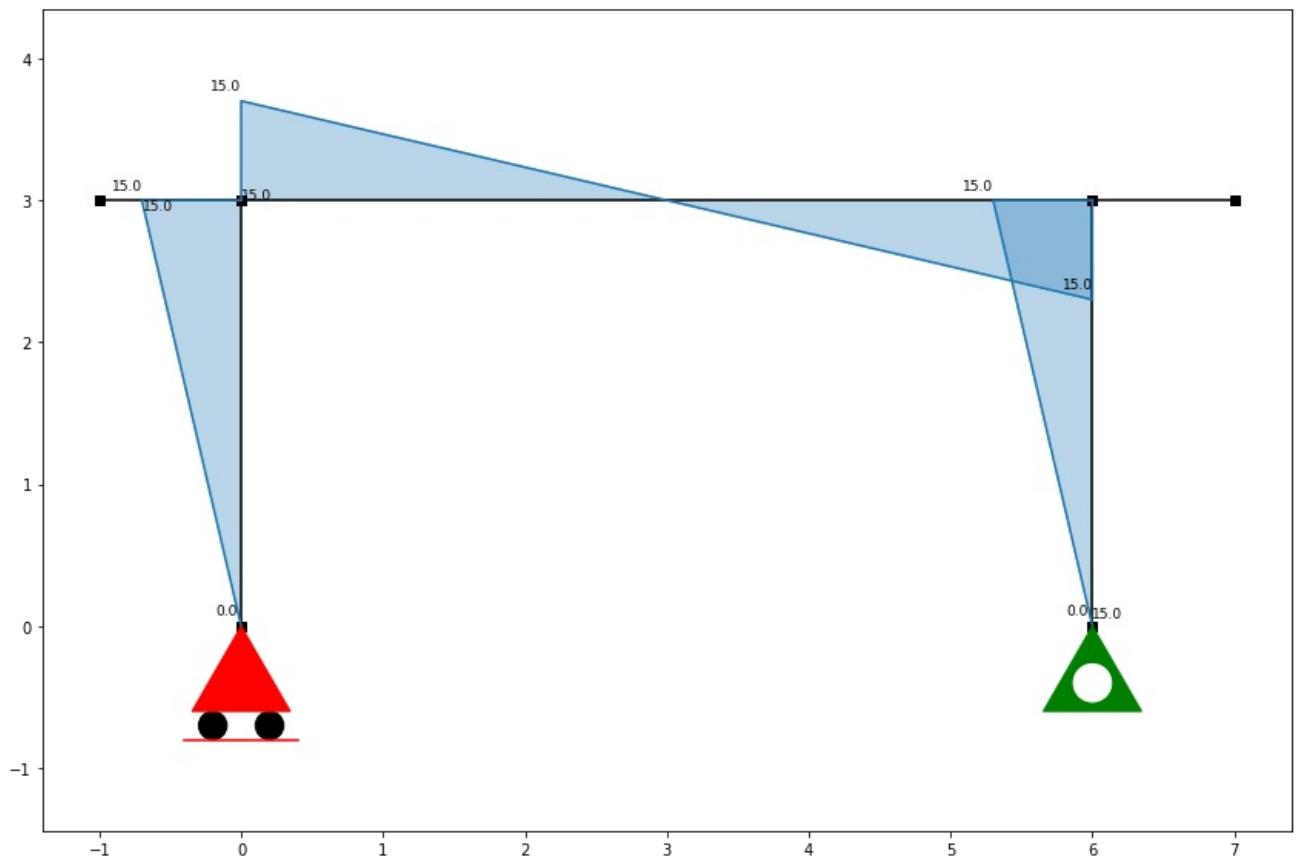
```
In []: # Mostramos cortantes
ss.show_shear_force()
```



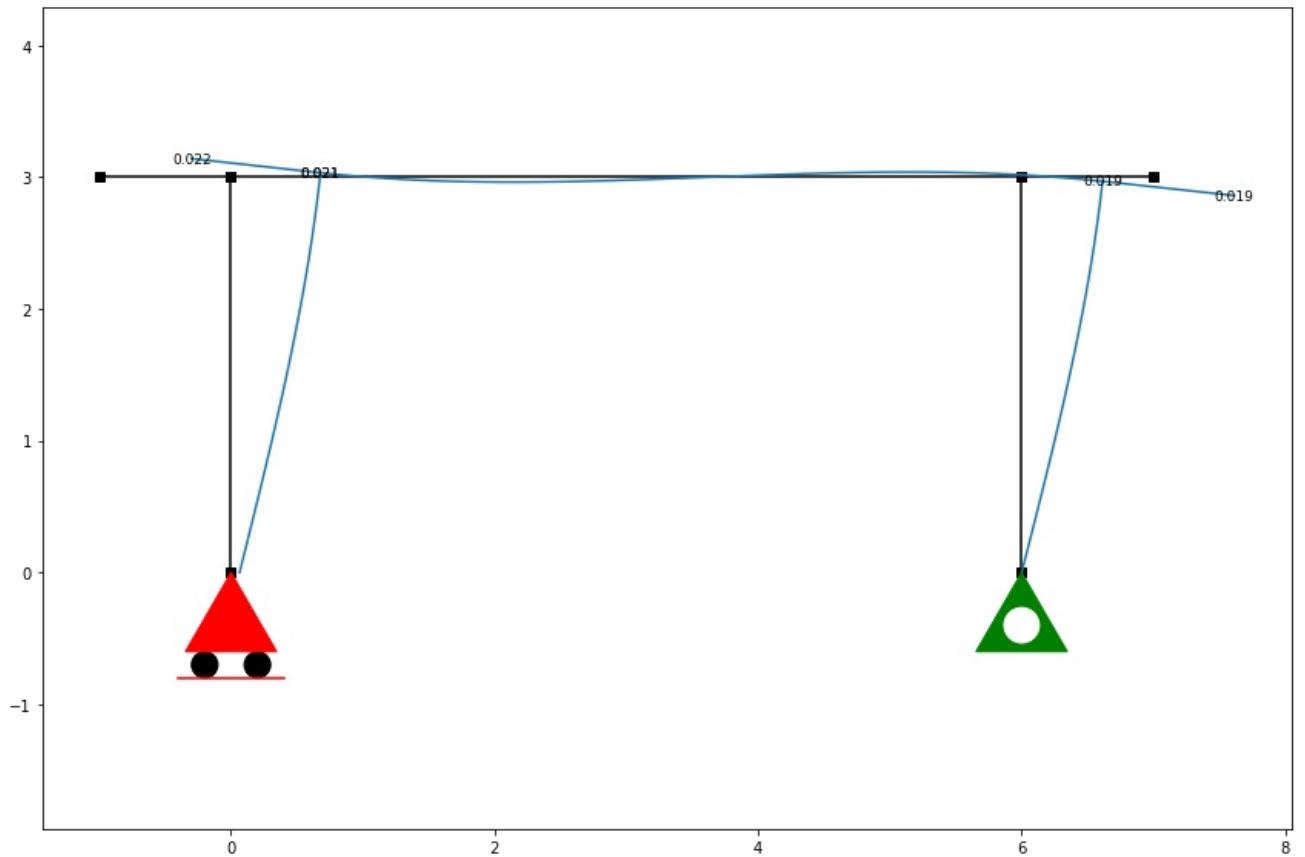
```
In []: # Axiles
ss.show_axial_force()
```



```
In []: ss.show_bending_moment()
```



In [ ]: `ss.show_displacement()`

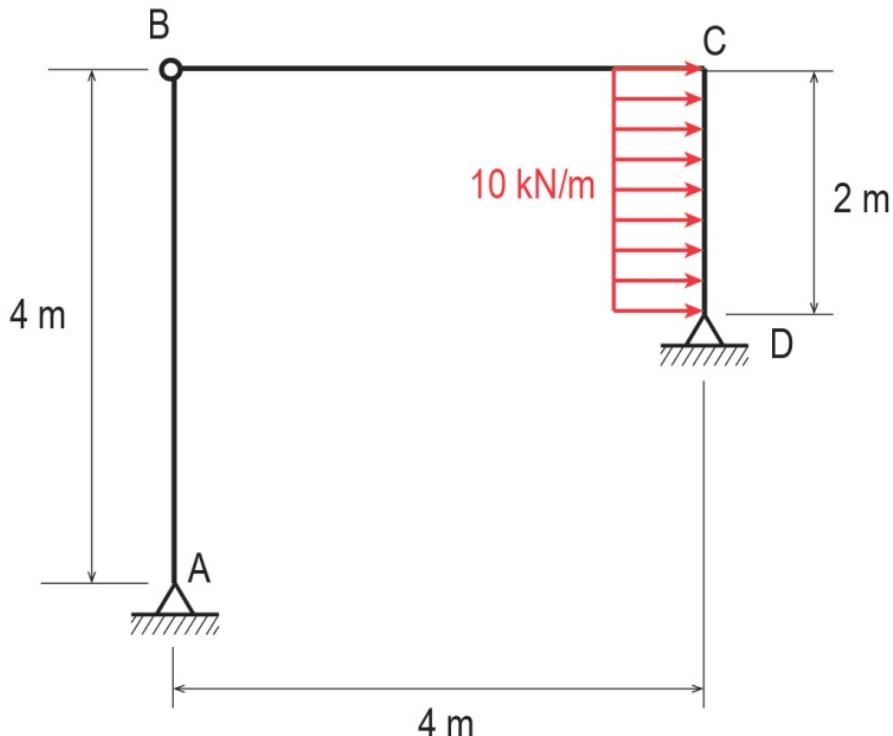


Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

# ESTRUCTURAS I - CURSO 2021-2022

Universidad de Granada

## Pórtico plano: Ejercicio 6 de la relación



```
In [1]: # Instalar paquete anastruct
!pip install git+https://github.com/EnriqueGarMac/Estructuras_I.git

Collecting git+https://github.com/EnriqueGarMac/Estructuras_I.git
 Cloning https://github.com/EnriqueGarMac/Estructuras_I.git to /tmp/pip-req-build-klb0mrjz
 Running command git clone -q https://github.com/EnriqueGarMac/Estructuras_I.git /tmp/pip-req-build-klb0mrjz
Requirement already satisfied: matplotlib>=3.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
) (3.2.2)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.19.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from anastruct==0.0.0)
(1.4.1)
Requirement already satisfied: pyparsing!=2.0.4,!!=2.1.2,!!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages
(from matplotlib>=3.0->anastruct==0.0.0) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=3.0->anastruct==0.0.0) (1.3.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=3.0->anastruct==0.0.0) (1.15.0)
Building wheels for collected packages: anastruct
 Building wheel for anastruct (setup.py) ... done
 Created wheel for anastruct: filename=anastruct-0.0.0-py3-none-any.whl size=58952 sha256=307ff8ed9a6a3484340e13cc3ca2a40913fcc655914b27589e9fe5189c296040
 Stored in directory: /tmp/pip-ephem-wheel-cache-1snndd_v/wheels/23/97/1a/d460d2d29ccd27f0842a8e862290c78834c55be12783542415
Successfully built anastruct
Installing collected packages: anastruct
Successfully installed anastruct-0.0.0
```

```
In [18]: # Cargamos el software anastruct
from anastruct import SystemElements

Cargamos el objeto para las viguetas
ss = SystemElements()
```

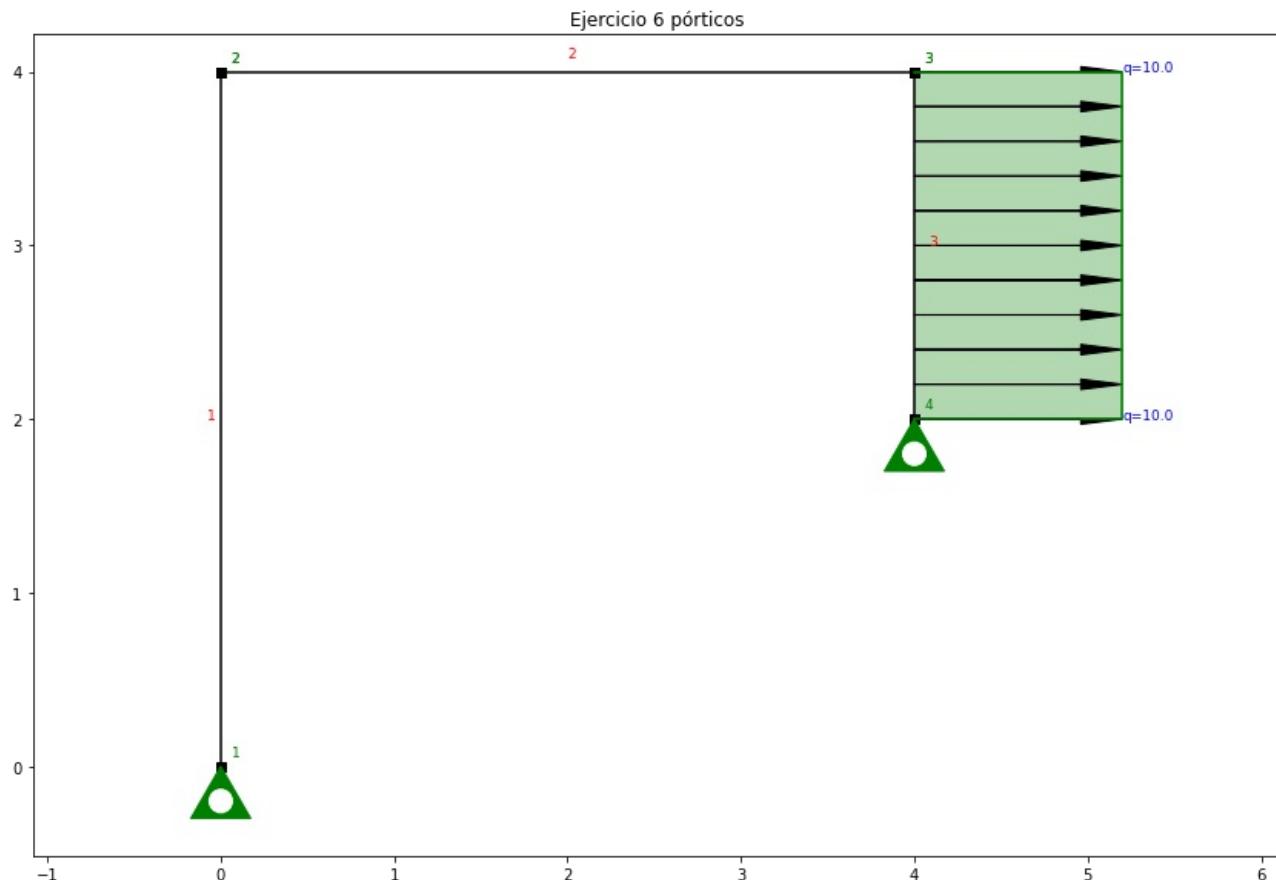
```
In [19]: # Vamos a definir ahora algunas variables
L = 4.0 # Luz del pórtico [m]
h1 = 4.0 # Altura del primer soporte [m]
h2 = 2.0 # Altura del segundo soporte [m]
q = 10 # Carga distribuida [kN/m]
```

```
In [20]: # Construcción de la estructura
Añadimos elemento barra 1
ss.add_element(location=[[0, 0], [0, h1]], spring={2: 0});
Añadimos elemento barra 1
ss.add_element(location=[[0, h1], [L, h1]]);
Añadimos elemento barra 1
ss.add_element(location=[[L, h1], [L, h1-h2]]);
```

```
In [21]: # Añadimos apoyo fijo al nodo 1
ss.add_support_hinged(node_id=1)
Añadimos carrito al nodo 3
ss.add_support_hinged(node_id=4)
```

```
In [22]: # Añadimos carga uniformemente distribuida
ss.q_load(element_id=3, q=-q)
```

```
In [23]: # Mostramos estructura generada
ss.show_structure(title='Ejercicio 6 pórticos')
```

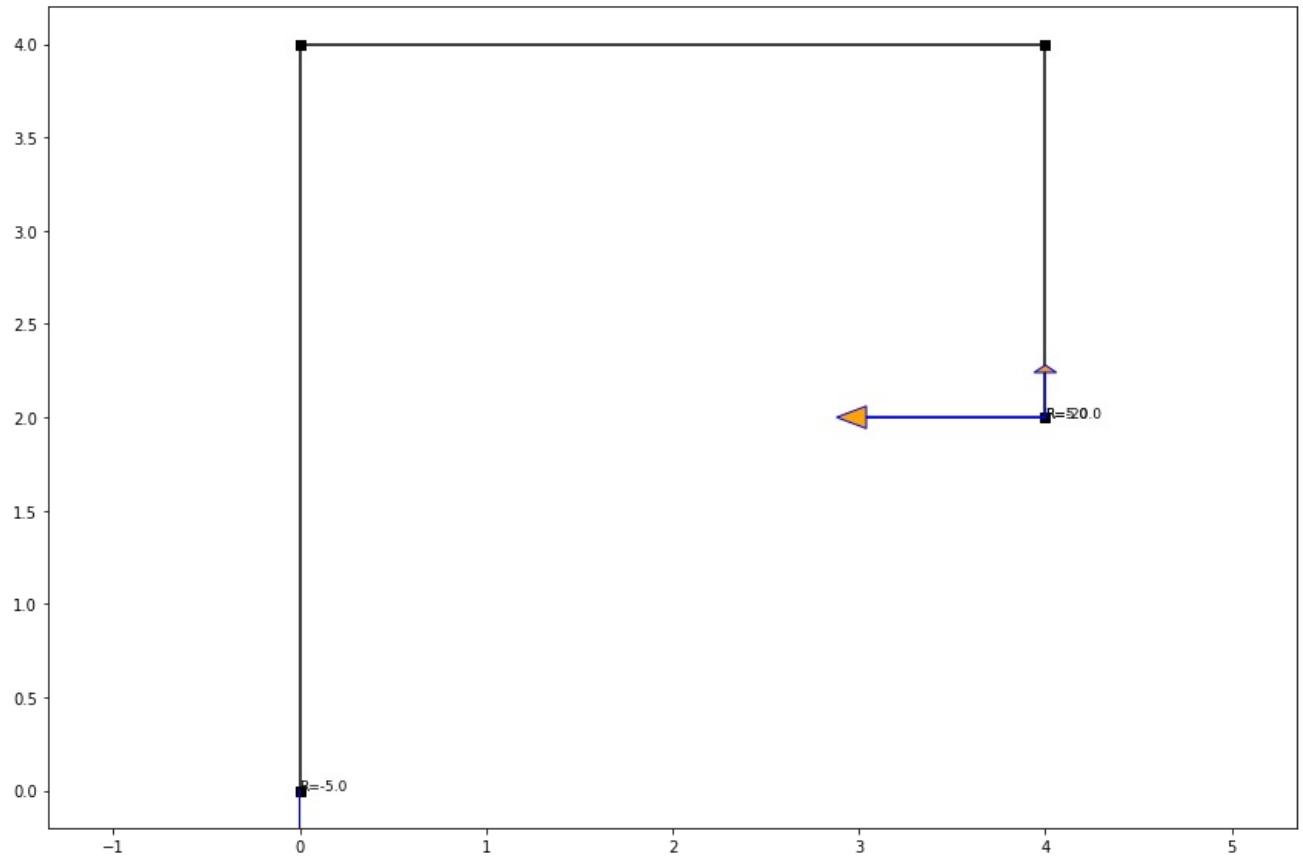


```
In [26]: # Resolvemos la estructura
ss.solve();
```

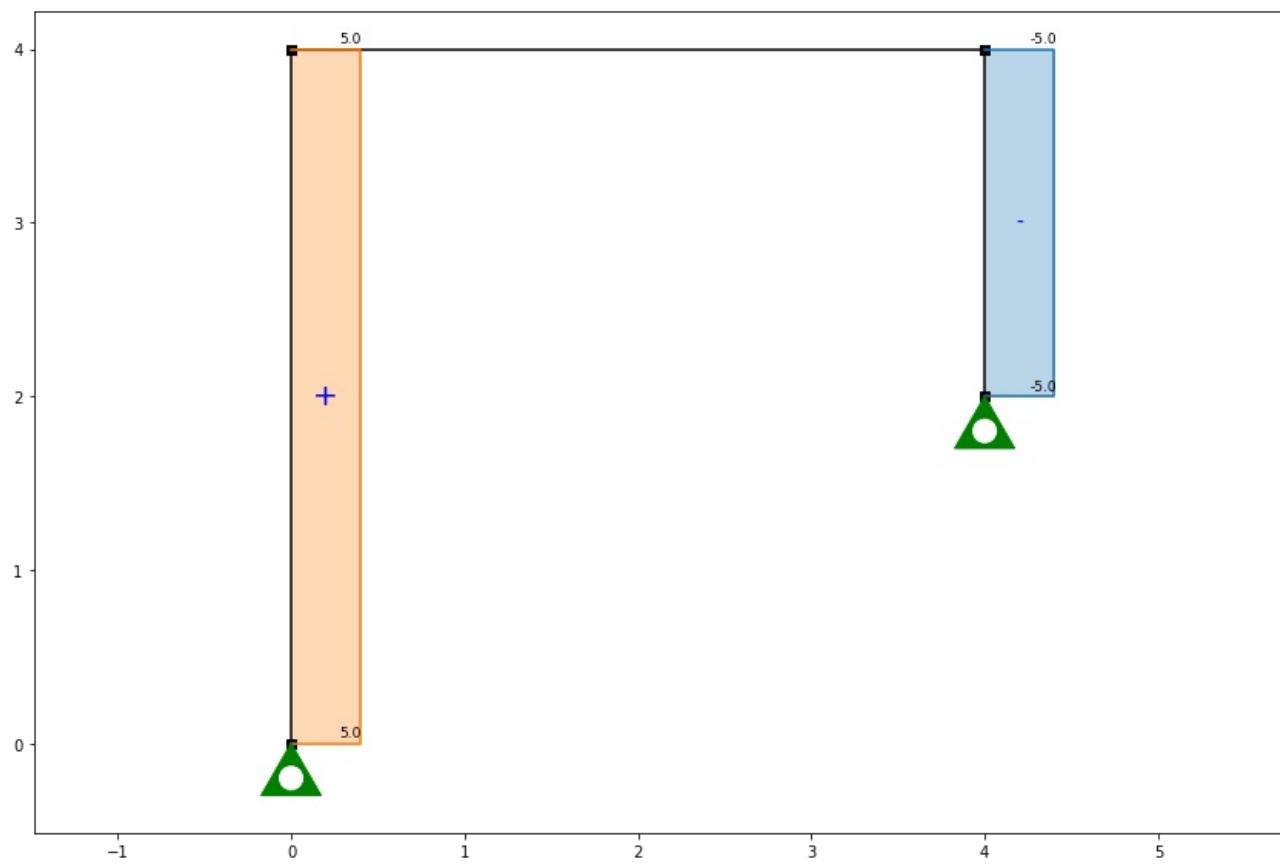
```
In [27]: # Mostramos las reacciones
ss.show_reaction_force()
```

```
Reacciones

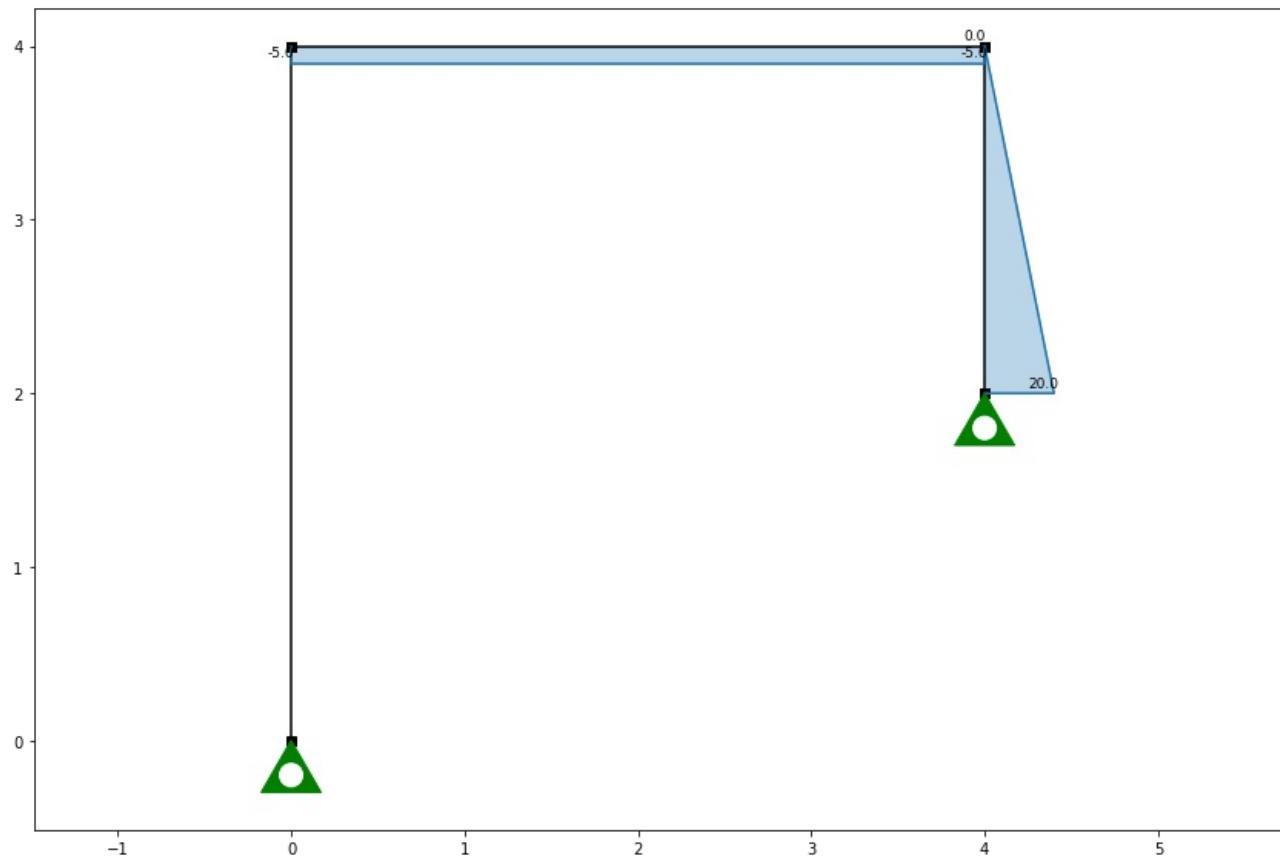
*Nodo: 1
Reacción Fy: -5.0
*Nodo: 4
Reacción Fx: -20.0
*Nodo: 4
Reacción Fy: 5.0
```



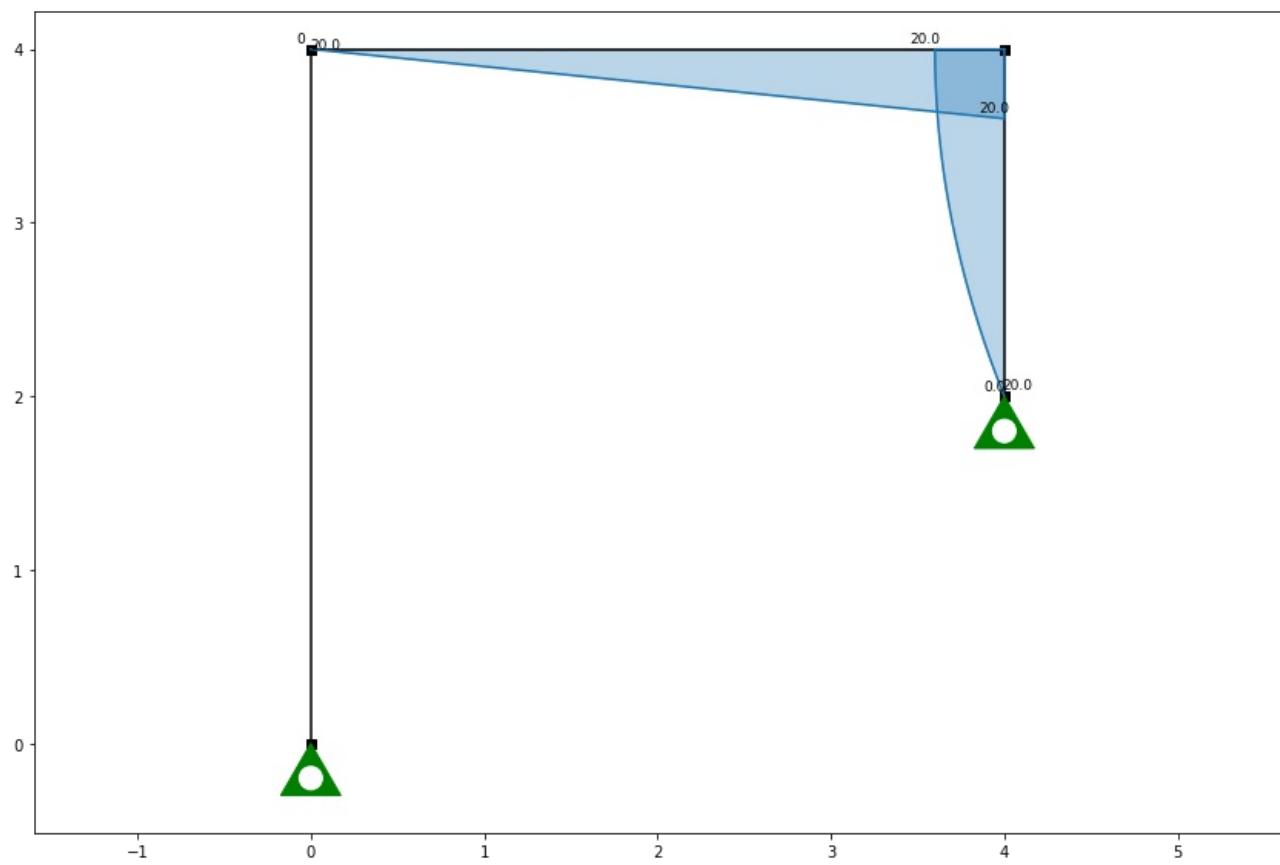
```
In [31]: # Mostramos axiles
ss.show_axial_force()
```



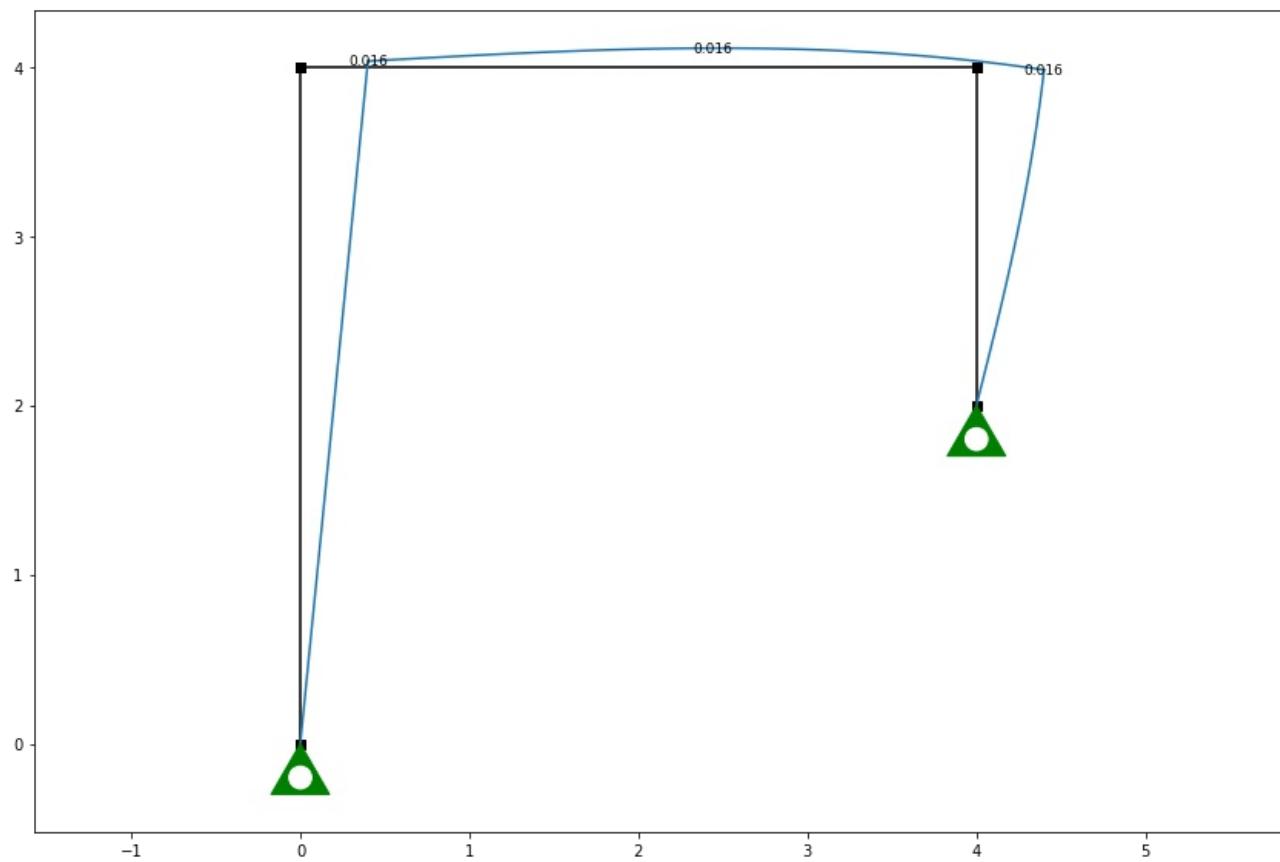
```
In [28]: # Mostramos cortantes
ss.show_shear_force()
```



```
In [29]: # Mostramos flectores
ss.show_bending_moment()
```



```
In [30]: # Mostramos deformada
ss.show_displacement()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js