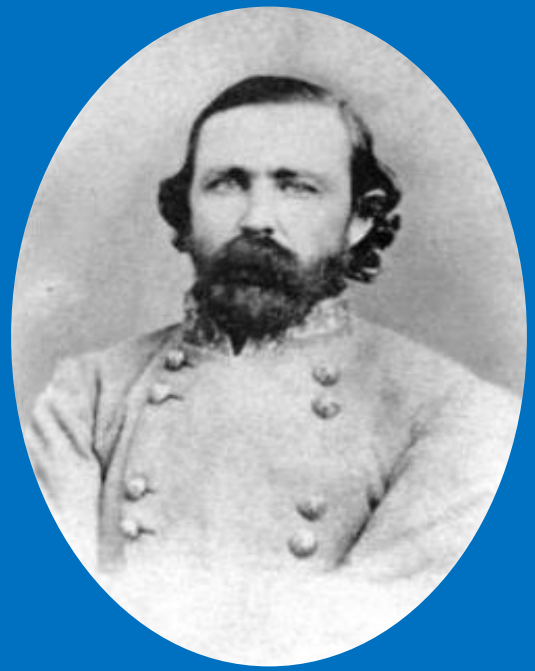
NIO

JOEL PÉREZ ZAHARANIS GOZAINE LUIS HERRERA EDWIN PIMENTEL ALEXIS CUBILLA

MARCO TEÓRICO

- Fue desarrollado por George Un Maney en 1914.
- Análisis de vigas y marcos indeterminados.
- Utilizado por más de una década hasta el desarrollo del método de distribución de momentos.



- Consiste en establecer ecuaciones con los desplazamientos en los nod os (giros y desplazamientos line ales) para caracterizar completa mente la configuración de la deformada de la estructura.

GRADOS DE LIBERTAD

- Los desplazamientos desconocidos en los nodos de un elemento estructural cargado.

- Consiste en ponerle nombre a los soportes y nudos para poder identificar los claros de la barra entre los nodos, dibujando así la deformada de la barra.

- En los métodos de desplazamiento los grados de libertad son iguales a las incógnitas del sistema, los grados de libertad indicarán las ecuaciones de compatibilidad necesarias.

ECUACIONES

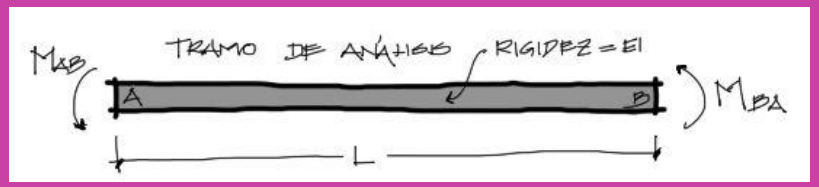
$$M_{AB} = \frac{2EI}{L}\left(2\theta_A + \theta_B - \frac{3\Delta}{L}\right) + MEP$$

$$M_{BA} = \frac{2EI}{L}\left(\theta_A + 2\theta_B - \frac{3\Delta}{L}\right) + MEP$$

- **A y B son los dos extremos de un tramo de la viga entre dos apoyos.**

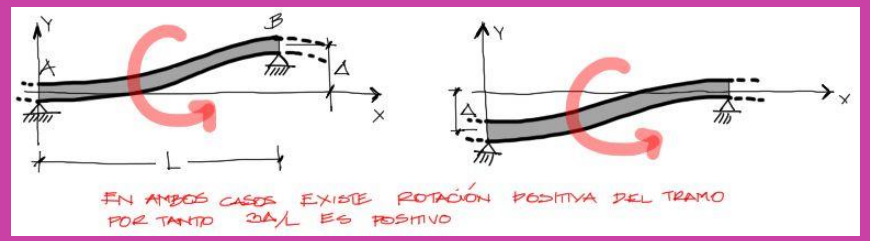
CONVENCIÓN DE SIGNOS

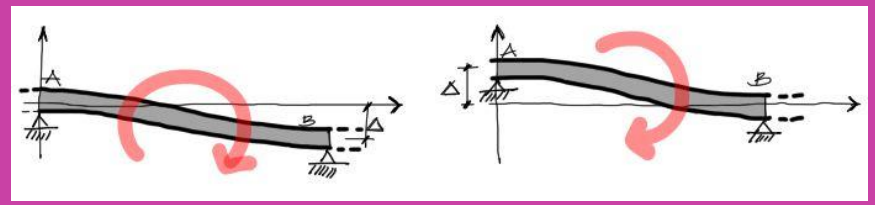
- Llevarán signo positivo si rotan en el sentido antihorario.



- Cuando se trata de giros, estos se miden en radianes [rad]

- Confusión respecto al asentamiento Δ

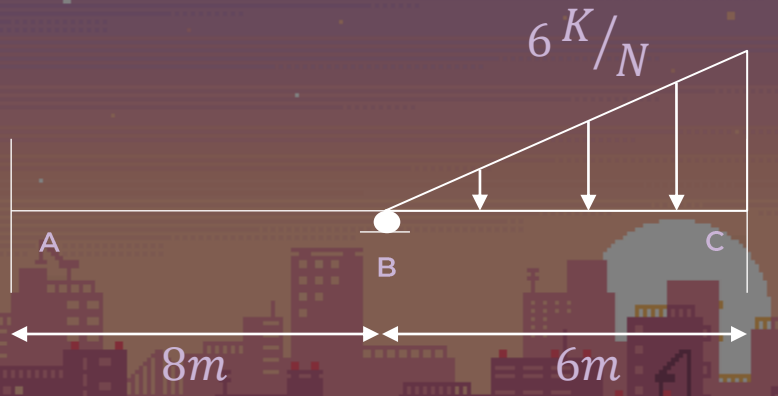




MÉTODO DE DEFLEXIÓN

DE EJEMPLO PRBLEMA

PRIMER PASO: DEFINIR LOS FEM



EI $= Constante$

$$BC = -\frac{WL^2}{30} \qquad CB = \frac{WL^2}{20}$$

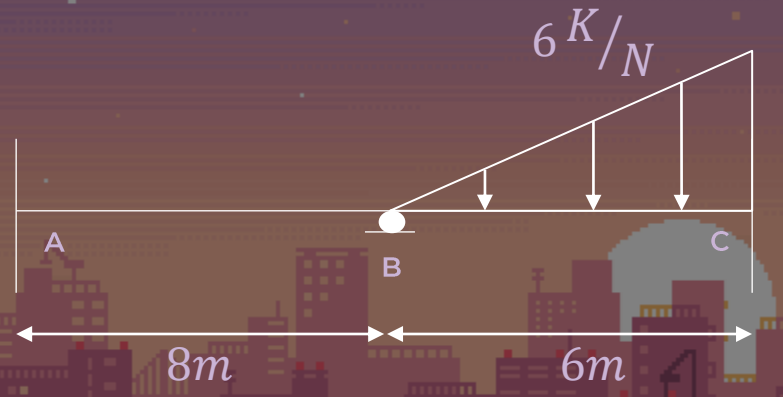
$$BC = -\frac{6\,(6)^2}{30} \qquad CB = \frac{6(6)^2}{20}$$

$$BC = -7.2\ KN \cdot m \qquad CB = 10.8\ \mathrm{KN} \cdot m$$

NOTA:

- No se analizan lo tramos AB y BA debido a que no hay ninguna fuerza y se assume que el resultado será 0.

SEGUNDO PASO: ANÁLISIS DE MOMENTOS POR TRAMO



EI $= Constante$

NOTA:

- La formula que se utilizará para éste paso será la siguiente:

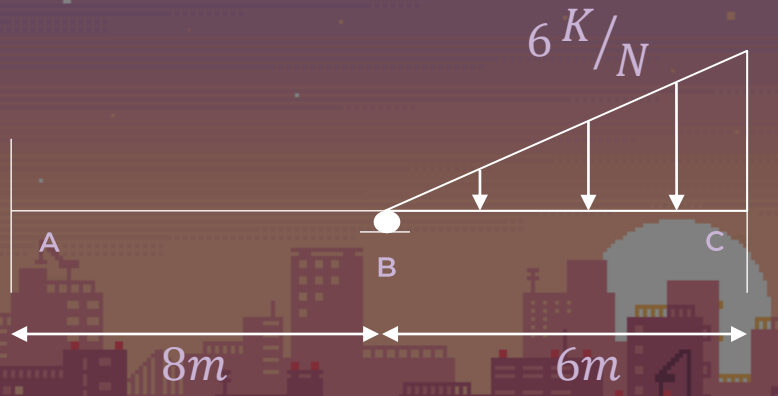
$$M_{NF} = 2\frac{EI}{L}\left(2\theta_N + \theta_F - \frac{3\Delta}{L}\right) + FEM_N$$

$$M_{AB} = 2\frac{EI}{8}(0 + \theta_B - 0) + 0 \qquad M_{BA} = 2\frac{EI}{8}(2\theta_B + 0 - 0) + 0$$

$$M_{AB} = \frac{EI}{4}\theta_B \qquad M_{BA} = \frac{EI}{2}\theta_B$$

$$M_{BC} = 2\frac{EI}{6}(2\theta_B + 0 - 0) + 0 \qquad M_{CB} = 2\frac{EI}{6}(0 + \theta_B - 0) + 0$$

$$M_{BC} = 2\frac{EI}{3}\theta_B - 7.2\ \text{KN} \cdot m \qquad M_{CB} = \frac{EI}{3}\theta_B + 10.8\ \text{KN} \cdot m$$



EI = *Constante*

NOTA:

- Se realiza un sistema de ecuaciones con Los resultados de los análisis de momentos por tramo.
- Se tomo el BA y BC para realizar el sistema de ecuaciones.

TERCER PASO: SISTEMA DE ECUACIONES

$$M_{BA} + M_{BC} = 0$$

$$\frac{EI}{2}\theta_B + 2\frac{EI}{3}\theta_B - 7.2\ \text{KN} \cdot m = 0$$

$$\theta_B = \frac{6.17}{EI}$$

CUARTO PASO: REEMPLAZAR Y RESULTADOS FINALES

$$M_{AB} = 1.54\ \text{KN} \cdot m \qquad M_{BC} = -3.09\ \text{KN} \cdot m$$

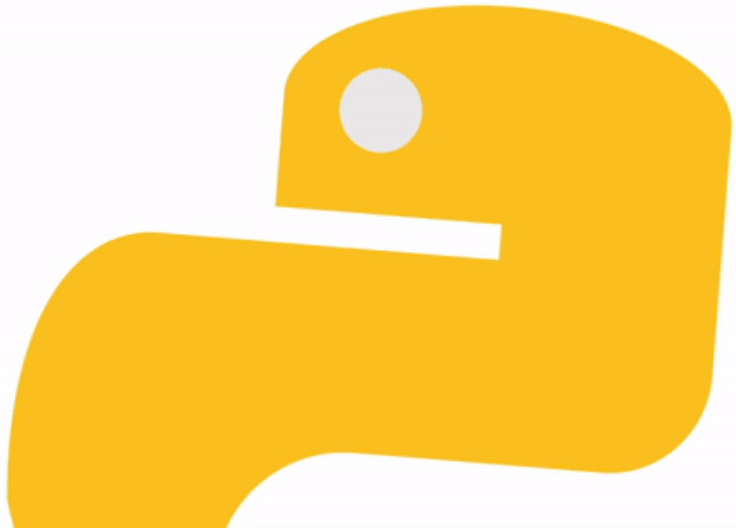
$$M_{BA} = 3.09\ \text{KN} \cdot m \qquad M_{CB} = 12.86\ \text{KN} \cdot m$$

¿QUÉ ES PYTHON?

Historia

Python fue creado a finales de los ochenta por Guido van Rossum en el Centro para las Matemáticas y la Informática, en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba.

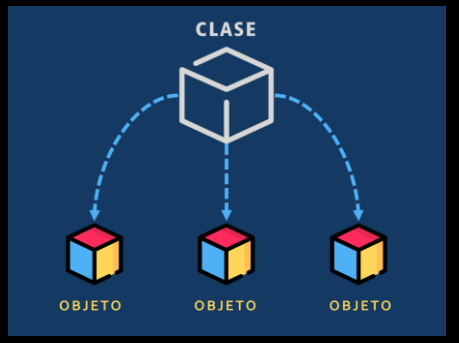
Además en este lanzamiento inicial aparecía un sistema de módulos adoptado de Modula-3; van Rossum describe el módulo como «una de las mayores unidades de programación de Python». El modelo de excepciones en Python es parecido al de Modula-3, con la adición de una cláusula else. En el año 1994 se formó “comp.lang.Python”, el foro de discusión principal de Python, marcando un hito en el crecimiento del grupo de usuarios de este lenguaje.



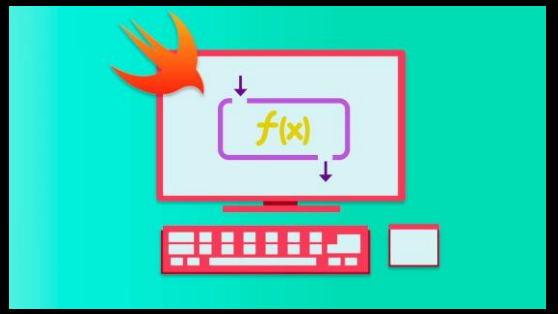
CARACTERÍSTICAS

- Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional.

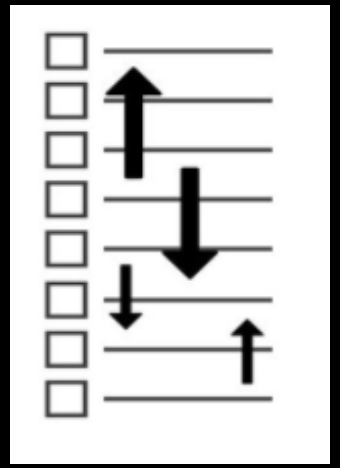
LA PROGRAMACIÓN ORIENTADA HA OBJETOS



LA PROGRAMACIÓN FUNCIONAL



PROGRAMACIÓN IMPERATIVA

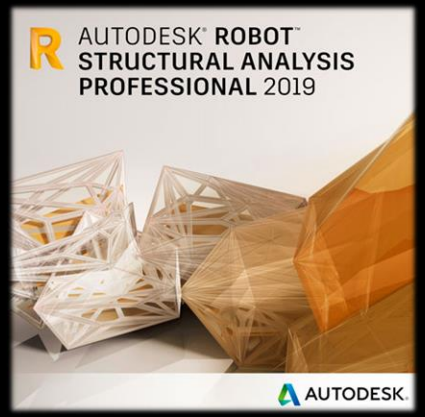


ANÁLISIS CON PROGRAMAS ESTRUCTURALES

- El método de los elementos finitos (MEF), también conocido como Finite Element Method (FEM) en inglés, es un procedimiento numérico pensado para su uso en software informático. En el ámbito de la ingeniería y la construcción es muy utilizado para el cálculo de resistencia de materiales y estructuras.
- La dinámica de funcionamiento de la simulación FEM es dividir un problema, a priori muy complejo, en un número limitado de partes más pequeñas más sencillas de analizar y así poder encontrar una solución global.
- Ejemplos.

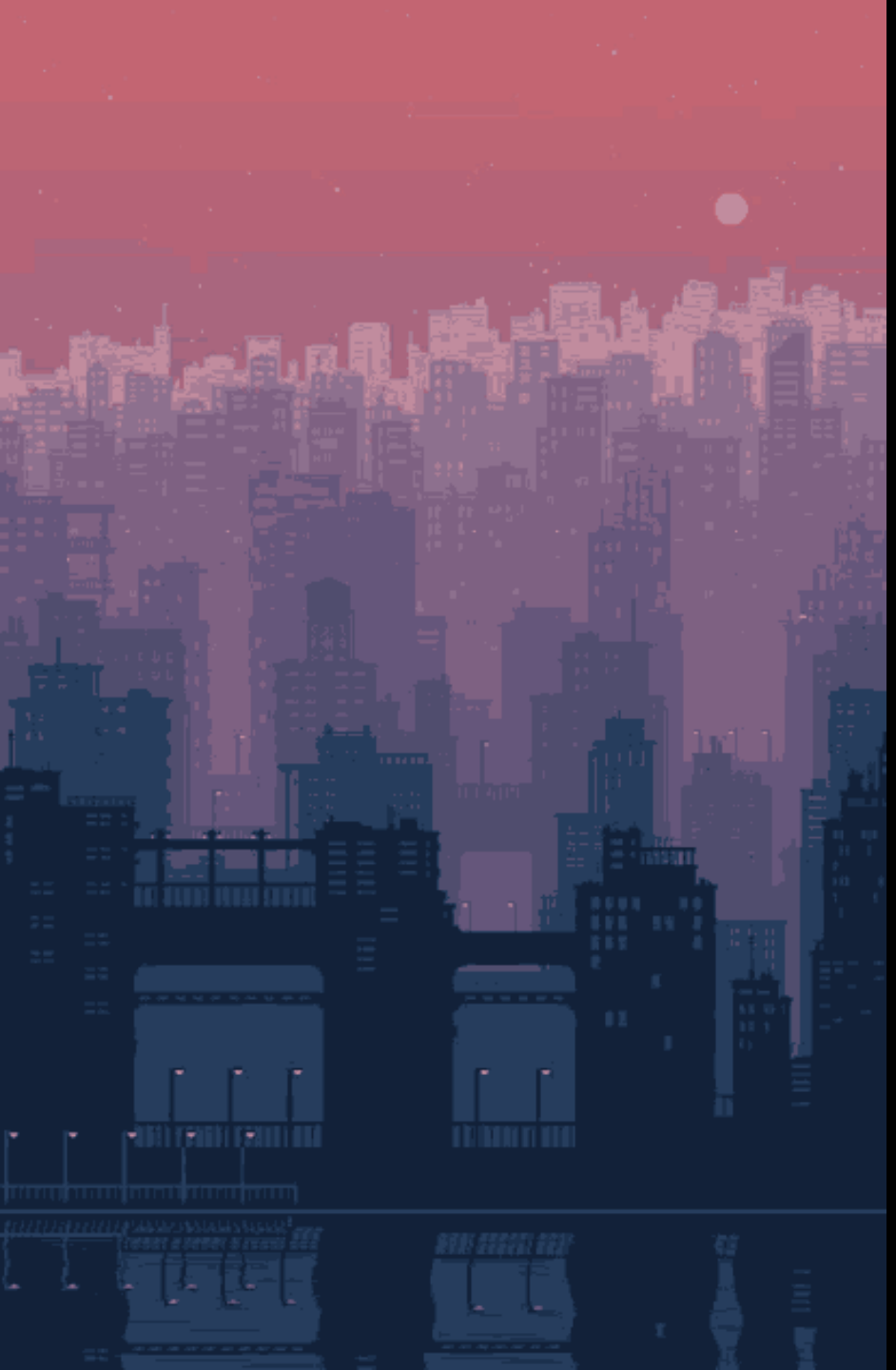
SIMSCALE

Simcenter 3D



SAP2000

INTRODUCCIÓN



CARACTERÍSTICAS

- Programa de elementos finitos
- Interfaz gráfico 3D para objetos
- Preparado para realizar, de forma totalmente integrada, la modelación, análisis y dimensionamiento de lo más amplio conjunto de problemas de ingeniería de estructuras.

¿QUÉ PERMITE SAP2000?

- Generar automáticamente:
  - Cargas de sismos.
  - Vientos.
  - Vehículos.
- Dimensionamiento y comprobación automática de:
  - Estructuras de hormigón armado.
  - Perfiles metálicos, de aluminio y conformados en frío.

SCRIPT NIO

PENDIENTE DEFLEXIÓN

SCRIPT NIO

PENDIENTE DEFLEXIÓN

SCRIPT NIO

PENDIENTE DEFLEXIÓN

SCRIPT NIO

PENDIENTE DEFLEXIÓN

ANALIZAR LA MEJOR TECNOLOGÍA

Determinar la tecnología a utilizar depende del tipo de trabajo a realizar, los tiempos de entrega, el presupuesto y la proyección del Proyecto.

Ver Más

AVANZAR

Luego cada integrante del equipo asumió un rol y lo desarrollo siguiendo los parámetros establecidos.

INFORME

Al finalizar el Script se redacto un informe con la implementación de dos ejemplo y anexado el código utilizado.

REUNION

Primero el equipo se reunió para establecer el sistema de diseño que determinaría el desarrollo del Script.

INSPECCIÓN

Para mantener un orden y armonia dentro del proyecto se establecieron inspecciones en grupo para evitar errores.

CHUZA!

Celebramos el éxito y culminación del proyecto y determinamos que este Script puede ser de gran ayuda para los demás.

PROS & CONSTRAS

Utilizar python en el área de las matemáticas no tiene ninguna contra, al contrario, es aqui donde se aprovecha su maximo potencial.

Ademas uno de los principales beneficios que presenta este en la escalabilidad que se le pueden dar a los script ya que estos pueden unirse he integrarse para formar softwares mas avanzados

Ver Más

SCRIPT NIO A FUTURO

Nio es un pequeño pero muy potente script que seguira creciendo con el pasar del tiempo por parte de nuestro equipo.

El objetivo final es incorporarle más casos y funcionalidades para que sea una opción gratuita y de gran ayuda para estudiantes y docentes

Ver Más


¿PORQUE PROGRAMACIÓN
PORQUE PYTHON?

MENÚ

NIO

SAP 2000

TESIS

INFORME

Gracias Por su Atención

UNIVERSIDAD SANTA MARÍA LA ANTIÜA

SEDE DAVID

ARQUITECTURA Y DISEÑO

ARQUITECTURA ESTRUCTURAL

ANALISIS ESTRUCTURAL III

PROF. ALONSO GONZALES LEZCANO

INFORME DE PROYECTO DE PROGRAMACÓN

INTEGRANTES:

ALEXIS D. CUBILLA M.	4-801-1260
ZAHARANI P. GOZAINE JEMÉNEZ	4-794-118
LUIS ABAD HERRERA PEÑA	4-796-1859
JOEL A. PÉREZ V.	4-800-858
EDWIN PIMENTEL L.	4-782-1948

TERCER CUATRIMESTRE

AÑO 2021

MARCO TEÓRICO

A lo largo de la historia el análisis de estructuras ha venido siendo indispensable en construcciones de estructuras que se conforman por barras, por eso se han creado algunos métodos de cálculos que sirven y a medida que pasa el tiempo facilitan el procedimiento de cálculo al personal ejecutor.

El método de pendiente deflexión fue desarrollado originalmente por Heinrich Manderla y Otto Mohr con el fin de estudiar los esfuerzos secundarios en armaduras. En 1914 por George Un. Maney desarrollo una versión perfeccionada y la aplico al análisis de vigas y marcos indeterminados. La pendiente-deflexión fue el método ampliamente utilizado por más de una década hasta el desarrollo del método de distribución de momentos.

El método pendiente – deflexión, se utiliza para analizar vigas y pórticos indeterminados, se considera muy importante porque de ahí se derivan muchos métodos como las de rigideces y el método de Cros.

Se basa en la determinación de los desplazamientos de los nodos. Para lograrlo, es necesario establecer las ecuaciones de equilibrio de fuerzas en la dirección de los desplazamientos considerados en la estructura.

GRADOS DE LIBERTAD

Se denominan grados de libertad a Los desplazamientos desconocidos en los nodos de un elemento estructural cargado. En los métodos de desplazamiento los grados de libertad son iguales a las incógnitas del sistema, indicarán las ecuaciones de compatibilidad necesarias. Ponerle nombre a los soportes y nudos para poder identificar los claros de la barra entre los nodos, dibujando así la deformada de la barra, así logramos identificar el número de grados de libertad actuantes en la barra. Es posible que cada nodo tenga un desplazamiento angular y un desplazamiento lineal.

Ponerle nombre a los soportes y nudos para poder identificar los claros de la barra entre los nodos, dibujando así la deformada de la barra, de esta manera logramos identificar el número de grados de libertad actuantes en la barra. Es posible que cada nodo tenga un desplazamiento angular y un desplazamiento lineal.

ECUACIONES FUNDAMENTALES DEL METODO

El método de pendiente deflexión cuenta con dos fórmulas básicas:

$$M_{AB} = \frac{2EI}{L}\left(2\theta_A + \theta_B - \frac{3\Delta}{L}\right) + MEP$$

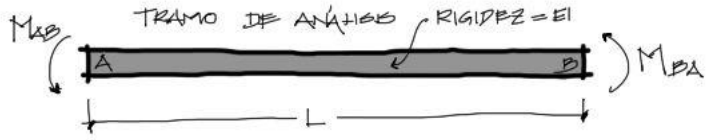
$$M_{BA} = \frac{2EI}{L}\left(\theta_A + 2\theta_B - \frac{3\Delta}{L}\right) + MEP$$

A y B son los dos extremos de un tramo de la viga entre dos apoyos. Como las vigas hiperestáticas tienen más de un tramo, en una viga de dos tramos se tendrán por ejemplo la dupla de momentos (Mab – Mba) y para el siguiente tramo la dupla de momentos (Mbc – Mcb).

CONVENCIÓN DE SIGNOS

La convención de signos para este método suele ser uno de los aspectos de mayor conflicto a la hora de aplicar las ecuaciones de arriba

Con respecto a los momentos en los extremos Mab y Mba, estos llevarán signo positivo si rotan en el sentido antihorario. Inicialmente, cuando no se conocen los valores de estos momentos, se asume que giran en el sentido positivo.

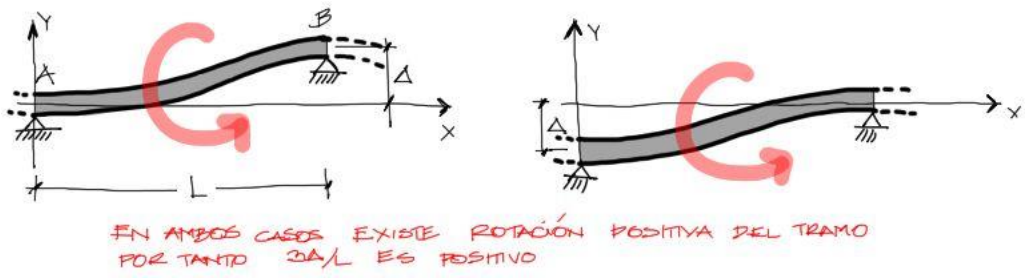


Cuando se trata de giros, estos se miden en radianes [rad]. En las ecuaciones de pendiente deflexión se cuenta con los giros en los extremos de cada tramo. Estos giros al igual que los momentos flectores, se consideran positivos si giran en el sentido antihorario.

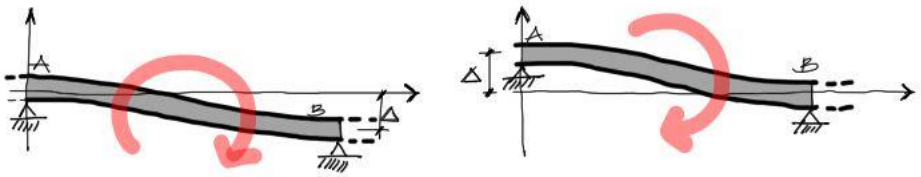
Tal vez el elemento que trae más confusión respecta al asentamiento Δ. Esto se debe a que por un lado la fórmula de pendiente deflexión ya lleva el signo negativo. Por otro lado, el signo de Δ *NO* depende directamente de si el asentamiento va hacia arriba o hacia abajo.

La manera de analizar el signo de la expresión 3Δ/L es la siguiente. Se debe analizar el tramo en su conjunto. No importa si el asentamiento se genera en el tramo izquierdo o en el tramo derecho, lo que importa es si el asentamiento en cualquiera de los dos extremos produce un «giro» global del tramo en sentido horario o antihorario.

Por ejemplo, la expresión 3Δ/L será positiva en cualquiera de las dos siguientes situaciones, para el tramo AB:



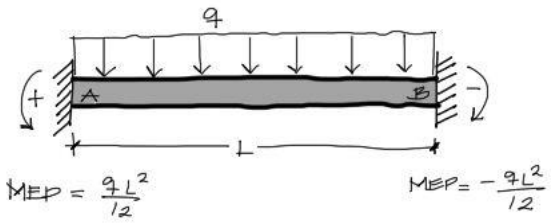
Por el contrario, esta expresión será negativa cuando se de cualquiera de las siguientes situaciones de asentamiento:



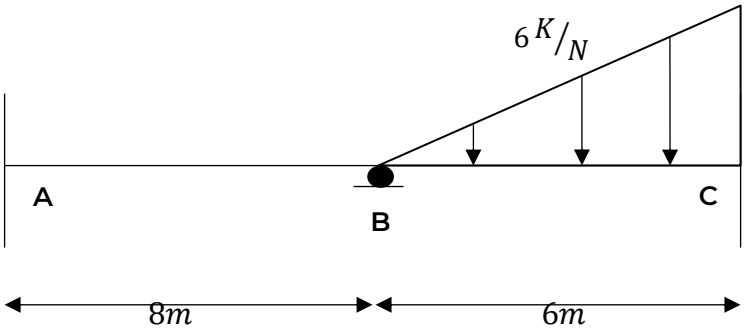
Finalmente, la convención de signos de los momentos de empotramiento perfecto MEP es igual al de los momentos Mab y Mba, positivo antihorario y negativo horario. Sin embargo, suele suceder casi siempre que, bajo cargas convencionales, el momento de empotramiento perfecto izquierdo es antihorario y el derecho horario. Esto se debe a que las cargas

convencionales verticales hacia abajo generan una reacción de momento en los extremos que describen ese comportamiento:

Por ejemplo, en la figura, se ve el tipo más común de carga y MEP.



PROBLEMA DE EJEMPLO



EI $= Constante$

Está empotrado en ambos lados.

- **PRIMER PASO: DEFINIR LOS FEM**

$$BC = -\frac{WL^2}{30} \qquad CB = \frac{WL^2}{20}$$

$$BC = -\frac{6\,(6)^2}{30} \qquad CB = \frac{6(6)^2}{20}$$

$$BC = -7.2\ KN \cdot m \qquad CB = 10.8\ \text{KN} \cdot m$$

***NOTA:**

No se analizan los tramos AB y BA debido a que no hay ninguna fuerza y se asume que el resultado será 0.
En este caso solo se analiza el tramo BC Y CB, para así obtener FEM_{BC} y FEM_{CB}.

- **SEGUNDO PASO: ANÁLISIS DE MOMENTOS POR TRAMO**

$$M_{AB} = 2\,\frac{EI}{8}(0 + \theta_B - 0) + 0 \qquad M_{BA} = 2\,\frac{EI}{8}(2\theta_B + 0 - 0) + 0$$

$$M_{AB} = \frac{EI}{4}\theta_B \qquad M_{BA} = \frac{EI}{2}\theta_B$$

$$M_{BC} = 2\,\frac{EI}{6}(2\theta_B + 0 - 0) + 0 \qquad M_{CB} = 2\,\frac{EI}{6}(0 + \theta_B - 0) + 0$$

$$M_{BC} = 2\frac{EI}{3}\theta_B - 7.2\ \text{KN} \cdot m \qquad M_{CB} = \frac{EI}{3}\theta_B + 10.8\ \text{KN} \cdot m$$

***NOTA:**

La fórmula que se utiliza para este paso es la siguiente:

$$\boldsymbol{M_{NF} = 2\,\frac{EI}{L}\left(2\theta_N + \theta_F - \frac{3\Delta}{L}\right) + FEM_N}$$

- **TERCER PASO: SISTEMA DE ECUACIONES**

$$M_{BA} + M_{BC} = 0$$

$$\frac{EI}{2}\theta_B + 2\frac{EI}{3}\theta_B - 7.2\ KN \cdot m = 0$$

$$\theta_B = \frac{6.17}{EI}$$

***NOTA:**

Se realiza un sistema de ecuaciones con Los resultados de los análisis de momentos por tramo.

Se tomo el BA y BC para realizar el sistema de ecuaciones.

- **CUARTO Y ÚLTIMO PASO: REEMPLAZAR Y OBTENER RESULTADOS FINALES**

$M_{AB} = 1.54\ \text{KN} \cdot m$ $\qquad$ $M_{BC} = -3.09\ \text{KN} \cdot m$

$M_{BA} = 3.09\ \text{KN} \cdot m$ $\qquad$ $M_{CB} = 12.86\ \text{KN} \cdot m$

¿QUÉ ES PYTHON?

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Python se clasifica constantemente como uno de los lenguajes de programación más populares.

HISTORIA DE CREACION DE PYTHON

Python fue creado a finales de los ochenta por Guido van Rossum en el Centro para las Matemáticas y la Informática, en los Países Bajos, como un sucesor del lenguaje de programación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba.

Además en este lanzamiento inicial aparecía un sistema de módulos adoptado de Modula-3; van Rossum describe el módulo como «una de las mayores unidades de programación de Python». El modelo de excepciones en Python es parecido al de Modula-3, con la adición de una cláusula else. En el año 1994 se formó "comp.lang.Python", el foro de discusión principal de Python, marcando un hito en el crecimiento del grupo de usuarios de este lenguaje.

CARACTERISTICAS

Python es un lenguaje de programación multiparadigma. Esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos:

programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Python usa tipado dinámico y conteo de referencias para la administración de memoria.

Una característica importante de Python es la resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado enlace dinámico de métodos).

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

Aunque la programación en Python podría considerarse en algunas situaciones hostil a la programación funcional tradicional del Lisp, existen bastantes analogías entre Python y los lenguajes minimalistas de la familia Lisp como puede ser Scheme.

El 20 de febrero de 1991, van Rossum publicó el código por primera vez en alt.sources, con el número de versión 0.9.0.9 En esta etapa del desarrollo ya estaban presentes clases con herencia, manejo de excepciones, funciones y los tipos modulares, como: str, list, dict, entre otros.

LA PROGRAMACIÓN ORIENTADA HA OBJETOS

Es un paradigma de programación que viene a innovar la forma de obtener resultados. Los objetos se utilizan como metáfora para emular las entidades reales del negocio a modelar.

LA PROGRAMACIÓN FUNCIONAL

Es el más usado en general, se basa en dar instrucciones al ordenador de como hacer las cosas en forma de algoritmos, en lugar de describir el problema o la solución, son también conceptos familiares similares en estilo a la programación imperativa; donde cada paso es una instrucción. Es la forma de programación más usada y la más antigua, el ejemplo principal es el lenguaje de máquina.

PROGRAMACIÓN IMPERATIVA

Es un paradigma de programación declarativa basado en el uso de verdaderas funciones matemáticas. En este estilo de programación las funciones son ciudadanas de primera clase, porque sus expresiones pueden ser asignadas a variables como se haría con cualquier otro valor; además de que pueden crearse funciones de orden superior.

ANÁLISIS CON PROGRAMAS ESTRUCTURALES

El método de los elementos finitos (MEF), también conocido como Finite Element Method (FEM) en inglés, es un procedimiento numérico pensado para su uso en software informático. En el ámbito de la ingeniería y la construcción es muy utilizado para el cálculo de resistencia de materiales y estructuras.

La dinámica de funcionamiento de la simulación FEM es dividir un problema, a priori muy complejo, en un número limitado de partes más pequeñas más sencillas de analizar y así poder encontrar una solución global.

EJEMPLOS

SIMSCALE: Es una potente herramienta FEM que puedes utilizar en tu navegador, sin necesidad de instalar ningún software. Todo lo que necesita hacer es cargar el modelo y ejecutar el análisis.

El componente de Análisis de Elementos Finitos de SimScale le permite probar y predecir el comportamiento de las estructuras. Esta herramienta le ayuda a resolver problemas complejos de ingeniería estructural sujetos a condiciones de carga estáticas y dinámicas.

LA HERRAMIENTA SOPORTA LOS SIGUIENTES MÓDULOS:

1. **Análisis estático:** Realiza simulaciones de estructura, incluyendo análisis estático lineal y cuasi estático no lineal. Este módulo es muy útil para la simulación de maquinaria pesada, equipos industriales, estructura de aeronaves, tuberías y diseño de puentes.
2. **Análisis Dinámico:** Analiza la respuesta dinámica de una estructura sometida a varias cargas y desplazamientos para calcular las cargas de impacto y la degradación estructural. Este módulo es muy útil para el análisis de automóviles, efectos sísmicos en edificios, y más.

3. **Análisis modal:** Determina las frecuencias y los modos propios de una estructura como resultado de la vibración libre. Puede utilizar este módulo para analizar la respuesta de pico de edificios, puentes o piezas de vehículos bajo varias frecuencias.
4. **Dinámica de Multicuerpos y Restricciones de Contacto:** Analizar el comportamiento de estructuras bajo grandes fricciones de deslizamiento y restricciones de contacto sin fricción.
5. **Modelos de materiales:** La herramienta de mecánica de sólidos ofrece varios modelos de materiales, permitiéndole analizar los efectos de la plasticidad, así como el comportamiento de grandes deformaciones usando modelos de materiales hiperelásticos

SIMCENTER 3D: Es una plataforma de simulación 3D avanzada que puede utilizar para modelar, así como para simular y analizar varios sistemas. La herramienta puede ser utilizada en una variedad de campos desde la etapa de diseño de un producto o sistema. En realidad, esta es una de las principales ventajas de Simcenter 3D: se puede utilizar desde la fase de diseño hasta la fase de pruebas.

Puede utilizar Simcenter 3D para los siguientes tipos de análisis: análisis estructural, análisis acústico, análisis de compuestos, análisis térmico, simulación de flujo, análisis de movimiento, multifísica, optimización de ingeniería.

SAP 2000: Programa de elementos finitos que cuenta con una interfaz gráfica 3D orientada a objetos. Debido a su fiabilidad, poder de cálculo y versatilidad, se utiliza para dimensionar puentes, presas, edificios y todo tipo de infraestructuras.

Cabe destacar también la amplia selección de plantillas con las que dispone, la generación de mallas de cálculo automáticas o la facilidad en la definición de vistas personalizadas.

Autodesk Robot Structural Analysis Professional: Con el respaldo de una compañía especializada como es Autodesk, está considerado uno de los más completos del mercado. Cuenta con tecnología MEF y es capaz de calcular las uniones de acero, estructuras de madera o secciones de armado, entre otras cosas. Structuralia este año se convirtió en Authorized Training Center de Autodesk, lo que supone un valor añadido para nuestra escuela.

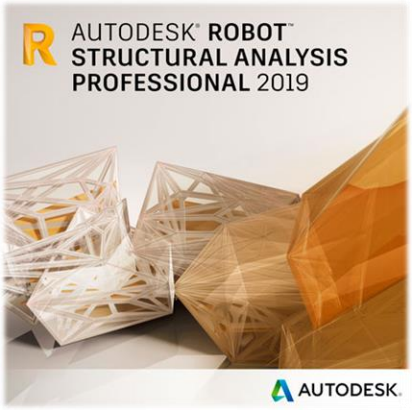
Con el uso cada vez más extendido de la metodología Building Information Modeling hay que destacar también su gran interoperabilidad con Revit, uno de los programas BIM más utilizados, mejorando de forma notable los flujos de trabajo.

midas Gen es un software para análisis y diseño de edificaciones y estructuras generales. Tiene una interfaz de usuario intuitiva, gráficos modernos y un potente algoritmo de solución.

El programa incluye una amplia gama de análisis y opciones de diseño que permiten modelar de manera eficiente todo tipo de estructuras.

AUTODESK ROBOT STRUCTURAL ANALYSIS PROFESSIONAL: Con el respaldo de una compañía especializada como es Autodesk, está considerado uno de los más completos del mercado. Cuenta con tecnología MEF y es capaz de calcular las uniones de acero, estructuras de madera o secciones de armado, entre otras cosas. Structuralia este año se convirtió en Authorized Training Center de Autodesk, lo que supone un valor añadido para nuestra escuela.

Con el uso cada vez más extendido de la metodología Building Information Modeling hay que destacar también su gran interoperabilidad con Revit, uno de los programas BIM más utilizados, mejorando de forma notable los flujos de trabajo.

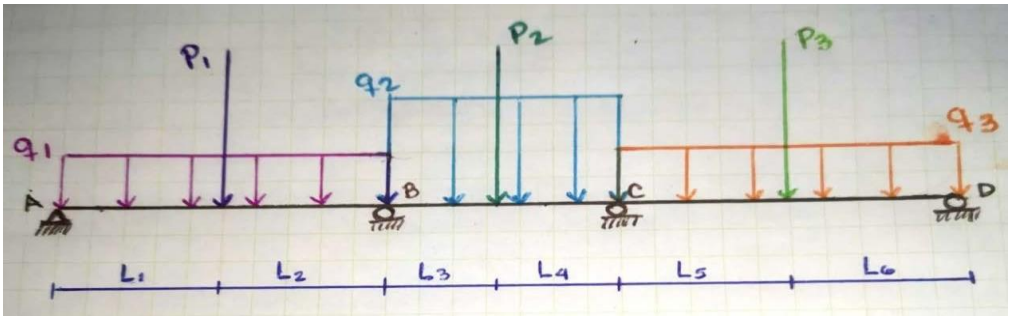


PYTHON

Metodo de pendiente deflexión

Luis Herrera
Zaharanis Gozaine
Alexis Cubilla
Joel Perez
Edwin Pimentel

CASO #1:



DATOS:

Longitudes $L_1 = 4$

m

$L_2 = 4$ m

$L_3 = 2$m

$L_4 = 2$m

$L_5 = 4$ m

$L_6 = 4$ m

Cargas Distribuidas $q_1 =$

150 N/m $q_2 = 350$ N/m

$q_3 = 150$ N/m

Cargas Puntuales

$P_1 = 45$ N

$P_2 = 78$ N

$P_3 = 94$ N

Otros

$E = 200$ N/m^2

$I = 164$ m^4

DESARROLLO EN PYTHON:

```
import sympy as sympy import matplotlib.pyplot as plt
x=sympy.symbols('x') print(23*'#'+'\n# PENDIENTE DEFLEXION
#\n'+23*'#') caso=int(input('CASOS\n\t1:CASO 1\n\t2:CASO
2\n\tCASO = ')) if caso==1:
   l1=float(input('L1 = '))    l2=float(input('L2 = '))    l3=float(input('L3 = '))
l4=float(input('L4 = '))    l5=float(input('L5 = '))    l6=float(input('L6 = '))
q1=float(input('q1 = '))    q2=float(input('q2 = '))    q3=float(input('q3 = '))
p1=float(input('P1 = '))    p2=float(input('P2 = '))    p3=float(input('P3 = '))
e=float(input('E = '))    i=float(input('I = '))    a_a=sympy.symbols('a_a')
a_b=sympy.symbols('a_b')    a_c=sympy.symbols('a_c')    a_d=sympy.symbols('a_d')
mab=2*e*i*(2*a_a+a_b)/(l1+l2)+(-q1*(l1+l2)**2/12-p1*l2**2*l1/(l1+l2)**2)
mba=2*e*i*(2*a_b+a_a)/(l1+l2)+(q1*(l1+l2)**2/12+p1*l2*l1**2/(l1+l2)**2)
mbc=2*e*i*(2*a_b+a_c)/(l3+l4)+(-q2*(l3+l4)**2/12-p2*l4**2*l3/(l3+l4)**2)
mcb=2*e*i*(2*a_c+a_b)/(l3+l4)+(q2*(l3+l4)**2/12+p2*l3**2*l4/(l3+l4)**2)
mcd=2*e*i*(2*a_c+a_d)/(l5+l6)+(-q3*(l5+l6)**2/12-p3*l6**2*l5/(l5+l6)**2)
mdc=2*e*i*(2*a_d+a_c)/(l5+l6)+(q3*(l5+l6)**2/12+p3*l5**2*l6/(l5+l6)**2)

   #ECUACIONES    ec1=mba+mbc    ec2=mcb+mcd    ec3=mab    ec4=mdc
angulos=sympy.solve([ec1,ec2,ec3,ec4],[a_a,a_b,a_c,a_d])
vba=1/(l1+l2)*(q1/2*(l1+l2)**2+p1*l1+mba.subs([(a_b,angulos[a_b]),(a_a,angulos[a_a])]))

vcb=1/(l3+l4)*(mbc.subs([(a_b,angulos[a_b]),(a_c,angulos[a_c])])+p2*l3+q2*(l3+l4)**2/2+mcb.s
ubs([(a_c,angulos[a_c]),(a_b,angulos[a_b])]))    vbc=-vcb+q2*(l3+l4)+p2
vdc=1/(l5+l6)*(mcd.subs([(a_c,angulos[a_c]),(a_d,angulos[a_d])])+q3/2*(l5+l6)**2+p3*l5)    vcd=-
vdc+q3*(l5+l6)+p3    vab=-vba+p1+q1*(l1+l2)    ra=vab    rb=vba+vbc    rc=vcb+vcd    rd=vdc
m1=ra*x-q1*x**2/2    v1=ra-q1*x    v2=ra-q1*x-p1    m2=ra*x-q1*x**2/2-p1*(x-l1)    v3=ra-
q1*(l1+l2)-p1-q2*(x-l1-l2)+rb    m3=-q2*(x-l1-l2)**2/2+rb*(x-l1-l2)-p1*(x-l1)+ra*x-q1*(l1+l2)*(x-
(l1+l2)/2)    v4=ra+rb-p1-q1*(l1+l2)-p2-q2*(x-l1-l2)    m4=ra*x+rb*(x-l1-l2)-p1*(x-l1)-
q1*(l1+l2)*(x-(l1+l2)/2)-q2*(x-l1-l2)**2/2-p2*(x-l1-l2-l3)    v5=-rd+q3*(l1+l2+l3+l4+l5+l6-x)+p3
```

```
m5=-q3*(l1+l2+l3+l4+l5+l6-x)**2/2-p3*(l1+l2+l3+l4+l5-x)+rd*(l1+l2+l3+l4+l5+l6-x)     v6=-
rd+q3*(l1+l2+l3+l4+l5+l6-x)     m6=-q3*(l1+l2+l3+l4+l5+l6-x)**2/2+rd*(l1+l2+l3+l4+l5+l6-x)

   x1=[]
m=[]     v=[]
i=0
   #p_d=float(input('POSICION DE LA DEFORMADA = '))
while i<=l1+l2+l3+l4+l5+l6:         if i<=l1:
          x1.append(i)
          m.append(m1.subs(x,i))
          v.append(v1.subs(x,i))         elif
l1<i<=l1+l2:             x1.append(i)
          m.append(m2.subs(x,i))
          v.append(v2.subs(x,i))         elif
l1+l2<i<=l1+l2+l3:
x1.append(i)
          m.append(m3.subs(x,i))
          v.append(v3.subs(x,i))         elif
l1+l2+l3<i<=l1+l2+l3+l4:
x1.append(i)
          m.append(m4.subs(x,i))
          v.append(v4.subs(x,i))         elif
l1+l2+l3+l4<i<=l1+l2+l3+l4+l5:
          x1.append(i)
          m.append(m5.subs(x,i))
          v.append(v5.subs(x,i))         elif
l1+l2+l3+l4+l5<i<=l1+l2+l3+l4+l5+l6:
          x1.append(i)
          m.append(m6.subs(x,i))
          v.append(v6.subs(x,i))
       i+=0.1
```

```
   print('\nREACCIONES\n\tRA = {:.2f}\n\tRB = {:.2f}\n\tRC = {:.2f}\n\tRD =
{:.2f}'.format(ra,rb,rc,rd))
   print('\nANGULOS\n\t\u03b8A = {:.8f} RAD\n\t\u03b8B = {:.8f} RAD\n\t\u03b8C = {:.8f}
RAD\n\t\u03b8D = {:.8f} RAD'.format(angulos[a_a],angulos[a_b],angulos[a_c],angulos[a_d]))
plt.plot(x1,v)    plt.title('CORTANTE')    plt.xlabel('X')    plt.ylabel('CORTANTE')
plt.grid(True,which='both')    plt.show()    plt.plot(x1,m)    plt.title('MOMENTO')    plt.xlabel('X')
   plt.ylabel('MOMENTO')
plt.grid(True,which='both')    plt.show()
```

RESULTADOS

REACCIONES:

RA = 505.92

RB = 1463.38

RC = 1524.62

RD = 523.08

ANGULOS:

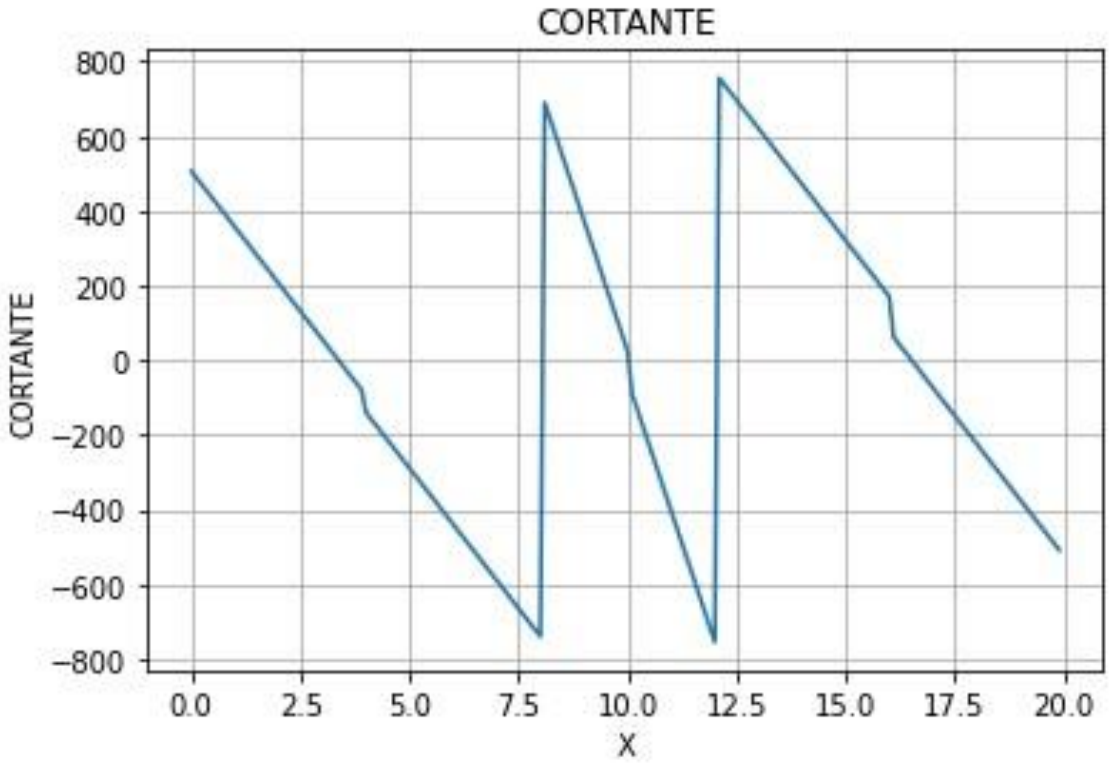
θA = 0.06513821 RAD

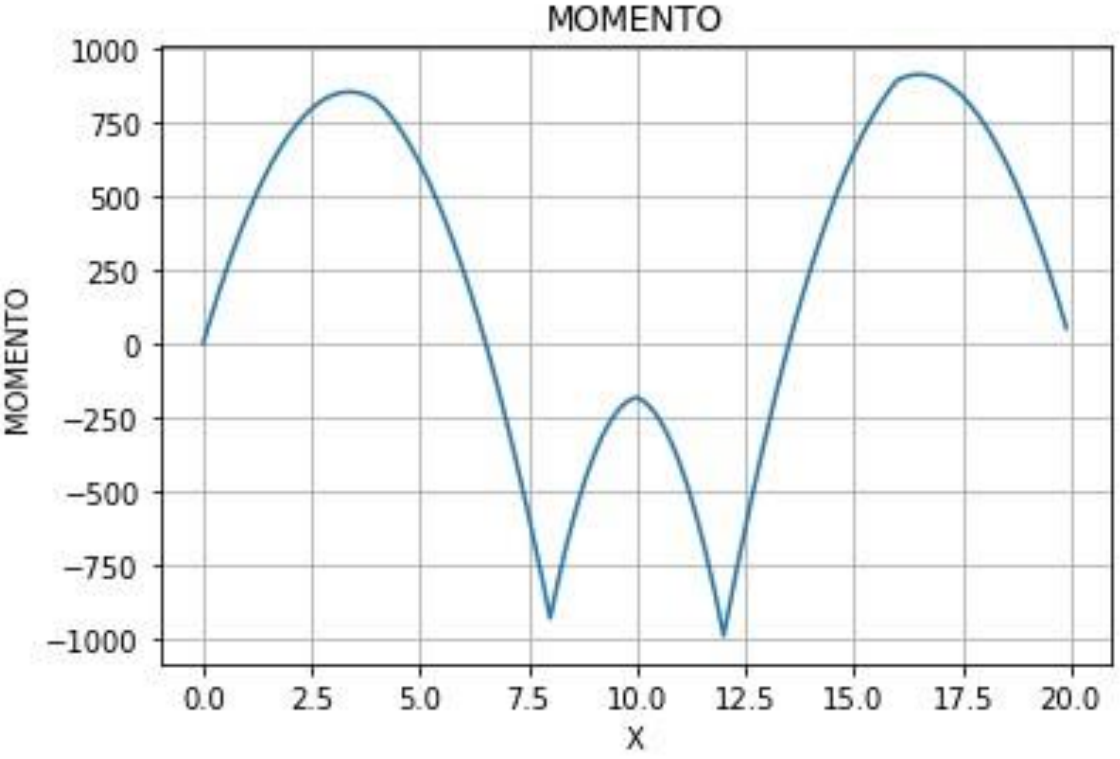
θB = -0.02722764 RAD

θC = 0.02842276 RAD

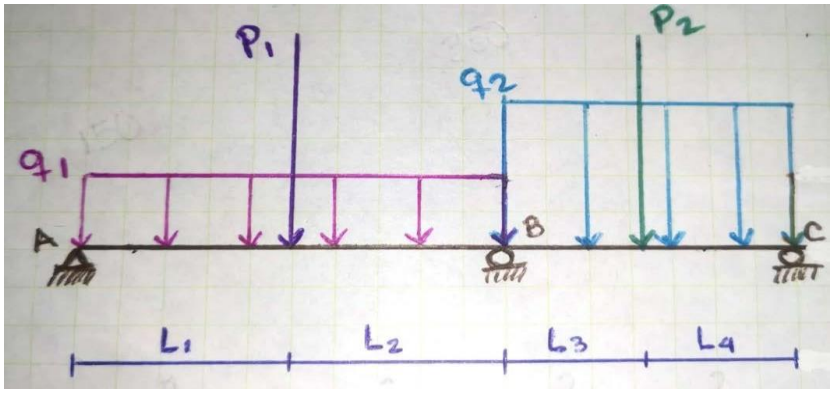
θD = -0.06872358 RAD

GRÁFICOS:


CORTANTE
800
600
400
200
0
−200
−400
−600
−800
CORTANTE
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X


MOMENTO
1000
750
500
250
0
−250
−500
−750
−1000
MOMENTO
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
X

CASO #2:



DATOS:

Longitudes

$L_1 = 4$ m

$L_2 = 4$ m

$L_3 = 2$m

$L_4 = 2$m

Cargas Distribuidas $q_1 =$

150 N/m

$q_2 = 350$ N/m

Cargas Puntuales

$P_1 = 45$ N

$P_2 = 78$ N

Otros

$E = 200$ N/m^2

$I = 164$ m^4

DESARROLLO EN PYTHON:

```
import sympy as sympy import matplotlib.pyplot as plt
x=sympy.symbols('x') print(23*'#'+'\n# PENDIENTE DEFLEXION
#\n'+23*'#') caso=int(input('CASOS\n\t1:CASO 1\n\t2:CASO
2\n\tCASO = ')) elif caso==2:
   a=sympy.symbols('a')    b=sympy.symbols('b')    c=sympy.symbols('c')
q1=float(input('q1 = '))    q2=float(input('q2 = '))    p1=float(input('p1 = '))
p2=float(input('p2 = '))    l1=float(input('L1 = '))    l2=float(input('L2 = '))
l3=float(input('L3 = '))    l4=float(input('L4 = '))    e=float(input('E = '))
i=float(input('I = '))    mab=2*e*i*(2*a+b)/(l1+l2)-q1*(l1+l2)**2/12-
p1*(l2)**2*l1/(l1+l2)**2
mba=2*e*i*(2*b+a)/(l1+l2)+q1*(l1+l2)**2/12+p1*(l2)*l1**2/(l1+l2)**2
mbc=2*e*i*(2*b+c)/(l3+l4)-q2*(l3+l4)**2/12-p2*l4**2*l3/(l3+l4)**2
mcb=2*e*i*(2*c+b)/(l3+l4)+q2*(l3+l4)**2/12+p2*l4*l3**2/(l3+l4)**2
ec1=mbc+mba    ec2=mcb    ec3=mab    angulos=sympy.solve([ec1,ec2,ec3],[a,b,c])
vba=(mba.subs([(a,angulos[a]),(b,angulos[b])])+p1*l1+q1*(l1+l2)**2/2)/(l1+l2)
vab=-vba+q1*(l1+l2)+p1
vcb=(mbc.subs([(b,angulos[b]),(c,angulos[c])])+p2*l3+q2*(l3+l4)**2/2)/(l3+l4)
vbc=q2*(l3+l4)+p2-vcb    rb=vba+vbc    rc=vcb    ra=vab    v1=-q1*x+ra
m1=ra*x-q1*x**2/2    v2=-q1*x-p1+ra    m2=-q1*x**2/2-p1*(x-l1)+ra*x
v3=p2+q2*(l1+l2+l3+l4-x)-rc    m3=-q2*(l1+l2+l3+l4-x)**2/2-p2*(l1+l2+l3-
x)+rc*(l1+l2+l3+l4-x)    v4=q2*(l1+l2+l3+l4-x)-rc    m4=-q2*(l1+l2+l3+l4-
x)**2/2+rc*(l1+l2+l3+l4-x)    print('\nREACCIONES\n\tRA = {:.2f}\n\tRB =
{:.2f}\n\tRC = {:.2f}'.format(ra,rb,rc))    print('\nANGULOS\n\t\u03b8A = {:.8f}
RAD\n\t\u03b8B = {:.8f} RAD\n\t\u03b8C = {:.2f}
RAD'.format(angulos[a],angulos[b],angulos[c]))    c=0    x_x=[]    v_x=[]    m_x=[]
while c<=l1+l2+l3+l4:        if c<=l1:
        x_x.append(c)
v_x.append(v1.subs(x,c))
```

```
m_x.append(m1.subs(x,c))        elif
l1<c<=l1+l2:            x_x.append(c)
v_x.append(v2.subs(x,c))
m_x.append(m2.subs(x,c))        elif
l1+l2<c<=l1+l2+l3:
x_x.append(c)
v_x.append(v3.subs(x,c))
m_x.append(m3.subs(x,c))        elif
l1+l2+l3<c<=l1+l2+l3+l4:
x_x.append(c)
v_x.append(v4.subs(x,c))
m_x.append(m4.subs(x,c))
c+=0.01     plt.plot(x_x,v_x)
plt.title('CORTANTE')     plt.xlabel('X')
plt.ylabel('CORTANTE')
plt.grid(True,which='both')     plt.show()
plt.plot(x_x,m_x)
plt.title('MOMENTO')     plt.xlabel('X')
   plt.ylabel('MOMENTO')
plt.grid(True,which='both')
   plt.show()
```

RESULTADOS

REACCIONES:

RA = 485.27

RB = 1773.19

RC = 464.54

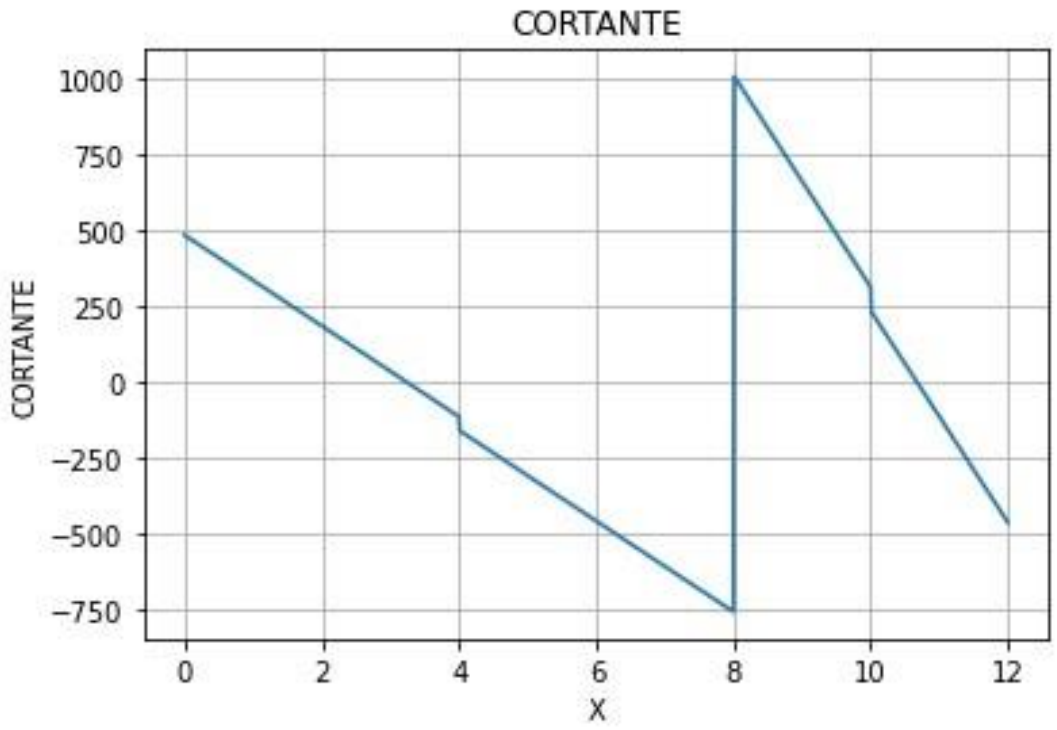
ANGULOS:

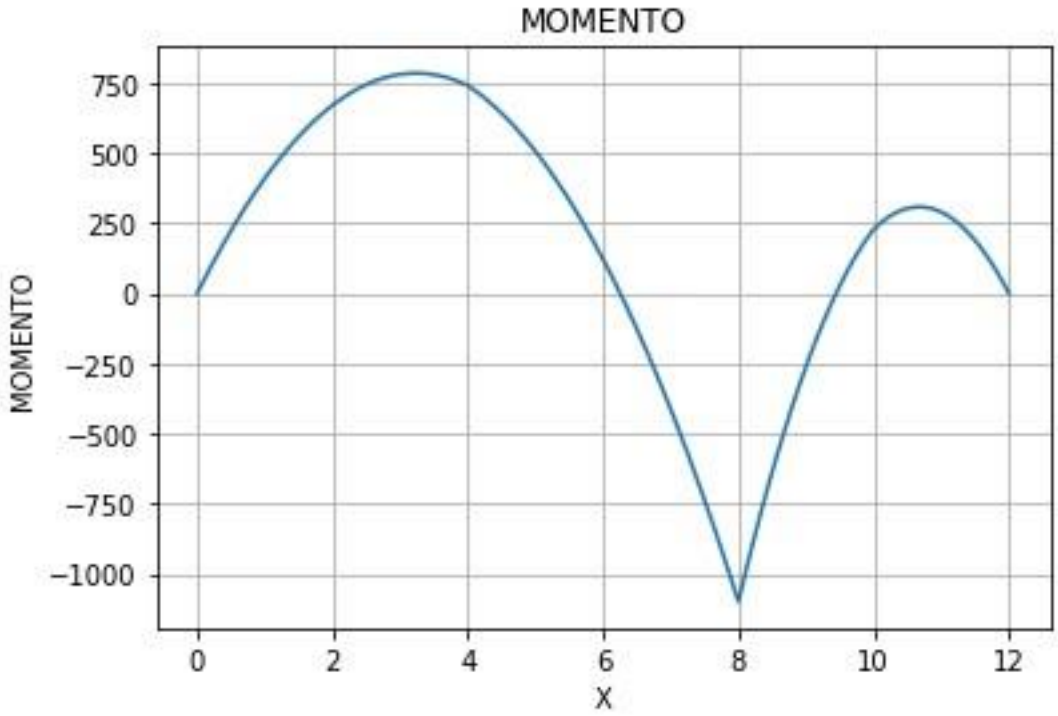
θA = 0.05842141 RAD

θB = -0.01379404 RAD

θC = -0.01 RAD

GRÁFICOS:


CORTANTE
1000
750
500
250
0
−250
−500
−750
CORTANTE
0
2
4
6
8
10
12
X


MOMENTO
750
500
250
0
−250
−500
−750
−1000
MOMENTO
0
2
4
6
8
10
12
X