Kakatiya Institute of Technology and Science, Warangal

Course Project Technical Report (2023-24)

Course: Engineering Mechanics – U18CE205

Project Code: CE205P2

Project Title: Analysis and Design of Simply Supported Beams using Python

Course Faculty:
Dr. N. Srikanth
Assistant Professor
Department of Civil Engineering

Done By:

Aryan Karamtoth
B23IT117

Contents

Introduction

Civil Engineering is one of the oldest branches of engineering in the world and has played a major role in developing the world. Civil Engineering is the most basic form of engineering which supports all of the other branches of engineering.

Civil Engineering provides infrastructure for Mechanical, Electrical, Computer and many other engineering fields so it is certain that Civil Engineering's role is irreplaceable and there'll be always a need of Civil Engineering and its applications.

Civil Engineering is a wide field and deals with various applications starting from planning to construction. A Civil Engineer needs to be well equipped with his knowledge and tools to make the best out of his skills.

The concepts of Civil Engineering are not just for Civil Engineers but almost every engineer needs to have some basic knowledge of Civil Engineering and how it works. It is our duty as an engineer to be well equipped with knowledge related to other fields.

As part of the course Engineering Mechanics offered by the Department of Civil Engineering at Kakatiya Institute of Technology and Science, I've been exposed to various basic concepts of Civil Engineering such as supports and beams.

The topic that we'll be focusing on in this project is Beams.

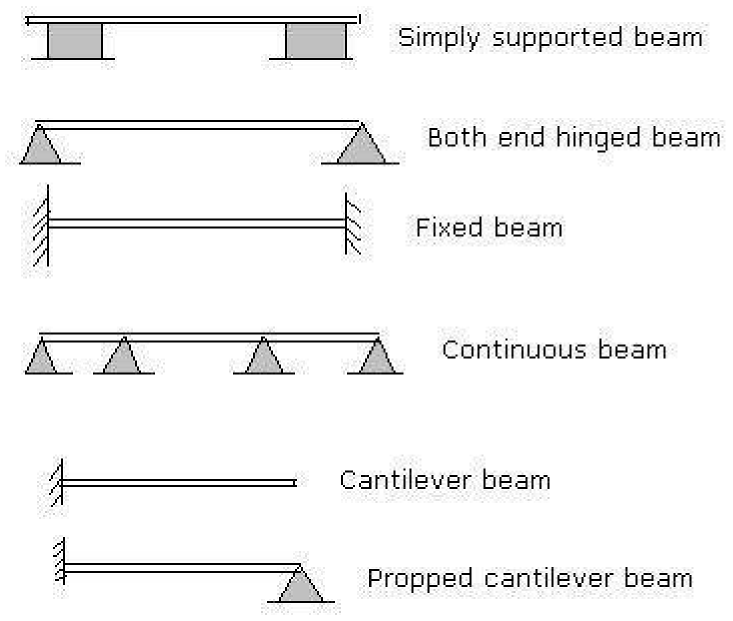
Beams are the structural elements that transfer load acting on the overall structure and transfer the load to the supports.

There are various types of Beams out there, namely:

(i) Simply Supported Beam

(ii) Cantilever Beam
(iii) Continuous Beam
(iv) Overhanging Beam
(v) Fixed Beam
(vi) And many more…

For this project, we will be mostly dealing with Simply Supported Beam, which is the most basic form of a Beam.



Loads
Loads are the force applied on the beam. It is of three types:

1. Point Load

2. Uniformly Distributed Load

3. Uniformly Varying Load.

We will be applying these on beams to calculate the support reactions.

Problem Definition

Beams are a very essential part of constructions hence they play a very important role in the design and maintain structural integrity of the construction.

Designing such beams needs a lot of calculations and measures. A lot of care needs to be taken to avoid any mistakes or imperfections while designing a beam as any mistake can make the structure collapse and may lead to loss of life and property.

A beam is meant to handle all of the load that's inserted on it and efficiently transfer it to the supports in order to maintain a stable structural integrity. Such beams can be designed only with proper calculations of support reactions and equilibrium equations.

In the course, Engineering Mechanics, we deal with basic types of beams. All of these beams are different from each other and are suitable for a specific purpose.

For example, Overhanging beams are used to handle structures like balconies and overhanging structures.

A cantilever beam is used for a similar purpose.

Each beam has its own purpose and needs to be designed for its specific use, hence it requires suitable calculations for the design and analysis.

In this project, we will be focusing on developing an algorithm using Python to make the design and analysis of beams a lot easier and accessible.

Proposed Method

About Python

Python is a general purpose programming language created by Guido Van Rossum in 1991. It has dynamical typing and garbage collection built into it, which makes it a very good choice for easy development of algorithms and applications without worrying too much about the complexity and the issues caused by memory.

Python finds its use in various fields especially the scientific world as well as in fields like data science, artificial intelligence and machine learning. It's wide software, platform and package support makes it a very good choice to import and export packages thereby making the work of the developer or the scientist a lot easier.

Even though there are languages like MATLAB which are more suitable for scientific and engineering purposes, licensing and the fact that MATLAB is a paid software makes it a bit less accessible to the common man and due to this reason, Python is slowly gaining popularity because it's free to use.

Python grows in popularity with each passing day and its ecosystem becomes bigger and bigger, making the libraries of it even better and more feature rich.

We are choosing Python for this project due to its simplicity and feature rich. Python provides a lot of tools for developer as well as for scientific purposes out of the box. Some of the most important libraries ,like NumPy, Pandas and others, are readily available and come preinstalled with the Python interpreter making it very easy to access these libraries and use them in our project.

Additionally, if we want other third party or community made libraries, we can easily install them using the built-in package manager called PIP (Preferred Installer Program). PIP has thousands of packages available to download with one single command and the python ecosystem keeps getting better.

Project Requirements

This project needs very few libraries due to it being limited to simply supported beams only. Hence, the project's setup will be quite simple so will be the implementation.

Language Used	Python 3.12
Libraries Required	NumPy
IDEs	PyCharm Professional
Version Control	Git

I will be using 3.12 version of Python which is the latest version of the language at the time of making this project.

I will be also using NumPy to make calculations and numerical operations much easier because NumPy has a lot of built in functions for numerical calculations which make our job easier and thereby saving us from the hassle of implementing our own functions for numerical calculations.

An IDE is a must to write code. IDEs also come with a lot of built in tools and help you arrange your files and package them easily for distribution. For this project, I chose PyCharm Professional which is a dedicated IDE for Python built by JetBrains.

Version Control is a must to manage the changes in the codebase and avoid any conflicts or merge errors. For this, I chose Git as the version control system and GitHub to manage the repository.

Approach

Now that we have defined our project requirements, let us have a look at the approach for the development of the project.

We will be creating some functions to make the code more reusable and more modular. These functions will take some arguments such as position of roller pins etc.

In this project, we will be considering a beam. The user has to enter the position of the supports and the algorithm will calculate the support reactions and return the value accordingly.

Creating a Beam

We will be considering a beam as an array of zeroes. This array will be created first in the beginning of the program.

0	0	0	0	0	0	0

Index:	0	1	2	3	4	5	6

The above array is a beam with the length of 7 metres.

Defining Pin and Roller Support Positions

Now, we will be using definePin() function to define the location of the Pin Support.

Let's say that we want to insert a pin support at 0th index. We just have the pass the value of the index into the function and the location of the index gets stored as pin support's location.

Same mechanism takes place for defining roller support.

Applying the Loads

Now that we have our beams set up, we need to apply the loads accordingly. We use 3 functions for this purpose; applyPointLoad(), applyUDL(), applyUVL().

Function	**Purpose**
applyPointLoad()	Apply point load on the beam
applyUDL()	Apply Uniformly Distributed Load on the beam
applyUVL()	Apply Uniformly Varying Load on the beam

We will be implementing similar array methods for using these.

Calculating Support Reactions

For calculating support reactions, we will be using a function named calcReactionsSSB(), which considers the case of types of loads and calculates the support reactions accordingly.

Algorithm

Step 1: Start

Step 2: Enter size of the beam

Step 3: Create the beam in memory

Step 4: Take inputs for Pin and Roller support positions and mark them in the memory

Step 5: Input the required data for support reactions

Step 6: Calculate Support Reactions

Step 7: Display Support Reactions

Step 8: Stop

Experiment

Let's create the necessary functions

createBeam()

```
def createBeam(len):
        beam = np.arange(len)
        return beam
```

definePin()

```
def definePin(pinPos,beam):
        pinposition = beam[pinPos]
        return pinposition
```

defineRoller()

```
def defineRoller(rollPos,beam):
        #assigning the roller position to the index of beam
        rollposition = beam[rollPos]
        return rollposition
```

calcReactionsSSB()

```
def calcReactionsSSB(loads, pinPos, rollPos, beam):
        # init the total reactions
        Ra_total = 0
        Rb_total = 0

        # calc length of beam
        L = len(beam)

        # using a loop to check all the load values
        for load in loads:
            #we are using a dictionary input for checking the type of load
            #in case of point load
            #----------------------------Point----------------------------------
--------
            if load['type'] == 'point':
                # calculating the distance from pin support
                a = abs(load['position'] - pinPos)
                # Calculate the reactions for this load
                Rb = (load['magnitude'] * (L - a)) / L  # Reaction at the roller
support

                Ra = (load['magnitude'] * a) / L  # Reaction at the pin support
                # Add the reactions for this load to the total reactions
                Ra_total += Ra
                Rb_total += Rb
            #---------------------------UDL-------------------------------------
---------
            elif load['type'] == 'udl':
                # For UDL, calculate the equivalent point load
                eqLoad = (load['end'] - load['start']) * load['magnitude']
                # in UDL, the CG is at the midpoint so we calculate the CG
                cg= (load['start'] + load['end']) / 2
                # Now calculate the reactions as for a point load
                a = abs(cg - pinPos)
                Rb = (eqLoad * (L - a)) / L
                Ra = (eqLoad * a) / L
                Ra_total += Ra
                Rb_total += Rb
            #--------------------------------UVL--------------------------------
-----------
            elif load['type'] == 'uvl':
                # For UVL, calculate the equivalent point load
                eqLoad = (load['end'] - load['start']) * (load['start_magnitude']
+ load['end_magnitude']) / 2
                # The position of the equivalent point load (CG) depends on the
magnitudes of the start and end loads
                if load['start_magnitude'] != load['end_magnitude']:
                    cg = load['start'] + (
                                2 * (load['end'] - load['start']) *
load['end_magnitude']) / (
                                                      load['start_magnitude'] +
load['end_magnitude'])
                else:
                    cg = (load['start'] + load['end']) / 2
                # Now calculate the reactions as for a point load
                a = abs(cg - pinPos)
                Rb = (eqLoad* (L - a)) / L
                Ra = (eqLoad * a) / L
                Ra_total += Ra
                Rb_total += Rb
            else:
                raise ValueError("Invalid load type. Expected 'point', 'udl', or
'uvl'.")

        return Ra_total, Rb_total
```

Applying it in a Program

```
from beamxs.beamxs import beamxs as bm

#createBeam() test
beam = bm.createBeam(10)
print(beam)

loads = [
    {'type': 'point', 'position': 2, 'magnitude': 50},
    {'type': 'udl', 'start': 3, 'end': 5, 'magnitude': 20},
    {'type': 'uvl', 'start': 6, 'end': 8, 'start_magnitude': 10, 'end_magnitude':
30}
]
pinPos = 1
rollPos = 9
Ra, Rb = bm.calcReactionsSSB(loads, pinPos, rollPos, beam)
print(Ra)
print(Rb)
```

Output

```
C:\Users\aryan\AppData\Local\Microsoft\WindowsApps\python3.12.exe
C:\Users\aryan\OneDrive\Documents\Github\beam-xs\main.py
[0 1 2 3 4 5 6 7 8 9]
49.0
```

```
81.0

Process finished with exit code 0
```

Result

From the experiment, we can observe that we got two values at the end which are the support reactions from both the ends.

We also got a list as an output which is the beam that's created and stored in the memory. Operations such as defining point support, roller support, and calculations of support reactions have been performed on this same beam.

This entire operation takes place in memory and is handled by the Python interpreter which is not possible in languages like C or C++, where the memory has to be managed by the user.

Support reactions can be used to analyse the effects of loads and supports on a beam and hence help the engineers design a beam accordingly.

Conclusion

We have successfully designed and implemented an algorithm for designing and analysing a simply supported beam using Python.

This algorithm can be used in complex programs and projects to quickly design and simulate beams along with loads and supports. This ensures that there is a robust design and care taken towards construction of beams.

Similar approach can be followed for other types of beams such as overhanging beam etc. but they may require more complex algorithms and libraries due to the dynamic nature of beams.

References

- StackOverFlow [https://stackoverflow.com]
- Google [https://google.com]
- Youtube [https://youtube.com]