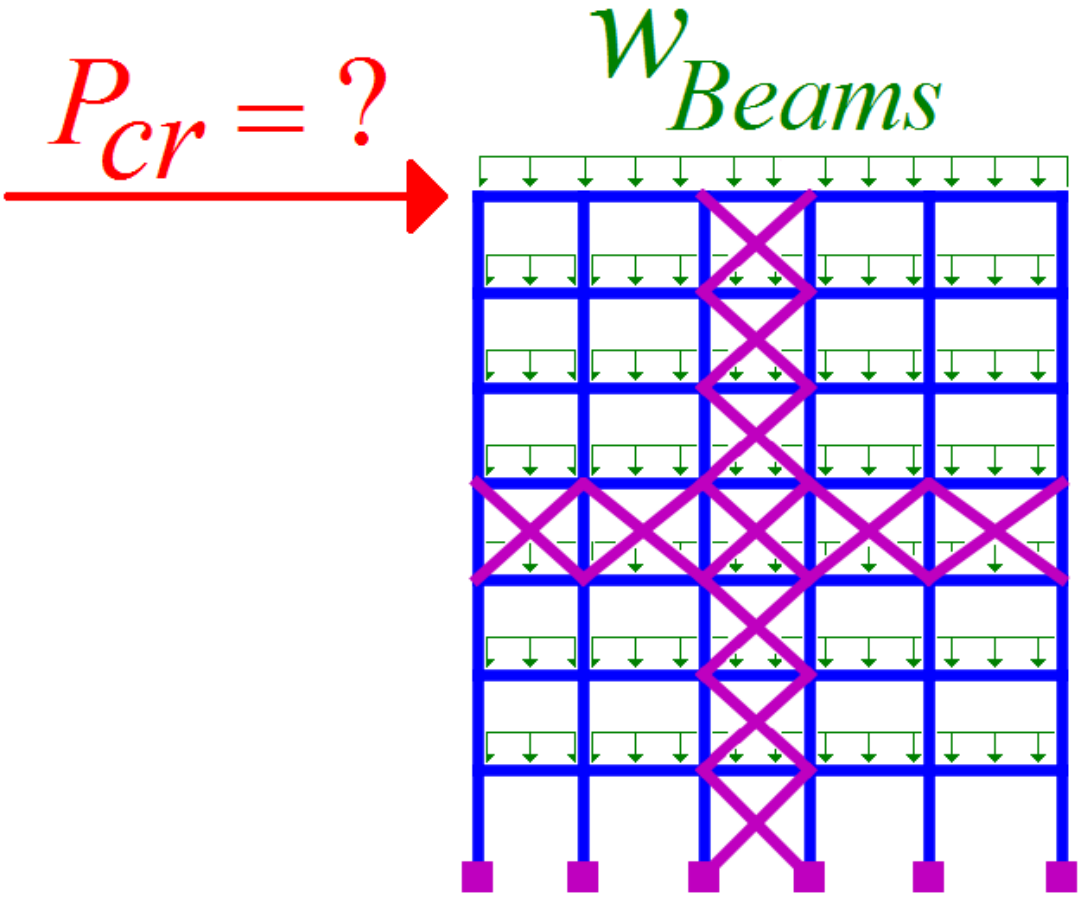
# "Buckling_Analysis"

**Calculating buckling-load of a six-story frame with core-and-outrigger with a distributed beam-load 0f 0.01 KN/mm**



In [1]:

```
# (auto) importing modules needed

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import copy

```

## Structure Data (input)

In [2]:

```
# (input)

nStory      = 25                                 # number of stories

hStory_typ  = 3600                              # height of typical stories
hStory_Base = 4000                              # height of base_floor

wBay = [  4000, 4500, 4000, 4500, 5000 ]        # width of bays ( any amount )

Braced_Bays    = [ 3  ]                         # Doubly (X) Braced  Bays
Braced_Stories = [ 4  ]                         # Doubly (X) Braced Storys

# sections:
# All Columns:    IPBv 220:  A=14900 (mm2)     I=146.0*10**6 (mm4)
# All Beams:      IPE  300:  A= 5380 (mm2)     I= 83.6*10**6 (mm4)
# All Diagonals:  UPE  120:  A= 1700 (mm2)

E_Cols, E_Beams, E_Diags = 200,200,200          # Modulus of Elasticity (KN/mm2)
A_Cols, A_Beams, A_Diags = 14900,5380,1700      # Area (mm2)
I_Cols, I_Beams          = 146e6,83.6e6         # Moment of Inertia (mm4)

W_Beams = 0.01                                  # rectangular Distributed Load on Beams (k
                                                # towards Ground is +ve
```

## Structure Data (auto)

In [3]:

```
# (auto) Checking & completing

hStory    = np.ones( (nStory,1) ) * hStory_typ
hStory[0] = hStory_Base

nBay = len(wBay)    # number of bays

Braced_Bays = sorted( Braced_Bays )
keep = []
for i in Braced_Bays:
    if i <= nBay: keep.append(i)
    else: print('\n Warning!',
                '\n Bay', i, "does not exist thus can't be braced.",
                i, 'is removed from Braced_Bays' )
Braced_Bays = sorted(keep)

Braced_Stories = sorted( Braced_Stories )
keep = []
for i in Braced_Stories:
    if i <= nStory: keep.append(i)
    else: print('\n Warning!',
                '\n Story', i, "does not exist thus can't be braced.",
                i, 'is removed from Braced_Stories' )
Braced_Stories = sorted(keep)
```

## Points & Connectivity Matrices

In [4]:

```
# (auto) Making Frame

# defining a function called PC
# to calculate Points & Connectivity matrices of Columns, beams and diagonals

def PC( wBay, hStory, Braced_Bays, Braced_Stories ):
    ...
    return [ Points, CnC, CnB, CnD ]

[ Points, CnC, CnB, CnD ] = PC( wBay, hStory, Braced_Bays, Braced_Stories )
```

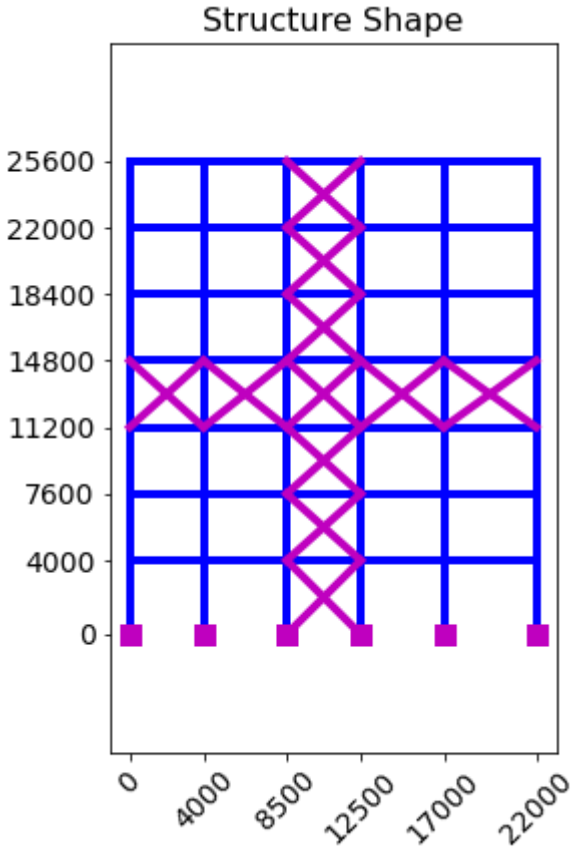
## Plotting Structure Shape

In [5]:

```
# define a function called Shape
# to check shape of frame

def Shape( Points, CnC, CnB, CnD, wBay, hStory ):
    ...
    return None

Shape( Points, CnC, CnB, CnD, wBay, hStory )
```



## Finding Static Coefficients

In [6]:

```
# (auto) Coefficients
# define a function called Static_Coeff
# to find Static Analysis Coefficients

def Static_Coeff( Points, CnC, CnB, CnD, wBay ):
    ...
    return [NOP, NOD, RD, FD, NFD, NC, NB, ND, IndxC, IndxB, IndxD]

[NOP, NOD, RD, FD, NFD, NC, NB, ND, IndxC, IndxB, IndxD] = Static_Coeff( Points,CnC,CnB
```

## Assigning

In [7]:

```
# (auto) define a function called Elements
# to asign material & shape Properties & distributed load
# to columns, beams and trusses

def Elements(
      NB, IndxB, E_Beams, A_Beams, I_Beams, W_Beams
    , NC, IndxC, E_Cols,  A_Cols,  I_Cols
    , ND, IndxD, E_Diags, A_Diags  ):
    ...
    return [Beams, Columns, Diagonals]

[ Beams, Columns, Diagonals ] = Elements( NB, IndxB, E_Beams, A_Beams, I_Beams, W_Beams
                                        , NC, IndxC, E_Cols,  A_Cols,  I_Cols
                                        , ND, IndxD, E_Diags, A_Diags )
```

## Nodal Forces

In [8]:

```
# define a function called Nodal_Forces to assign External Point load to top left of St

def Nodal_Forces( Roof_Load, NOD, nStory, nBay ):
    ...
    return NF

Roof_Load = 1
NF = Nodal_Forces( Roof_Load, NOD, nStory, nBay )
```

## Elastic Stiffness Matrices

In [9]:

```
# (auto) defining Stiffness Matrices

def ke_frame( A, E, I, L ):
    ...
    return df

def ke_truss( A, E, L ):
    ...
    return df

def kg_frame( P, L ):
    ...
    return df

def kg_truss( P, L ):
    ...
    return df

def T_frame( c, s ):
    ...
    return df.T

def T_truss( c, s ):
    ...
    return df.T
```

## Ke & Qf

In [10]:

```
# (auto) defining a function called KeQf
# to form:
#    Elastic Stiffness Matrices, Ke, for all elements
#    External Distributed Loads Matrix, Qf

def KeQf(  Points,  NOD
         , CnB, NB, IndxB, Beams
         , CnC, NC, IndxC, Columns
         , CnD, ND, IndxD, Diagonals ):
    ...
    return [ KE,QF, LC,TC,keC,KeC, LB,TB,keB,KeB,qfB, LD,TD,keD,KeD ]

[ KE,QF, LC,TC,keC,KeC, LB,TB,keB,KeB,qfB, LD,TD,keD,KeD ] = KeQf(
        Points,  NOD
      , CnB, NB, IndxB, Beams
      , CnC, NC, IndxC, Columns
      , CnD, ND, IndxD, Diagonals )
```

## P-Delta Analysis

In [11]:

```
# (auto) define a function called P-Delta
# Which Performs P-Delta Analysis on Frame
# and returng Global Geometry Matrix, KG
# and number of Iterations.

def P_Delta(
      NF,  NOD,  FD, KE, QF
    , NC, IndxC, TC, keC, KeC, Columns
    , NB, IndxB, TB, keB, KeB, Beams, qfB
    , ND, IndxD, TD, keD, KeD, Diagonals  ):
    ...
    ...
    ...
    return [ nIter, KG ]


# P-Delta Analysis (Checking)
[ nIter, KG ] = P_Delta(
      NF,  NOD,  FD, KE, QF
    , NC, IndxC, TC, keC, KeC, Columns
    , NB, IndxB, TB, keB, KeB, Beams, qfB
    , ND, IndxD, TD, keD, KeD, Diagonals  )
```

```
Iteration 1
Iteration 2
Iteration 3
Iteration 4
P-Delta Analysis Converged after 4 Iterations
```

## Buckling Analysis

In [12]:

```
# define a function called Buckling_Load
# which returns the Buckling Load

def Buckling_Load( KE, KG, FD ):
    ...
    return round( alpha[-2], -1 )

Pcr = Buckling_Load( KE, KG, FD )
print( '\nPcr = ',Pcr )
```

```
Ckecking P = 10
Iteration 1
Iteration 2
Iteration 3
Iteration 4
P-Delta Analysis Converged after 4 Iterations

Ckecking P = 100
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
P-Delta Analysis Converged after 5 Iterations

Ckecking P = 1000
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
P-Delta Analysis Converged after 6 Iterations

Ckecking P = 10000
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
P-Delta Analysis Converged after 10 Iterations

Ckecking P = 100000
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
```

```
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
Iteration 19
Iteration 20
Iteration 21

 Warning!
 No Convergence after 20 iterations.

Ckecking P = 20000
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
P-Delta Analysis Converged after 14 Iterations

Ckecking P = 40000
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
Iteration 19
Iteration 20
Iteration 21

 Warning!
 No Convergence after 20 iterations.

Ckecking P = 22000.0
Iteration 1
```

```
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
P-Delta Analysis Converged after 14 Iterations

Ckecking P = 24200.000000000004
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
P-Delta Analysis Converged after 15 Iterations

Ckecking P = 26620.000000000007
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
P-Delta Analysis Converged after 16 Iterations

Ckecking P = 29282.00000000001
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
```

```
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
P-Delta Analysis Converged after 16 Iterations

Ckecking P = 32210.200000000015
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
P-Delta Analysis Converged after 18 Iterations

Ckecking P = 35431.22000000002
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
P-Delta Analysis Converged after 18 Iterations

Ckecking P = 38974.342000000026
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
```

```
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
Iteration 19
Iteration 20
Iteration 21

 Warning!
 No Convergence after 20 iterations.

Ckecking P = 37202.781000000025
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
Iteration 19
P-Delta Analysis Converged after 19 Iterations

Ckecking P = 39062.92005000003
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
Iteration 19
Iteration 20
Iteration 21
```

```
 Warning!
 No Convergence after 20 iterations.

Pcr =  37200.0
```

## Result:

Buckling Load, Pcr = 37200 (KN)