The Asian Review of Civil Engineering
ISSN: 2249-6203 (P) Vol.11 No.2, 2022, pp.15-24
© The Research Publication, www.trp.org.in
DOI: https://doi.org/10.51983/tarce-2022.11.2.3494

Comparative Study of Wind Pressure Variations on Rectangular Buildings Using Python Programming

Pankaj Dumka[1], Nitin Kumar Samaiya[2], Sumit Gandhi[3], Subas Dash[4], Sumit Dubey[5] and Dhananjay R. Mishra[6]
[1,4&6]Department of Mechanical Engineering, [2&3]Department of Civil Engineering, [5]Wind Engineering Application Centre,
Jaypee University of Engineering and Technology, Guna, Madhya Pradesh, India
E-mail: p.dumka.ipec@gmail.com, nitin.samaiya@juetguna.in, sumit.gandh@juetguna.in, subas.dash@juetguna.in, dm30680@gmail.com

Abstract - **In this research article an attempt has been made to analyse the design wind pressure on the rectangular buildings using Python programming. For this purpose, design wind pressure is calculated and compared using IS 875 (Part 3) 1987 and the revised code IS 875 (Part 3) 2015. The concept have been applied to three different building heights i.e., 20, 40, and 60 m having same plan 10×10 m^2. It has been observed that the impact of wind pressure on the building rises with the building height along with the fact that the design wind pressures obtained from revised code show more pressures in comparison to the old one. Therefore, the design based on the revised code will be more close to the reality as it incorporates risk factor, directionality factor, area averaging factor, and combination factors. Also, the module developed for wind loads can be readily used by the researchers/designers for the better understanding of programming and the design loads.**
Keywords: **Design Wind Pressure, Design Wind Velocity, Python Programming, Terrain Simulation, IS 875 (Part 3)**

I. INTRODUCTION

Making some consideration for wind forces is now a widely accepted design principle for practically all above-ground buildings [1], [2]. It may be a significant factor affecting the initial cost and ultimately the inherent safety of the project for structures that are particularly exposed, such as skyscrapers, tall poles, long-span bridges, radio telescopes, etc. [3]. Historically, these pressures were computed under the presumption that it was safe to ignore the variations in velocity caused by gusts and to treat the velocity as being constant throughout both time and space[4], [5]. This simplification was useful since pressures could then be estimated from straightforward wind tunnel tests on models in a steady airstream [6], [7].

These investigations led to the introduction of specific aerodynamic coefficients, such as the pressure and drag coefficients, which were considered to be applicable to both the model and its prototype [8]. The only additional data required to determine the pressures was an acceptable design wind velocity [9]. When it comes to the analytical modelling of wind pressure several country have their own code of practice for design of buildings [10]. But more or less each code gives the same information in different system of units. In order to evaluate the design wind pressure following methodology is adopted.

1. Select the location to find the base wind speed.
2. Modify the base wind speed to get the design wind speed.
3. From the design wind speed to evaluate the design wind pressure.

The above mentioned tasks require several intermediate steps which involves the evaluation of many influencing factors. The range and values of factors are evaluated based on the standard codes of a country. In India, for wind load IS 875 (Part 3) [11]-[14] is adopted as the code for structural design considering the wind effects. The first revision of the above code was adopted in year 1987 which was further revised in year 2015[15], [16]. Determining wind pressure manually takes time and includes errors moreover, when the tables become hefty it becomes sometime very difficult to interpolate several data point with accuracy and precision. The errors in the designs can lead to false prediction of results which can be avoided if we use the computer to perform the computation.

Keeping all above factors in view, Python programming is being applied for automating the process of charting and analysing load computations, as it provides platforms for sophisticated and scientific computation[17]-[20]. The actual advantages of Python are its extensive community and comprehensive library. Some of the libraries that may do numerical, scientific, and symbolic computations as well as data visualisation and interpretation are NumPy [21]-[24], SciPy [21], [22], [25], SymPy [26], [27], and Matplotlib [28], [29]. With a very small number of lines of code, the modules are extremely capable of carrying out any mathematical computation, including those using algebra, trigonometry, calculus, set theory, etc. Even though Python is a computer tool, academicians are now adopting it to automate research methods in all areas of science.

In this article, the guidelines of both old and the revised IS codes IS 875 (Part 3) have been followed for design wind speed; and pressures have been compared for three different building. The Python module thus developed will be useful to understand the effects of different factors which influences the wind pressure and velocity based on modifications done in refereed IS Code.

II. THEORY

For the comparison of both the IS codes three different rectangular building structures have been considered (Table I). In order to fulfil the objective of present study one of the geometrical parameters (i.e., the height) of the building is varied whereas, the other parameters (i.e., length and width) of the building are kept constant. The location of the building is considered at Bhopal (23° 15' 35.7588" N and 77° 24' 45.4068" E), India. The following are the other technical specifications:

Life of building: 50 Years
Upwind hill angle: 2° (less than 3°)
Terrain category: II
Structure from the point of view of cyclonic importance: Industrial structures
Choice of wind direction: choice 1 (i.e., wind interaction with triangular, square, rectangular buildings)

TABLE I BUILDING DIMENSIONS

Building	Length (m)	Width (m)	Height (m)
B1	10	10	20
B2	10	10	40
B3	10	10	60

III. METHODOLOGY

First, the wind pressure evaluation is explained based on IS 875 (Part 3), 1987 and then the design pressure evaluation based on revised IS 875 (Part 3) 2015 is discussed.

A. Standards as Per is 875 (Part 3), 1987

Basic Wind Speed (v_b) is based on the peak gust velocity which is averaged over a time interval of about 3 seconds. This wind speed corresponds to mean heights above ground level in an open terrain (at a height of 10m above the mean ground level). IS code gives the map of India showing basic wind speed (v_b) for different zones. Also, in the Annex A (IS 875 (Part 3), 1987) the same has been tabulated for different cities.

Further, if designer needs to consider risk level, terrain roughness, height & size of structure, and local topography; then, design wind speed (v_z)can be obtained from the basic wind speed. This component v_z depends on certain factors and coefficients. Risk level is associated with the probability factor (risk coefficient, k_1),terrain roughness, height & size of structure are associated with terrain factor (k_2), and local topography is related to topography factor (k_3).

In these factors, k_2 and k_3 are the most critical in terms of evaluation as they tackle the terrain and topology of the region. The selection of terrain is made considering the ground surface roughness.

The terrain category used in the design of a structure may change based on the wind direction in hand. Terrain of a stipulated structure can be categorised into the four categories as shown in the Table II.

TABLE II TERRAIN CATEGORIES [11], [15]

Category			
1	2	3	4
Open terrain where the average height of any nearby object is less than 1.5 m and there are little or no obstructions (open seacoasts and flat treeless plains.)	Open landscape with barriers that are widely spaced and typically range in height from I.5 to 10 m (airfields, open parklands, and lightly developed edges of towns and suburbs). Due of the roughness of the enormous sea waves and the high winds, open area next to the seacoast may also be categorised as Category 2.)	Terrain with many small, closely spaced obstacles that are up to 10 metres tall, with or without a few isolated large structures. (This group includes densely forested areas, shrubs, fully or partially built towns, and industrial sites.)	Terrain with several massive, tall, and closely placed obstacles. (This group consists of well-developed industrial complexes and significant metropolitan centres, typically with obstructions over 25 m.)

These categories are further subdivided into classes based on the size of the structures and/or their components such as cladding, glazing, roofing, etc. as class A, class B, and class C. The effect of topography will be significant at a site when the upwind slope (θ) is greater than3° ($k_3 = 1.0$ to 1.36), and below that, the value of k_3 may be taken to be equal to 1.0. It may be noted that the value of k_3 varies with height above ground and it is maximum near the ground surface whereas, it becomes 1.0 at higher levels. Combining the above factors and basic wind speed the design wind speed at any height zcan be written as [15]

$$v_z = k_1 k_2 k_3 v_b \quad (1)$$

The design wind pressure is evaluated based on the design wind speed as

$$p_z = 0.6 \times v_z^2 \quad (2)$$

B. Standards as Per Revised is 875 (Part 3), 2015

Major revisions includes in k_2 factor for considering terrain only. New factor for cyclonic region has been introduced as k_4 that defines the effects on emergency services structures like cyclone shelters, hospitals, schools, communication towers etc. The effect of cyclone is very severe upto60 km distance from coast. It varies from 1.0 (general purpose structures) to 1.3 (Structure of post-cyclone importance). Accordingly, the design wind speed formula can be written as [15]

$$v_z = k_1 k_2 k_3 k_4 v_b \qquad (3)$$

In the revised code the design pressure (p_d) is obtained by considering the wind directionality factor (K_d), area averaging factor (K_a) and the combination factor (K_c) along with p_z as follows[15]

$$p_d = K_d K_a K_c p_z \qquad (4)$$

K_d considers the randomness in the directionality of wind, it is either considered 0.9 or 1.0depending whether it is building/solid signs/open signs/lattice frameworks/truss towers or a building with circular/close to circular form. Factor K_a is called as the area averaging factor which considers the decrease in pressure due to increase in the tributary area. The factor K_c is for wind load on frames specially for the cases when roof is subjected to the pressure and the internal pressure is suction. In that case K_c will be taken as 0.9.

IV. SIMULATING WIND PRESSURE IN PYTHON

Initially, the function for wind speed is created. This function has a dictionary which contains locations as a key and basic wind speed at that location as its pair. The function has one argument, which is the location and will return the corresponding value at that location. The function is as follows.

Function for Basic wind speed

```
def Basic_Wind_Speed(loc):
    """
    Function to Evaluate Basic Wind Speed
    Input: Location
    Output: Basic wind speed in m/s
    """

    data={“Agra”:47,”Ahmadabad”:39,”Ajmer”:47, “Almora”:47,

“Amritsar”:47,”Asansol”:47,”Aurangabad”:39,”Bahraich”:47,

“Bangalore”:33,”Barauni”:47,”Bareilly”:47,”Bhatinda”:47,

“Bhilai”:39,”Bhopal”:39,”Bhubaneshwar”:50,”Bhuj”:50,

“Bikaner”:47,”Bokaro”:47,”Mumbai”:44,”Calcutta”:50,

“Calicut”:39,”Chandigarh”:47,”Chenni”:50,”Coimbatore”:39,

“Kolkata”:50,”Darbhanga”:55,”Darjelling”:47,”DehraDun”:47,

“Delhi”:47,”Durgapur”:47,”Gangtok”:47,”Gauhati”:50,”Gaya”:39,

“Gorakhpur”:47,”Hyderabad”:44,”Imphal”:47,”Jabalpur”:47,

“Jaipur”:47,”Jamshedpur”:47,”Jhansi”: 47,”Jodhpur”:47,

“Kanpur”:47,”Kohima”:44,”Kurnool”:39,”Lakshadweep”:39,

“Lucknow”:47,”Ludhiana”:47,”Madras”:50,”Madurai”:39,

“Mandi”:39,”Mangalore”:39,”Moradaba”:47,”Maysore”:33,

“Nagpur”:44,”Nainital”:47,”Nasik”:39,”Nellore”:50,

“Panjim”:39,”Patiala”:47,”Patna”:47,”Pondicherry”:50,

“PortBlair”:44,”Pune”:39,”Raipur”:39,”Rajkot”:39,

“Ranchi”:39,”Roorkee”:39,”Rourkela”:39,”Simla”:39,

“Srinagar”:39,”Surat”:44,”Tiruchchirrappaalli”:47,

“Trivandrum”:39,”Udaipur”:47,”Vadodara”:44,”Varanasi”:47,

“Vijaywada”:50,”Visakhapatnam”:50}
    return data[loc]
```

Next is the function for probability factor/risk coefficient (k_1). This function accepts two arguments i.e., the life of the building and the basic wind speed and returns k_1. Here two one dimensional arrays are defined for building life & basic wind speeds are created along with the two dimensional array for k_1 factor. Function where () is used to search for the index of the supplied year and basic wind speed and the outputs are assigned **i** and **j** variables. Then corresponding to the [i,j] the k_1 will be searched and returned.The function is as follows:

Probability factor/Risk coefficient (k_1)

```
def Probability_factor(life,vb):
    """
    Function to Risk Coefficient/Probability
    factor
    Input: Life of structure and Basic wind speed
    Output: k1
    """
    year=array([50,5,25,100])
    bws=array([33,39,44,47,50,55])
    data=array([[1,1,1,1,1,1],
    [0.82,0.76,0.73,0.71,0.70,0.67],
    [0.94,0.92,0.91,0.90,0.90,0.89],
    [1.05,1.06,1.07,1.07,1.08,1.08]])
    i=where(year==life)[0][0]
    j=where(bws==vb)[0][0]
    return data[i,j]
```

In the next step terrain factor (k_2) is defined. In this all the categories are kept in two dimensional arrays and class is written as one dimensional array. Class of the building is selected based on the condition of maximum dimension of building.

The where () function is used to find the index of the class in Class variable and assign it to j. As in this case k_2 corresponds to varying heights (10 m to the given height), therefore arrange () function will create the array of heights respectivek_2.

Then within a for loop corresponding to the respective heights indexes of vales in H is found out with the help of where () function and stored in j. Now corresponding to [i,j] the respective value of k_2 in the supplied terrain category is picked up and appended to the list k2. NumPy's inbuilt function interp () is used if the intermediate value of height is not there in the list. The function returns the array of building heights and corresponding k_2 values.

Terrain factor (k_2) (Based on IS 875 (Part 3), 1987)

```
def Terrain_factor_old(BL,BH,BW,Category):
    """
    Function to Evaluate Terrain Factor (k2)
    Input: L, H, W of structure and its terrain category
    Output: k2
    """
    BHH=arange(10,BH+1,1)

    # Selection of Class
    if max(BL, BH, BW)<20:
        cls='A'

    elif 20<=max(BL, BH, BW)<=50:
        cls='B'

    else:
        cls='C'
    #print("Class of building is: \n",cls)

    # Number of Classes
    Class=array(['A','B','C'])
    j=where(Class==cls)[0][0]

    # For recording k2 at different height
    k2=[]

    for BH in BHH:

        # Different heights
        H=array([10,15,20,30,50,100,150,200,250,300,350,400,450,500])

        # Different Category
        if Category == "I":
            Cat=array([[1.05, 1.03, 0.99],[1.09, 1.07, 1.03],
                      [1.12, 1.10, 1.06],[1.15, 1.13, 1.09],
                      [1.20, 1.18, 1.14],[1.26, 1.24, 1.20],
                      [1.30, 1.28, 1.24],[1.32, 1.30, 1.26],
                      [1.34, 1.32, 1.28],[1.35, 1.34, 1.30],
                      [1.37, 1.35, 1.31],[1.38, 1.36, 1.32],
                      [1.39, 1.37, 1.33],[1.40, 1.38, 1.34]])

        elif Category =="II":
            Cat=array([[1.00, 0.98, 0.93],[1.05, 1.02, 0.97],
                       [1.07, 1.05, 1.00],[1.12, 1.10, 1.04],
                       [1.17, 1.15, 1.10],[1.24, 1.22, 1.17],
                       [1.28, 1.25, 1.21],[1.30, 1.28, 1.24],
                       [1.32, 1.31, 1.26],[1.34, 1.32, 1.28],
                       [1.36, 1.34, 1.29],[1.37, 1.35, 1.30],
                       [1.38, 1.36, 1.31],[1.39, 1.37, 1.32]])

       elif Category =="III":
            Cat=array([[0.91, 0.88, 0.82],[0.97, 0.94, 0.87],
                       [1.01, 0.98, 0.91],[1.06, 1.03, 0.96],
                       [1.12, 1.09, 1.02],[1.20, 1.17, 1.10],
                       [1.24, 1.21, 1.15],[1.27, 1.24, 1.18],
                       [1.29, 1.26, 1.20],[1.31, 1.28, 1.22],
                       [1.32, 1.30, 1.24],[1.34, 1.31, 1.25],
                       [1.35, 1.32, 1.26],[1.36, 1.33, 1.28]])

       else:
            Cat=array([[0.80, 0.76, 0.67],[0.80, 0.76, 0.67],
                       [0.80, 0.76, 0.67],[0.97, 0.93, 0.83],
                       [1.10, 1.05, 0.95],[1.20, 1.15, 1.05],
                       [1.24, 1.20, 1.10],[1.27, 1.22, 1.13],
                       [1.28, 1.24, 1.16],[1.30, 1.26, 1.17],
                       [1.31, 1.27, 1.19],[1.32, 1.28, 1.20],
                       [1.33, 1.29, 1.21],[1.34, 1.30, 1.22]])
        if BH in H:
            i=where(H==BH)[0][0]
            k2.append(Cat[i,j])
        else:
            k2.append(interp(BH, H, Cat[:,j]))

    k2=array(k2)

    return BHH, k2
```

Then function for topography factor (k_3) is obtained by considering the condition that if the uphill angle is less than 3 then return 1 else return 1.36. The function accepts the value of θ. The function is as follows.

Topography factor (k_3)

```
   def Topography_factor(θ):
       """
 Function to Evaluate Topography Factor (k3)
       Input: Angle of hill
       Output: k3
       """
       if θ>3:
           return 1.36
       else:
                          return 1
```

Next design wind speed function (v_z) and design wind pressure functions (p_z) were developed on the basis of Eq's. 1 and 2 (for old IS codes). The former accepts design wind speed, k_1,k_2, and k_3 as arguments whereas, the later accepts the output of wind speed function. The functions areas follows.

Design wind speed (v_z)
(Based on IS 875 (Part 3), 1987)

```
def
Design_Wind_Speed_old(vb,k
1,k2,k3):
    """
    Function to Evaluate
Design Wind Speed
    Input: vb,k1,k2,k3
    Output: Design wind
speed in m/s
    """
      return vb*k1*k2*k3
```

Design wind pressure (p_z)
(Based on IS 875 (Part 3), 1987)

```
def
Design_Wind_Pressure_
old(v):
     return 0.6*v**2
```

To make the program more user friendly all the above functions are bundled in one big function called as Old_IS(). This function will accept buildings location, life, length, height, width, terrain category, and upwind hill angle as arguments.

It will return different z locations along the height of the building and the corresponding design wind speeds and pressures corresponding to the old IS code. This function will reduce the burden on the user to check that whether they have taken coefficients and inputs in correct order or not. The function is as follows.

Collection of all the above functions in one

```
def
Old_IS(Location,Life,BL,BH,BW,Category,θ):
    """
    Input:
Location,Life,BL,BH,BW,Category,θ
    Returns: H,vz,pd

    """
    vb=Basic_Wind_Speed(Location)

k1=Probability_factor(Life,vb)

H,k2=Terrain_factor_old
(BL,BH,BW,Category)

k3=Topography_factor(θ)

vz=Design_Wind_Speed_old(vb,k1,k2,k3)

pd=Design_Wind_Pressure_old(vz)
                return H,vz,pd
```

Now based on revised IS code the function to obtain terrain factor at different elevations is as follows.

Terrain factor (k_2) (Based on IS 875 (Part 3), 2015)

```
def Terrain_factor_new(BL,BH,BW,Category):
    """
    Function to Evaluate Terrain Factor (k2)
    Input: L, H, W of structure and its terrain category
    Output: k2
    """
    BHH=arange(10,BH+1,1)

    # For recording k2 at different height
    k2=[]
    for BH in BHH:

    # Different heights
        H=array([10,15,20,30,50,100,150,200,250,300,350,400,450,500])

    # Different Category
        if Category == "I":
            Cat=array([1.05,1.09,1.12,1.15,1.2,1.26,1.3,1.32,
                      1.34,1.35,1.35,1.35,1.35,1.35])
        elif Category =="II":
            Cat=array([1,1.05,1.07,1.12,1.17,1.24,1.28,1.3,
                       1.32,1.34,1.35,1.35,1.35,1.35])
        elif Category =="III":
            Cat=array([0.91,0.97,1.01,1.06,1.12,1.2,1.24,
                       1.27,1.29,1.31,1.32,1.34,1.35,1.35])
        else:
            Cat=array([0.8,0.8,0.8,0.97,1.1,1.2,1.24,1.27,
                       1.28,1.3,1.31,1.32,1.33,1.34])
        if BH in H:
            i=where(H==BH)[0][0]
            k2.append(Cat[i])
        else:
            k2.append(interp(BH, H, Cat))
   k2=array(k2)
                              return BHH, k2
```

Apart from revised k_2 list the revised IS code has also incorporated the risk factor (I) whose function is as follows.

Risk factor (I) (Based on IS 875 (Part 3), 2015)

```
def Risk_factor(choice):
    """
    Structures of post cyclone importance for emergency
services such as hospitals, schools, = 1.3
    communication towers etc.

    Industrial structures  = 1.15

    All other structures   = 1
    """
    if choice==1:
        return 1.3
    elif choice==2:
        return 1.15
    else:
                        return 1
```

Then based on the Eq's. 3 the function for new design wind speed is developed, which is presented as follows.

Design wind speed (v_z)
(Based on IS 875 (Part 3), 2015)

```
def Design_Wind_Speed_new(vb,k1,k2,k3,k4):
    """
    Function to Evaluate Design Wind Speed
    Input: vb,k1,k2,k3
    Output: Design wind speed in m/s
    """
                 return vb*k1*k2*k3*k4
```

To obtain design wind pressure based on the revised IS code, wind directionality and area averaging factors are to be evaluated which are modelled in the functional form as follows.

Wind directionality factor (K_d)
(Based on IS 875 (Part 3), 2015)

```
def Directionality_factor(a):
    """
    Triangular, square, rectangular buildings
=0.9
    Circular or near-circular buildings = 1
    Cyclone affected regions = 1
    """
    if a==1:
        return 0.9
    elif a==2:
        return 1
    elif a==3:
        return 1
    else:
                print("Wrong entry")
```

Area averaging factor (K_a)
(Based on IS 875 (Part 3), 2015)

```
def
Area_Averag_factor(area):
    if area<=10:
        return 1
    elif 10<area<25:
        return 1-6.667E-
3*(area-10)
    elif area==25:
        return 0.9
    elif 25<area<100:
        return 1-1.33E-
3*(area-25)
    elif area>=100:
        return 0.8
```

Now based on Eq. 4 the design wind pressure corresponding to the revised IS code is developed for which K_c is taken as 1. The function is as follows.

Design wind pressure (p_d) (Based on IS 875 (Part 3), 2015)

```
def Design_Wind_Pressure_new(Kd,Ka,pz,Kc=1):
                return Kd*Ka*Kc*pz
```

Again, to simplify the task for the users the functions for the evaluation of the design wind speed and pressures at elevations are bundled in one function called as New_IS (). This function accepts buildings location, life, length, height, width, category, uphill angle, risk factor, and wind direction as the arguments and return different building elevations along with corresponding design wind speed and pressures.

Finally, two more functions were developed for data plotting and inputs to fully automate the task of wind pressure calculations on the building. These functions are as follows.

Function for plotting the results

```
def
plotting(Location,Life,BL,BH,BW,Category,θ,I,
wind_direction):
    """
    Input:
Location,Life,BL,BH,BW,Category,θ,I,
wind_direction
    """

H,vz_n,pd_n=New_IS(Location,Life,BL,BH,BW,
Category,θ,I,wind_direction)


H,vz_o,pd_o=Old_IS(Location,Life,BL,BH,BW,
Category,θ)

    figure(1,dpi=200)
    plot(vz_n,H,'r-o',label="New")
    plot(vz_o,H,'b-s',label="Old")
    xlabel("vz")
    ylabel("H")
    legend()

    figure(2,dpi=200)
    plot(pd_n,H,'r-o',label="New")
    plot(pd_o,H,'b-s',label="Old")
    xlabel("vz")
    ylabel("H")
    legend()
                show()
```

Function for asking data from user

```
def Input_data():
    """
    Returns
:Location,Life,BL,BH,BW,Category,θ,I,
wind_direction
    """
    Location=input("Enter the location to
evaluate the basic wind Speed:\n")
    print("-----------------------------------
--------------------------")

    Life=int(input("Enter the life of
building\n"))
    print("-----------------------------------
-----------")

    Category=input("Enter the terrain
category('I','II','III','IV')\n")
    print("-----------------------------------
--------------------------")

    θ=float(input("Enter the value of θ
(hill):\n"))
    print("-----------------------------------
--------")

    I=int(input("Enter from the below
mentioned choices:\n\
    -------------------------------------\n\
              Choice = 1\n\
    -------------------------------------\n\
    -->If structures of post cyclone
importance\n\
    -->Emergency services such as:\n\
    (a)hospitals\n\
    (b)schools\n\
    (c)communication towers etc\n\
    \n\
    -------------------------------------\n\
              Choice = 2\n\
    -------------------------------------\n\
    --> Industrial structures\n\
    -------------------------------------\n\
    \n\
              Choice = 3\n\
    -------------------------------------\n\
    Otherwise for all other structures"))

    wind_direction=int(input("""Enter the
wind direction choice:\n
```

```
        -------------------------------------
                     Choice = 1

          Triangular, square, rectangular
 buildings
        -------------------------------------
                     Choice = 2

          Circular or near-circular buildings
        -------------------------------------
                     Choice = 3

          Cyclone affected regions
        -------------------------------------
          """))

        BL=float(input("Enter Length\n"))
        BH=float(input("Enter Height\n"))
        BW=float(input("Enter Width\n"))
                          return

         Location,Life,BL,BH,BW,Category,θ,I,
                    wind_direction
```

At last all the above functions are bundled in one .py file a module is created for ready use. The name of the module is “Wind_Load”.

V. RESULTS AND DISCUSSION

First, the function input() is called to supply all the necessary data. Arrays for length, width, and height were generated for its comparison at later stages.

From both the IS codes, values of v_b, k_1, and k_3 will come out to be the same i.e.: $v_b = 39$, $k_1 = 1.0$, and $k_2 = 1$. Fig. 1 gives the variation of $k_2$for all the buildings.

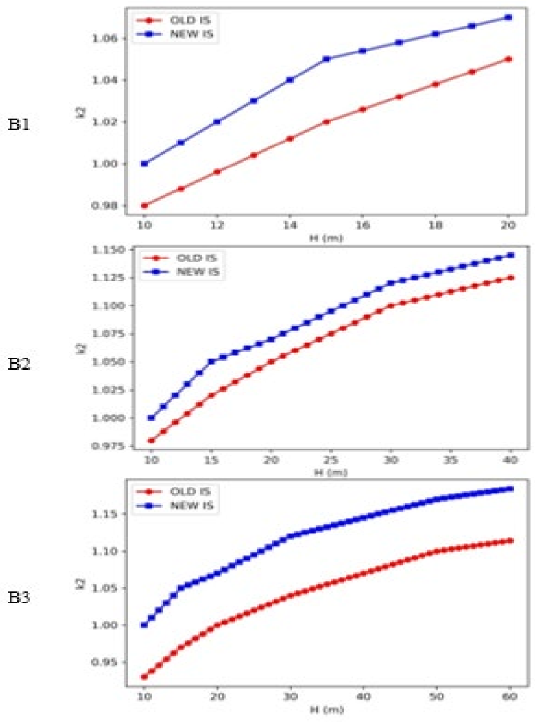

Fig. 1 Variation of k_2 as a function of building height

From the Fig. 1, it is observed that in all the buildings the value of k_2 given by revised IS code is more than the one given by old. This may be attributed to giving more weight age to the terrain property than the dimension of the building in the revised code (i.e., removal of building classes from k_2). But more interesting thing can be seen if we plot the percentage change in the values of k_2 obtained from revised as compared from old IS code (Fig. 2).

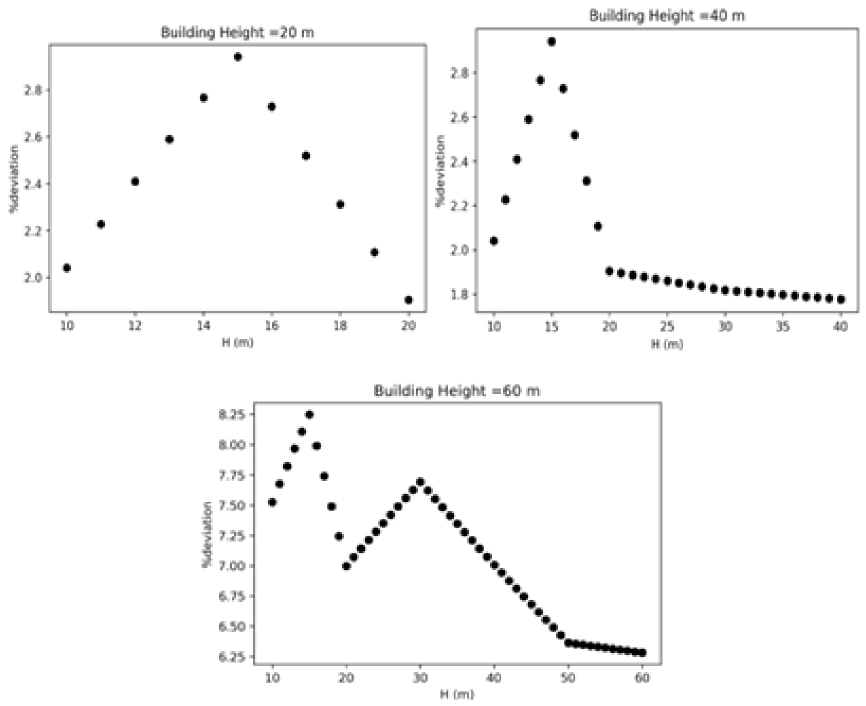

Fig. 2 Percentage deviation of k_2 from old and revised IS codes

Now it can be observed that the maximum variation for all the cases is observed up to 15 m height and subsequently decreases with the increasing height of the building. Moreover, for 20 m and 40 m high building the maximum error is close to 3% but as we move to 60 m high building the maximum % error reaches close to 8.5% at a height of 15 m. Also, for 60 m height building the % error rises up to 7.75% at the height of 30 m.

The variation in the design wind velocity and pressure for the buildings is shown in Fig. 3.

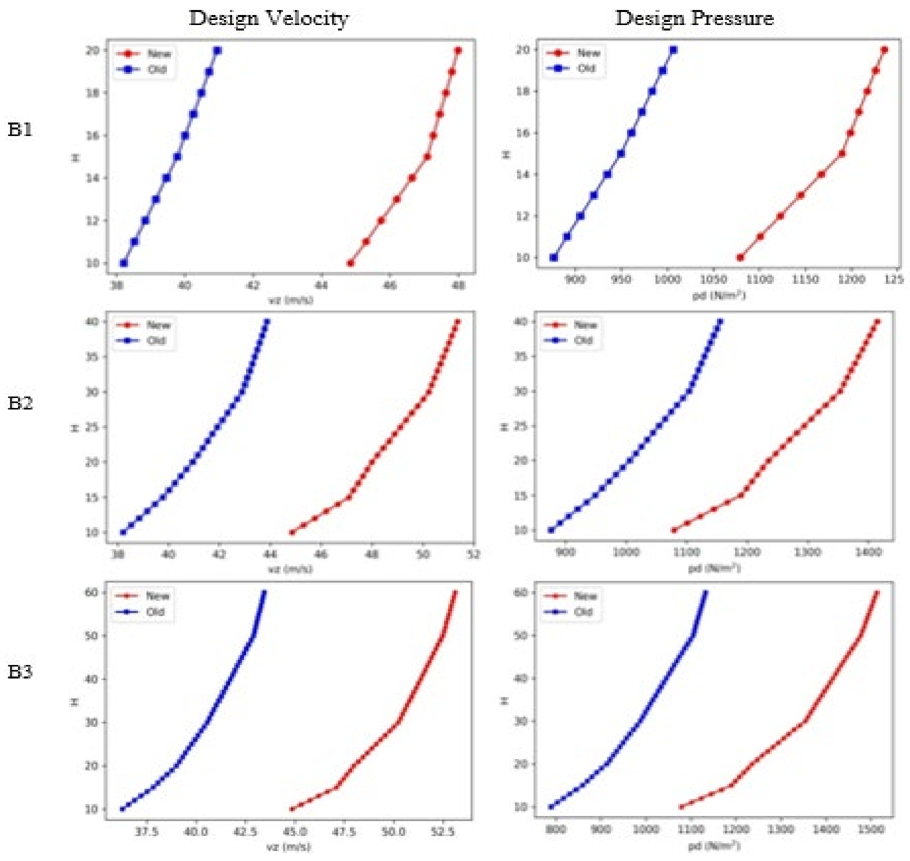

Fig. 3 Variation of design wind speed and pressure for buildings

The design velocity and pressure as obtained from revised IS code are higher than that obtained from the old. Probably, this is due to the fact that in revised code while calculating the design velocity importance factor (k_4) has been incorporated (which includes factors for cyclonic regions). As the trends for all height buildings are approximately similar. Hence, for better understanding the percent deviation for 60 m height building is shown in Fig. 4.

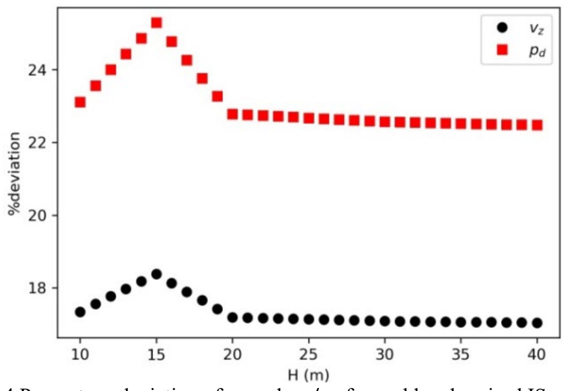

Fig. 4 Percentage deviation of v_z and p_d/p_z from old and revised IS codes

Therefore, it can be noted that the difference between old and revised codes show a substantive difference, but it lies at a constant value of 34 and 22% for pressure and velocity as the height of building increases.

VI. CONCLUSION

The Python programming language was used to model the wind load pressures in this article. Additionally, the factors needed to comprehend wind loads have been documented as Python functions. The implementation of three typical building structures has been demonstrated, and the findings are consistent with those of manual calculations. It has been observed that for all the buildings the value of k_2 obtained from the revised IS code is more than the one obtained from the old. Also the magnitudes of pressure forces on the building obtained from the revised codes are higher in comparison to the old. This may be due to the incorporation of importance factor and removal of the building classes in the revised codes. Therefore, this article will assist wind practicing engineers in creating a programming-based strategy to solve technical issues. Additionally, the generated modules can be directly applied by wind engineering researchers to assess the wind loads more accurately.

Nomenclature

I	Risk factor
k_1	Risk coefficient/Probability factor
k_2	Terrain factor
k_3	Topography factor
k_4	Importance factor
K_a	Area averaging factor
K_c	Wind load on frames
K_d	Randomness in the directionality of wind
p_d	Design pressure for new code
p_z	Wind pressure at height z/Design wind pressure for old code
v_b	Basic wind speed
v_z	Design wind speed

Abbreviations

IS	Indian Standards

REFERENCES

[1] J. E. Cermak and R. E. Atkins, "Wind loads on structures," 1976. DOI: 10.1680/ijoti.1950.12925.

[2] J. D. Holmes, C. Paton, and R. Kerwin, "Wind Loading of Structures," CRC press, 2007. DOI: 10.4324/9780203964286.

[3] J. D. Holmes, C. Paton, and R. Kerwin, "Wind Loading of Structures," *Wind Load. Struct.,* Vol. 88, 2007, DOI: 10.4324/9780203964286.

[4] K. Butler, S. Cao, A. Kareem, Y. Tamura, and S. Ozono, "Surface pressure and wind load characteristics on prisms immersed in a simulated transient gust front flow field," *J. Wind Eng. Ind. Aerodyn.,* Vol. 98, No. 6-7, pp. 299-316, 2010, DOI: 10.1016/j.jweia.2009.11.003.

[5] M. E. Greenway, "An analytical approach to wind velocity gust factors," *J. Wind Eng. Ind. Aerodyn.,* Vol. 5, No. 1-2, pp. 61-91, 1979, DOI: 10.1016/0167-6105(79)90025-4.

[6] P. Bot, I. M. Viola, R. G. J. Flay, and J. S. Brett, "Wind-tunnel pressure measurements on model-scale rigid downwind sails," *Ocean Eng.,* Vol. 90, pp. 84-92, 2014, DOI: 10.1016/j.oceaneng.2014.07.024.

[7] J. E. Cermak and W. Z. Sadeh, "Wind-tunnel simulation of wind loading on structures," *J. Wind Eng. Ind. Aerodyn.,* Vol. 91, No. 12-15, pp. 1627-1649, 1971.

[8] L. D. Zhu, L. Li, Y. L. Xu, and Q. Zhu, "Wind tunnel investigations of aerodynamic coefficients of road vehicles on bridge deck," *J. Fluids Struct.,* Vol. 30, pp. 35-50, 2012, DOI: 10.1016/j.jfluidstructs.2011.09.002.

[9] Y. Uematsu and N. Isyumov, "Wind pressures acting on low-rise buildings," *J. Wind Eng. Ind. Aerodyn.,* Vol. 82, No. 1, pp. 1-25, 1999, DOI: 10.1016/S0167-6105(99)00036-7.

[10] R. Woo Kim, I. Bok Lee, U. Hyeon Yeo, and S. Yeon Lee, "Evaluation of various national greenhouse design standards for wind loading," *Biosyst. Eng.,* Vol. 188, pp. 136-154, 2019, DOI: 10.1016/j.biosystems eng.2019.10.004.

[11] M. M. M. Al-deraan, "A Comparative Study of Wind Forces on Tall Building as Per Is 875-Part-III (1987) and Draft Code (2011) using Gust Factor Method A Comparative Study of Wind Forces on Tall Building as Per Is 875- Part-III (1987) and Draft Code (2011) using Gust Fa," *Int. J. Sci. Technol. Res*. ISSN, September 2013, pp. 2319-8885, 2018.

[12] S. M. Pimpalkar and K. Padmawar, "To Analyze and Comparing a G+12 story RCC building using IS-875 (part 3)-2015 and ASCE-07 for basic wind speed of 50m/s using STADD PRO software," *Int. Res. J. Eng. Technol*., 2022, [Online]. Available: www.irjet.net.

[13] K. K. N.M. Bhandari and Prem Krishna, "An Explanatory Handbook on Proposed IS 875 (Part3) Wind Loads on Buildings and Structures," *News. Ge,* [Online]. Available: https://news.ge/anakliis-porti-aris-qveynis-momava, 2018.

[14] M. M. M. Al-deraan, "A Comparative Study of Wind Forces on Tall Building as Per Is 875-Part-III (1987) and Draft Code (2011) using Gust Factor Method A Comparative Study of Wind Forces on Tall Building as Per Is 875- Part-III (1987) and Draft Code (2011) using Gust Fa," No. September 2013, 2018.

[15] P. K. Goyal and N. Suthar, "Comparison of Response on Building Due to Wind Load as Per Wind Codes [IS 875-(Part 3)-2015] and [AS/NZ1170.2-2011]," *in Lecture Notes in Civil Engineering*, *Springer,* Vol. 274, 2023, pp. 307-317. DOI: 10.1007/978-981-19-4055-2_25.

[16] S. Prajapati and K. Soni, "Analysis of A Tall Structure Using Staad. Pro Providing Different Wind Intensities as Per 875 Part-III," 2019.

[17] P. Dumka, A. Deo, K. Gajula, V. Sharma, R. Chauhan, and D. R. Mishra, "Load and Load Duration Curves Using Python," *Int. J. All Res. Educ. Sci. Methods,* Vol. 10, No. 8, pp. 2127-2134, 2022.

[18] P. S. Pawar, D. R. Mishra, P. Dumka, and M. Pradesh, "Obtaining Exact Solutions of Visco-Incompressible Parallel Flows Using Python," *Int. J. Eng. Appl. Sci. Technol.,* Vol. 6, No. 11, pp. 213-217, 2022.

[19] K. Gajula, V. Sharma, B. Sharma, D. R. Mishra, and P. Dumka, "Modelling of Energy in Transit Using Python," *Int. J. Innov. Sci. Res. Technol.,* Vol. 7, No. 8, pp. 1152-1156, 2022.

[20] P. Dumka, P. S. Pawar, A. Sauda, G. Shukla, and D. R. Mishra, "Application of He's homotopy and perturbation method to solve heat transfer equations: A python approach," *Adv. Eng. Softw.,* Vol. 170, pp. 103160, May 2022, DOI: 10.1016/j.advengsoft.2022.103160.

[21] C. Fuhrer, O. Verdier, J. E. Solem, C. Führer, O. Verdier, and J. E. Solem, Scientific Computing with Python. High-performance scientific computing with NumPy, SciPy, and pandas. *Packt Publishing Ltd*, 2021.

[22] C. Bauckhage, "NumPy/SciPy Recipes for Data Science: Subset-Constrained Vector Quantization via Mean Discrepancy Minimization," pp. 1-4, February 2020.

[23] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," *Comput. Sci. Eng.,* Vol. 13, No. 2, pp. 22-30, 2011, DOI: 10.1109/MCSE.2011.37.

[24] P. Dumka, K. Rana, S. Pratap, S. Tomar, P. S. Pawar, and D. R. Mishra, "Modelling air standard thermodynamic cycles using python," *Adv. Eng. Softw.,* Vol. 172, pp. 103186, July 2022, DOI: 10.1016/j.advengsoft.2022.103186.

[25] J. Ranjani, A. Sheela, and K. Pandi Meena, "Combination of NumPy, SciPy and Matplotlib/Pylab-A good alternative methodology to MATLAB-A Comparative analysis," *in Proceedings of 1st International Conference on Innovations in Information and Communication Technology, ICIICT 2019*, pp. 1-5. DOI: 10.1109/ICIICT1.2019.8741475.

[26] P. Dumka, R. Chauhan, A. Singh, G. Singh, and D. Mishra, "Implementation of Buckingham ' s Pi theorem using Python," *Adv. Eng. Softw.,* Vol. 173, pp. 103232, July 2022, DOI: 10.1016/j.adv engsoft.2022.103232.

[27] M. Cywiak and D. Cywiak, "SymPy," in Multi-Platform Graphics Programming with Kivy: Basic Analytical Programming for 2D, 3D, and Stereoscopic Design, Berkeley, CA: Apress, pp. 173-190, 2021. DOI: 10.1007/978-1-4842-7113-111.

[28] E. Bisong, "Matplotlib and Seaborn," *in Building Machine Learning and Deep Learning Models on Google Cloud Platform,* Berkeley, CA: Apress, pp. 151-165, 2019. DOI: 10.1007/978-1-4842-4470-8_12.

[29] V. Porcu, "Matplotlib," in Python for Data Mining Quick Syntax Reference, Berkeley, CA: A press, pp. 201-234, 2018. DOI: 10.1007/978-1-4842-4113-4_10.