International Journal of Pure and Applied Mathematical Sciences
ISSN 0972-9828 Volume 19, Number 1 (2026) pp. 29-59
Research India Publications
https://dx.doi.org/10.37622/IJPAMS/19.1.2026.29-59

Computational Modelling of Chemical Graphs Using Python:A Study Based on Chemical Graph Theory

[1]Vishwas V Rudraswamy Math , [2]Gowri S.B, [2]Pavan Hiremath , [2]Pallavi Mallappa, [2]Pallavi R
[1]Assistant Professor, [2]Post Graduate Students
Department of Mathematics, Sharnbasva University,
Kalaburagi, Karnataka, India
Email id :vishwasrmath@gmail.com

Abstract

This study presents the computational modelling of chemical graphs using Python IDLE and the Turtle graphics library based on the principles of Chemical Graph Theory.Chemical compounds are represented as graphs, where atoms are vertices and chemical bonds are edges. The work computes and visualizes important graph-theoretical descriptors, including the First and Second Zagreb Indices,Wiener Index, Randic Index,vertex degree, eccentricity, diameter, and radius.These topological indices are widely used in QSAR and QSPR studies to predict molecular structure and properties.

The study also examines physiochemical and electronic descriptors related to molecular size, branching, connectivity, electron distribution, and molecular stability, which are useful for predicting chemical reactivity and biological activity. Python algorithms are developed to calculate these descriptors efficiently, while Turtle graphics provides an interactive visualization of molecular structures.

The proposed approach offers a simple, low-cost, and effective computational framework for learning and analyzing chemical graphs. It demonstrates that Python can accurately model molecular structures and compute essential graph descriptors, making it valuable for education, cheminformatics, drug discovery, and materials science.

Introduction

Chemical Graph Theory (CGT) is an important branch of mathematical chemistry that combines graph theory, chemistry, and computer science to represent and analyze molecular structures using mathematical models. In Chemical Graph Theory, a molecule is represented as a graph (G = (V, E)), where vertices denote atoms and edges represent chemical bonds. This graphical representation provides an efficient mathematical framework for studying the structural, topological, and physicochemical

properties of chemical compounds without relying solely on laboratory experiments. Over the past few decades, CGT has become a significant research area due to its wide applications in computational chemistry, cheminformatics, pharmaceutical research, and materials science.

One of the major contributions of Chemical Graph Theory is the development of **topological indices**, which are numerical descriptors derived from molecular graphs. These indices capture structural information such as branching, connectivity, and molecular compactness, and they are extensively used in **Quantitative Structure–Activity Relationship (QSAR)** and **Quantitative Structure–Property Relationship (QSPR)** studies. QSAR and QSPR models establish mathematical relationships between molecular structure and biological or physicochemical properties, enabling researchers to predict toxicity, boiling point, stability, solubility, biological activity, and pharmacological behavior before experimental synthesis. This significantly reduces experimental costs and accelerates drug discovery and material design.

Among the numerous graph-theoretical descriptors, the **First Zagreb Index**, **Second Zagreb Index**, **Wiener Index**, and **Randić Index** are widely recognized because of their effectiveness in describing molecular branching and connectivity. The Wiener Index, introduced by Harry Wiener in 1947, is based on the sum of the shortest-path distances between all pairs of vertices and has been successfully applied in predicting boiling points and other physicochemical properties of hydrocarbons. Similarly, the Zagreb indices and the Randić connectivity index have demonstrated strong correlations with molecular stability, chemical reactivity, and biological activity. In addition, graph parameters such as vertex degree, eccentricity, diameter, and radius provide valuable information about the connectivity and structural characteristics of molecular graphs.

The primary objective of this study is to integrate graph theory with Python programming for the computational modeling of chemical structures. The proposed approach computes major topological indices, analyzes molecular descriptors, and visualizes chemical graphs using simple Python programs. By combining mathematical modeling, algorithmic computation, and graphical representation, this work demonstrates an efficient, low-cost, and accessible framework for studying molecular structures. The proposed methodology can serve as a useful educational tool and has potential applications in cheminformatics, drug discovery, molecular property prediction, bioinformatics, and materials science.

Research Methodology

The present work proposes a computational framework for modelling chemical compounds using the principles of Chemical Graph Theory and Python programming. The methodology consists of representing molecular structures as mathematical graphs, computing graph-theoretical descriptors, and visualizing molecular graphs using Python IDLE and the Turtle Graphics library. The complete work flow adopted in this research is illustrated through the benzene molecule (C_6H_6) as a representative example.

Step 1: Selection of the Chemical Molecule
The first step involves selecting a suitable chemical compound for computational analysis.In this study, the benzene molecule(C_6H_6) is chosen because of its simple cyclic structure and its importance inorganic chemistry. The proposed methodology can also be applied to other molecules such as methane (CH_4), water (H_2O), octane (C_8H_{18}), ethanol, and similar chemical compounds.

Step 2: Graph Representation of the Molecule
The selected molecule is transformed into its corresponding molecular graph using the concepts of Chemical Graph Theory.
In the molecular graph,
- Each atom is represented as a **vertex(node)**,
- Each chemical bond is represented as an **edge** connecting two vertices. For benzene,
- Carbon atoms are represented as six vertices.
- Carbon–carbon bonds form the cyclic hexagonal graph.
- Hydrogen atoms may be included depending on the graph representation adopted.

This graph serves as the mathematical model for all subsequent computations.

Step 3: Computational Modelling Using Python
The computational model is implemented using **Python IDLE**.
Initially, the coordinates of every atom are defined to represent the molecular geometry. The molecular graph is then constructed by connecting the corresponding vertices according to the bonding structure of the compound.
Python algorithms are developed to
- Create the molecular graph,
- Store vertex and edge information,
- Calculate graph-theoretical descriptors,
- Display the computational results.

This approach enables efficient and accurate computation of molecular graph parameters.

Step 4: Molecular Graph Visualization
The **Python Turtle Graphics** library is employed for graphical visualization of molecular structures.
The Turtle module is used to
- Draw vertices representing atoms,
- Draw edges representing chemical bonds,
- Label individual atoms,
- Display the complete molecular graph.

The visualization provides an interactive representation of the molecular structure and improves understanding of graph-theoretical concepts.

Step 5: Computation of Topological Indices
After constructing the molecular graph, several important topological indices are computed using Python algorithms.
The computed indices include
- First Zagreb Index(M_1)
- Second Zagreb Index (M_2)
- Wiener Index(W)
- Randić Index (R)

These indices quantify molecular branching, connectivity, and structural characteristics and are widely applied in QSAR and QSPR studies.

Step 6: Calculation of Graph Parameters.
In addition to topological indices, important graph-theoretical parameters are computed. These include
- Degree of each vertex
- Eccentricity
- Radius
- Diameter
- Shortest path distances

These parameters describe the connectivity, compactness, and structural properties of the molecular graph.

Step 7: Physicochemical and Electronic Descriptor Analysis
The methodology further incorporates the analysis of molecular descriptors associated with the selected chemical compound.
The physicochemical descriptors considered include
- Molecular weight
- Boiling point
- Melting point
- Density
- Surface characteristics
- Solubility
- LogP value

Electronic descriptors related to molecular stability and electron distribution are also studied to understand the chemical behaviour of the molecule.

Step 8:Analysis and Validation of Results
The computed graph indices and molecular descriptors are analysed and compared with established theoretical values reported in the literature. The obtained results are used to verify the correctness of the computational model and demonstrate the applicability of Python in chemical graph analysis.
The proposed methodology provides a simple, accurate, and computationally efficient framework for modelling chemical structures using graph theory and can be extended to

larger molecular networks in cheminformatics, computational chemistry, pharmaceutical research, and materials science.

Result And Discussion

Eccentricity

Theoretical: Benzene

Bond of hydrogen atom

H1→H6=1

The degree of vertex

D(H1→H6)=1

Eccentricity of Carbon atom

C1→C6=4

E(C1→C6)=4

Eccentricity of Hydrogen atom

H1→H6=5

E(H1→H6)=5

Radius of Graph=4

Diameter of the Graph=5

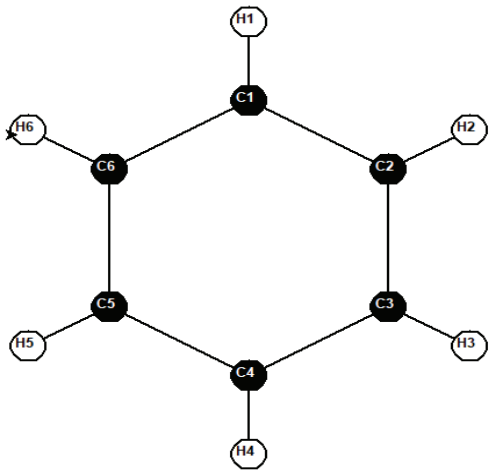
Program Code:

```
import turtle
import math
from collections import deque
graph = {
   'C1': ['C2', 'C6', 'H1'],
   'C2': ['C1', 'C3', 'H2'],
   'C3': ['C2', 'C4', 'H3'],
   'C4': ['C3', 'C5', 'H4'],
   'C5': ['C4', 'C6', 'H5'],
   'C6': ['C5', 'C1', 'H6'],
   'H1': ['C1'],
   'H2': ['C2'],
   'H3': ['C3'],
   'H4': ['C4'],
   'H5': ['C5'],
   'H6': ['C6']
}
positions = {
   'C1': (0, 140),
   'C2': (120, 70),
   'C3': (120, -70),
   'C4': (0, -140),
   'C5': (-120, -70),
   'C6': (-120, 70),
   'H1': (0, 220),
   'H2': (190, 110),
   'H3': (190, -110),
   'H4': (0, -220),
   'H5': (-190, -110),
   'H6': (-190, 110)
}
screen = turtle.Screen()
screen.title("Benzene C6H6")
t = turtle.Turtle()
t.speed(0)
t.pensize(2)
drawn = set()
for vertex in graph:
for neighbor in graph[vertex]:
edge = tuple(sorted([vertex, neighbor]))
      if edge not in drawn:
      x1, y1 = positions[vertex]
x2, y2 = positions[neighbor]
          t.penup()
```

```
            t.goto(x1, y1)
            t.pendown()
            t.goto(x2, y2)
            drawn.add(edge)
for vertex in positions:
x, y = positions[vertex]
t.penup()
t.goto(x, y - 15)
t.pendown()
    if vertex.startswith('C'):
    t.fillcolor("black")
    else:
    t.fillcolor("white")
    t.begin_fill()
    t.circle(15)
    t.end_fill()
    t.penup()
    t.goto(x - 10, y - 5)
    if vertex.startswith('C'):
        t.color("white")
        else:
        t.color("black")
        t.write(vertex, font=("Arial", 10, "bold"))
        t.color("black")
        print("\nDegree of Vertices:")
        for vertex in graph:
        degree = len(graph[vertex])
        print(f"Degree({vertex}) = {degree}")
def bfs_distances(start):
    distances = {v: math.inf for v in graph}
    distances[start] = 0
    queue = deque([start])
    while queue:
    current = queue.popleft()
    for neighbor in graph[current]:
    if distances[neighbor] == math.inf:
    distances[neighbor] = distances[current] + 1
    queue.append(neighbor)
    return distances
    eccentricity = {}
    print("\nEccentricity:")
    for vertex in graph:
    distances = bfs_distances(vertex)
    ecc = max(distances.values())
    eccentricity[vertex] = ecc
```

```
print(f"Eccentricity({vertex}) = {ecc}")
radius = min(eccentrprint("\nRadius =", radius)
diameter = max(eccentricity.values())
print("Diameter =", diameter)
turtle.done()
eccentricity.values())
```

Output:



Degree of vertices	Eccentricity
Degree(C1)=3	Eccentricity(C1)=4
Degree(C2)=3	Eccentricity(C2)=4
Degree(C3)=3	Eccentricity(C3)=4
Degree(C4)=3	Eccentricity(C4)=4
Degree(C5)=3	Eccentricity(C5)=4
Degree(C6)=3	Eccentricity(C6)=4
Degree(H1)=1	Eccentricity(H1)=5
Degree(H2)=1	Eccentricity(H2)=5
Degree(H3)=1	Eccentricity(H3)=5
Degree(H4)=1	Eccentricity(H4)=5
Degree(H5)=1	Eccentricity(H5)=5
Degree(H6)=1	Eccentricity(H6)=5

Radius=4
Diameter=5

Theoretical:

Randic Index

The Randic index of a graph is defined as half of the some of the matrix elements of its Randic matrix.

Mathematically defined as,

$R(G) = \Sigma_{u,v \in E(G)} \frac{1}{\sqrt{d(u)d(v)}}$

cyclobutane

$R(G) = \Sigma_{u,v \in E(G)} \frac{1}{\sqrt{d(u)d(v)}}$

Consider the carbon edges the degree of each carbon atom.

$d(u) = 4, \quad d(v) = 4$

$d(u) \times d(v) = 4 \times 4 = 16$

$R(G) = \Sigma_{u,v \in E(G)} \frac{1}{\sqrt{d(u)d(v)}}$

$= \Sigma \frac{1}{\sqrt{4 \times 4}} = \frac{1}{\sqrt{16}} = \frac{1}{4}$

Now, for each edge of carbon:

$4 \times \frac{1}{4} = 1$

$R(G) = 1$

Consider the hydrogen edges for each carbon atom.

$d(u) = 4, \quad d(v) = 1$

$R(G) = \Sigma_{u,v \in E(G)} \frac{1}{\sqrt{4 \times 1}} = \frac{1}{4} = \frac{1}{2}$

For each edge of hydrogen:8

$R(G) = 8 \times \frac{1}{4} = 4$

$R(G) = 1 + 4$

$R(G) = 5$

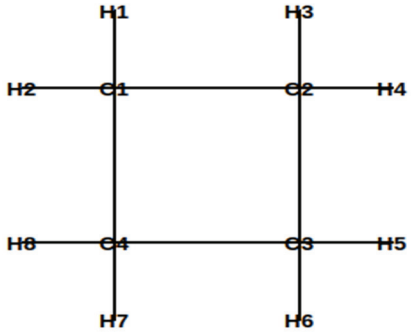
Program Code

```
import turtle
import math
graph = {
"C1": ["C2", "C4", "H1", "H2"],
"C2": ["C1", "C3", "H3", "H4"],
"C3": ["C2", "C4", "H5", "H6"],
"C4": ["C3", "C1", "H7", "H8"],
"H1": ["C1"],
"H2": ["C1"],
"H3": ["C2"],
```

```
"H4": ["C2"],
"H5": ["C3"],
"H6": ["C3"],
"H7": ["C4"],
"H8": ["C4"]
}
pos =
{    "C1": (-60, 60),
"C2": (60, 60),
"C3": (60, -60),
"C4": (-60, -60),
"H1": (-60, 120),
"H2": (-120, 60),
"H3": (60, 120),
"H4": (120, 60),
"H5": (120, -60),
"H6": (60, -120),
"H7": (-60, -120),
"H8": (-120, -60)
}
t = turtle.Turtle()
t.speed(0)
t.pensize(2)
drawn = set()
for u in graph:
for v in graph[u]:
if (v, u) not in drawn:
t.penup()
t.goto(pos[u])
t.pendown()
t.goto(pos[v])
drawn.add((u, v))
for atom, (x, y) in pos.items():
t.penup()
t.goto(x, y - 10)
t.write(atom, align="center",
font=("Arial", 11, "bold"))
t.hideturtle()
R = 0
done = set()for u in graph:
for v in graph[u]:
if (v, u) not in done:
du = len(graph[u])
dv = len(graph[v])
R += 1 / math.sqrt(du * dv)
```

```
done.add((u, v))
print("Randic Index of Cyclobutane (C4H8) =", round(R, 4))
turtle.done()
```

OUT PUT



Randic index of cyclobutane (C4H8)= 5.0

Wiener Index
Theoretically:($C_{10}H_8$)
Distance number of pairs contribution

1	11	11
2	14	28
3	12	36
4	7	28
5	1	5

Total:
W(G)=11+28+36+28+5=108
W(G)=108
Program code

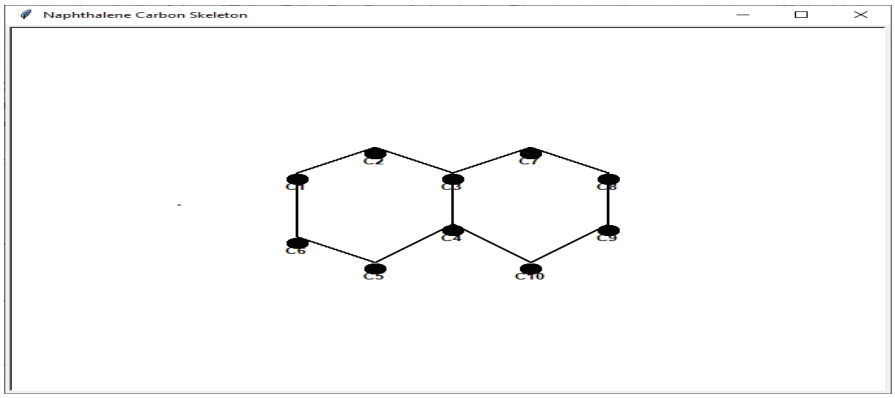
```
import turtle

graph = {
    "C1": ["C2", "C6"],
    "C2": ["C1", "C3"],
    "C3": ["C2", "C4", "C7"],
    "C4": ["C3", "C5", "C10"],
    "C5": ["C4", "C6"],
    "C6": ["C5", "C1"],
    "C7": ["C3", "C8"],
    "C8": ["C7", "C9"],
```

```
 "C9": ["C8", "C10"],
 "C10": ["C9", "C4"]
 }
 "C1": (-120, 60),
 "C2": (-60, 100),
 "C3": (0, 60),
 "C4": (0, -20),
 "C5": (-60, -80),
 "C6": (-120, -40),
 "C7": (60, 100),
 "C8": (120, 60),
 "C9": (120, -20),
 "C10": (60, -80)
}
screen = turtle.Screen()
screen.title("Naphthalene Carbon Skeleton")
t = turtle.Turtle()
t.speed(0)
t.pensize(2)
drawn = set()
for u in graph:
for v in graph[u]:
if (v, u) not in drawn:
t.penup()
t.goto(pos[u])
t.pendown()
t.goto(pos[v])
drawn.add((u, v))
for atom, (x, y) in pos.items():
 t.penup()
 t.goto(x, y - 10)
 t.dot(18, "black")
 t.goto(x, y - 30)
 t.write(atom, align="center", font=("Arial", 10, "bold"))
t.hideturtle()
vertices = list(graph.keys())
n = len(vertices)
INF = 999
dist = [[INF] * n for _ in range(n)]
for i in range(n):
dist[i][i] = 0
for i, u in enumerate(vertices):
for v in graph[u]:
j = vertices.index(v)
 dist[i][j] = 1
```

```
for k in range(n):
for i in range(n):
for j in range(n):
if dist[i][j] > dist[i][k] + dist[k][j]:
dist[i][j] = dist[i][k] + dist[k][j]
W = 0
for i in range(n):
for j in range(i + 1, n):
W += dist[i][j]
print("Wiener Index of Naphthalene (Carbon-Carbon) =", W)
turtle.done()
```

OUT PUT



Wiener index of naptalene 109

- My project studies the computational modelling of chemical graphs using Python based on chemical graph theory .Choosing the such as Benzene (C6H6),are represented by the carbon carbon skeletons similarly using the Wiener Index ,Zagreb Index are theoretically calculated for these carbon carbon skeleton structure computationally verified using Python.This approach provides a consistent framework or analyzing the connectivity ,branching,and distance properties of chemical structures through graph.

Theoretical:

First zagreb index(M_1)

Degree of carbon atoms:

*number of carbon atoms have degree 2

* 2 based carbon atoms have degree 3

M_1=8(2^2)+2(3^2)

=8(4)+2(9)

=32+18,$M_{1=50}$

Second zagreb index(M_2)

Edges:

* 8 edges join degree-2 and degree 2-vertices

8(2×2)=8(4)=32

*2 edges join degree-2 and degree-3 vertices
2(2×3)=12
* 1 edge join the two degree-3 vertices
3×3=9
Total
M_2=32+12+9=53,$M_{2=57}$

Program Code

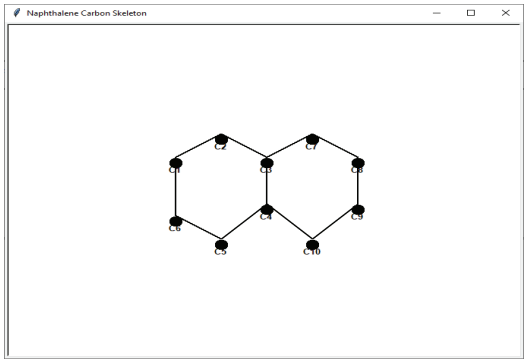
```
import turtle
graph = {
   "C1":["C2","C6"],
   "C2":["C1","C3"],
   "C3":["C2","C4","C7"],
   "C4":["C3","C5","C10"],
   "C5":["C4","C6"],
   "C6":["C5","C1"],
   "C7":["C3","C8"],
   "C8":["C7","C9"],
   "C9":["C8","C10"],
   "C10":["C9","C4"]
}
pos = {
   "C1":(-120,60),
   "C2":(-60,100),
   "C3":(0,60),
   "C4":(0,-20),
   "C5":(-60,-80),
   "C6":(-120,-40),
   "C7":(60,100),
   "C8":(120,60),
   "C9":(120,-20),
   "C10":(60,-80)
}
screen = turtle.Screen()
screen.title("Naphthalene Carbon Skeleton")
t = turtle.Turtle()
t.speed(0)
t.pensize(2)

drawn = set()
for u in graph:
for v in graph[u]:
edge = tuple(sorted((u, v)))
if edge not in drawn:
t.penup()
 t.goto(pos[u])
```

```
 t.pendown()
 t.goto(pos[v])
 drawn.add(edge)
for atom, (x, y) in pos.items():
 t.penup()
 t.goto(x, y-10)
 t.dot(18, "black")
 t.goto(x, y-30)
 t.write(atom, align="center", font=("Arial", 10, "bold"))
t.hideturtle()
degree = {}
for v in graph:
    degree[v] = len(graph[v])
M1 = 0
for v in graph:
    M1 += degree[v] ** 2
visited = set()
M2 = 0
for u in graph:
for v in graph[u]:
edge = tuple(sorted((u, v)))
if edge not in visited:
visited.add(edge)

            M2 += degree[u] * degree[v]
print("Vertex Degrees")
for v in degree:
print(v, ":", degree[v])
print("\nFirst Zagreb Index (M1) =", M1)
print("Second Zagreb Index (M2) =", M2)
turtle.done()
```

Out Put



Zagreb index of naptalene M1=50,M2=57
Theoretical:Cyclopentane
(C_5H_{10})
Now by using distance matrix for wiener index
W(G)=$\sum_{uv\epsilon E(G)} d(uv)$
Now add the distance about the diagonal of matric
Here 2 and 1 are repeating distance
Then ,distance of 1:5 pairs=5×1=5
Distance of 2:5 pairs=5×2=10
Now winer index is
W(G)=5+10=15
W(G)=15

PROGRAM CODE

```
import turtle
import math
from collections import deque

Define the Graph Structure
graph = {
   "C1": ["C2", "C5"],
   "C2": ["C1", "C3"],
   "C3": ["C2", "C4"],
   "C4": ["C3", "C5"],
   "C5": ["C4", "C1"]
}
Setup Turtle Window
screen = turtle.Screen()
screen.setup(width=800, height=600)
screen.title("Cyclopentane Skeleton & Wiener Index")

t = turtle.Turtle()
t.speed(0)
t.pensize(2)

Calculate Vertex Coordinates (Same logic as your image)
R = 120
carbon_pos = {}
nodes = list(graph.keys())
for i in range(5):
angle = math.radians(90 - i * 72)
x = R * math.cos(angle)
y = R * math.sin(angle)
carbon_pos[f"C{i+1}"] = (x, y)
# 4. Draw Edges (Bonds)
```

```
t.color("black")
for i in range(5):
c1 = f"C{i+1}"
c2 = f"C{(i+1)%5 + 1}"
t.penup()
t.goto(carbon_pos[c1])
t.pendown()
t.goto(carbon_pos[c2])

Draw Atom Nodes and Labels
for node, (x, y) in carbon_pos.items():
 t.penup()
 t.goto(x, y - 10)  # Offset slightly to center the dot
 t.dot(22, "black")
 t.dot(16, "white")

Write node name inside the circle
 t.goto(x - 8, y - 8)
 t.color("black")
 t.write(node, font=("Arial", 10, "bold"))

BFS Function for Shortest Paths
def bfs_distance(graph, start, end):
queue = deque([(start, 0)])
 visited = {start}
 while queue:
 curr, dist = queue.popleft()
 if curr == end:
return dist
for neighbor in graph[curr]:
if neighbor not in visited:
visited.add(neighbor)
queue.append((neighbor, dist + 1))
    return 0

t.penup()
t.goto(-350, 200)
t.write("Pairwise Shortest Distances:", font=("Arial", 14, "bold"))

total_wiener_index = 0
text_y_offset = 170

for i in range(len(nodes)):
for j in range(i + 1, len(nodes)):
 u, v = nodes[i], nodes[j]
```

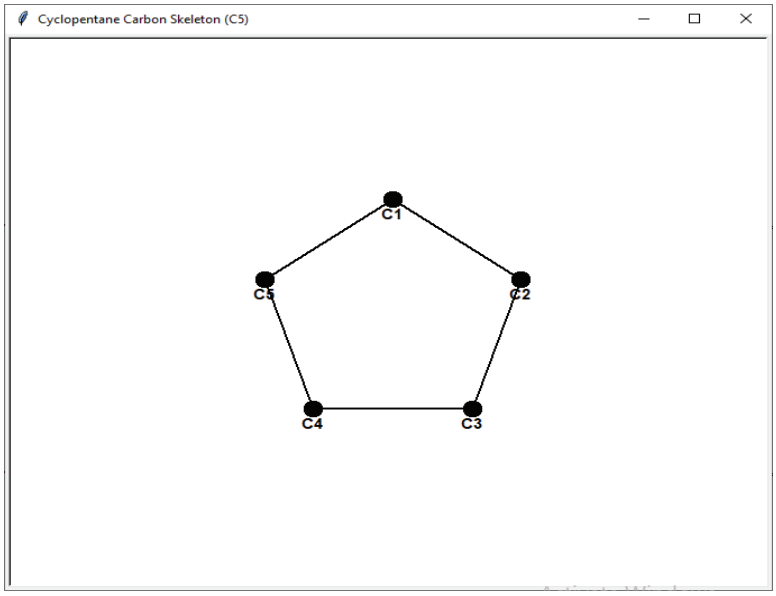
```
dist = bfs_distance(graph, u, v)
total_wiener_index += dist

# Write each pair's distance on the left side of screen
t.goto(-350, text_y_offset)
t.write(f"d({u}, {v}) = {dist}", font=("Arial", 11, "normal"))
text_y_offset -= 20

t.goto(-350, text_y_offset - 20)
t.color("darkred")
t.write(f"Wiener Index (W) = {total_wiener_index}", font=("Arial", 14, "bold"))

t.hideturtle()
screen.mainloop()
```

Out Put



Weiner Index of Cyclopentane = 15

Physiochemical descriptors

Defn**: Physicochemical descriptors are mathematical quantities that represent the physical and chemical properties of a molecule**.

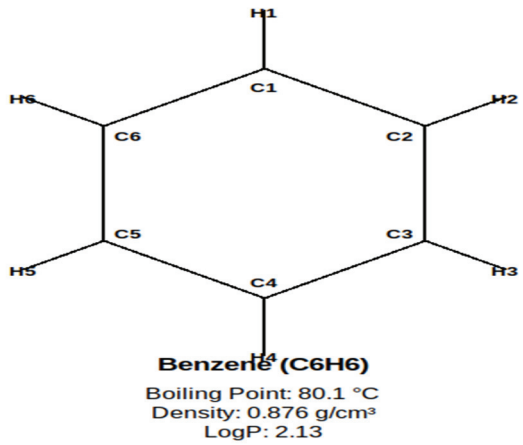
These descriptors help in studying:drug design chemical property prediction molecular stability analysis

Toxicity prediction

Examples of physio chemical discriptors

Dicriptor	Meaning
Molecular weight	Total mass of molecules

Boiling point	Temperature at boiling
Surface area	Area occupied by molecule
Solibility	Ability to dissolve



Graph theory representation
Ex:

Theoretical

Physiochemical descriptors theoretical for benzene (C6H6)

* Number of vertices n =6
  Number vertices of Edges(bonds) m =6
  Degrees of Each Carbon atom =2
  Degene sequence: (2, 2, 2, 2, 2, 2)
  Step2: Calculate . Graph descriptors
  (a) first Zagreb .Index
  $M_1 = \sum d(v)^2$
  Since Every vertex has degree 2:
  $M_1F = 6\times 2^2$
  Mi= 6×4 = 24
  (b) Second Zagreb Index
  For Each Edge
  d(u)× d (v) = 2×2=4
  there are 6 Edges
  M2= 6×4 =24
  (c) Wiener indes the Sum of the shortest distance between are pairs vertices in benzene is
  W= 27

(d) Randic index.
Each Edge Contributes

$$\frac{1}{\sqrt{2*2}} = \frac{1}{2}$$

there are 6 Edges

$$R = 6 * \frac{1}{2} = 3$$

Step3: Use Sample QSPR models
Sample Bailing point model

BP = 20+ 1.5M1 + 0.8W
Substitute the values:
BP == 20 + (1.5×24) +(0.8×27)
B = 20 +36 +21.6
BP = 77.6°C

Experimentalvalue ≈ 80.1°C.
Sample density model
for actual density , remember the physical
Deformation $p=\frac{m}{v}$
A Sample QSPR Equation is

$P = 0.40 + 0.01m + 0.08R$
Substitute :

P=0.40+(0.01*24)+ (0.08×3)
P = 0.40 + 0.24+ 0.2
P =0.88g/cm³
Experimental value ≈0.8768 g /cm³
Sample logp model
Use the sample Equations
logp = 0.10 +0.05W + 0.23R
Substitute
logp = 0.10 + (0.05*27) + (0.23*3)
= 0.10 + 1.35 + 0.69
logp = 2.14
Experimental value ≈ 2.13

Example 2:
Physiochemical descriptor for Verbenone
Step1: Chemical Graph of verbenone vertices(atoms):V=11(10 carbon +1oxygen)
.edges(bonds):E=11

Vertex degree:

Atom	Degree
C1	2
C2	3
C3	3
C4	2
C5	3
C6	3
C7	1
C8	1
C9	1
C10	1
O	1

Step2:First Zagreb Index
Formul
M1=$\sum d(v)2$
d(v) = number of bonds attached to the atom
Calculation
=$2^2 + 3^2 + 3^2 + 2^2 + 3^2 + 3^2 + 1^1 + 1^1 + 1^1 + 1^1 + 1^1$
=4+9+9+4+9+9+1+1+1+1+1
M1=49
* Carbon attached to there atoms →degree = 3
* Oxygen =1attached to one Carbon→degree = 1

Step3: Second Zagreb Index
Formul
M2=$\sum d(u)d(v)$
For eah bond:
. (2×3)=6
. (3×3)=9
. (3×2)=6
. (2×3)=6
. (3×3)=9
. (3×2)=6
. (3×1)=3
. (3×1)=3
. (3×1)=3
. (3×1)=3
. (3×1)=3
Total
6+9+6+6+9+6+3+3+3+3+3=57
Steps 4: Randic index: The Randic index measure molecular branching

$$R(G) = \sum_{uv \epsilon E(G)} \cdot \frac{1}{\sqrt{d(u)d(v)}}$$

Example $\frac{1}{\sqrt{6}}+\frac{1}{\sqrt{9}}+\frac{1}{\sqrt{6}}+$……..
More branching changes the randic index and often influence boiling point and logP

Step 5 : Wiener index

W(G) = Σd(u,v)
Where d(u,v) is the shortest path between every pair of vertices
It measures the overall molecular size and connectivity
For the verbenone graph, W ≈88
(The exact value depends on the graph representation used.)

Step 6: QSPR Equations

Relationship with physio chemical properties(QSPR)
A typical QSPR(Quantitative structure property relationship)
Boiling point
BP=a+bM_1+cW
Where
P= physio chemical property
M_1=zagreb index
R= randic index
W=wiener index
A,b,c,d=constant obtained from regression using experimental data
For example
Boiling point=a+bM_1+cW
BP=65+1.15(49)+1.20(88)
=65+56.35+105.6
BP≈227^0C

Density
ρ=a+bR+cM_2
Example
ρ=0.40+0.05(3.87)+0.006(57)
=0.40+0.193+0.365
ρ≈0.958g/cm^3

Log P
Log P=a+bM_1-cR
Example
Log P=0.80+0.06(49)-0.28(3.87)
=0.80+2.94-1.08
Log P≈2.8

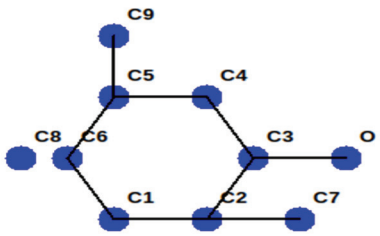
Program Code:

```
import turtle
molecule = "Verbenone (C10H14O)"
boiling_point = 227.0
density = 0.958
```

```
logP = 2.8
screen = turtle.Screen()
screen.title("Physicochemical Descriptors")
t = turtle.Turtle()
t.speed(3)
t.pensize(2)
def atom(x, y, label):
    t.penup()
    t.goto(x, y)
    t.dot(20, "blue")
    t.goto(x + 10, y + 10)
    t.write(label, font=("Arial", 10, "bold"))
def bond(x1, y1, x2, y2):
    t.penup()
    t.goto(x1, y1)
    t.pendown()
    t.goto(x2, y2)
atoms = {
    "C1": (0, 0),
    "C2": (60, 0),
    "C3": (90, 50),
    "C4": (60, 100),
    "C5": (0, 100),
    "C6": (-30, 50),
    "C7": (120, 0),
    "C8": (-60, 50),
    "C9": (0, 150),
    "O": (150, 50)
}
for label, (x, y) in atoms.items():
    atom(x, y, label)
bonds = [
    ("C1", "C2"),
    ("C2", "C3"),
    ("C3", "C4"),
    ("C4", "C5"),
    ("C5", "C6"),
    ("C6", "C1"),
    ("C2", "C7"),
    ("C5", "C9"),
    ("C3", "O")
]
for a, b in bonds:
    x1, y1 = atoms[a]
    x2, y2 = atoms[b]
```

```
    bond(x1, y1, x2, y2)
t.penup()
t.goto(-200, -150)
t.write("Molecule: {}".format(molecule), font=("Arial", 12, "bold"))
t.goto(-200, -170)
t.write("Boiling Point: {} °C".format(boiling_point), font=("Arial", 11, "normal"))
t.goto(-200, -190)
t.write("Density: {} g/cm³".format(density), font=("Arial", 11, "normal"))
t.goto(-200, -210)
t.write("LogP: {}".format(logP), font=("Arial", 11, "normal"))
t.hideturtle()
screen.mainloop()
```

Out Put



Molecule: Verbenone (C10H14O)
Boiling Point: 227.0 °C
Density: 0.958 g/cm³
LogP: 2.8

Electronic Descriptors

definition : electronic descriptors describe the electronic behavior of molecules.
They are useful in:
. semiconductor research
. drug activity prediction
. quantum chemistry
. molecular orbital analysis
examples of electronic descriptors

Descriptors	**Meaning**
electron density	distribution of electrons
dipole moment	charge separation
ionization energy	energy to remove electronic
electronegativity	attraction of electronic
HOMO-LUMO gap	energy dofference

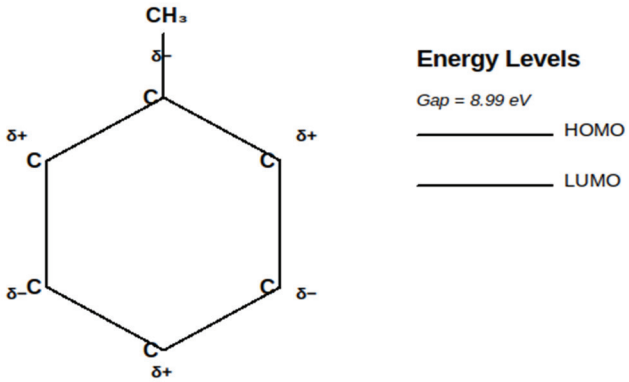
Program code 2:

```
import turtle
import math
molecule = "Toluene (C7H8)"
HOMO = -8.82
LUMO = 0.17
energy_gap = 8.99
dipole = 0.37
screen = turtle.Screen()
screen.title("Electronic Descriptors")
pen = turtle.Turtle()
pen.speed(0)
pen.pensize(2)
radius = 100
points = []
for i in range(6):
   angle = math.radians(90 - i*60)
   x = radius * math.cos(angle)
   y = radius * math.sin(angle)
   points.append((x,y))
def draw(p1,p2):
   pen.penup()
   pen.goto(p1)
   pen.pendown()
   pen.goto(p2)
for i in range(6):
   draw(points[i],points[(i+1)%6]
x,y = points[0]
pen.penup()
pen.goto(x,y)
pen.pendown()
pen.goto(x,y+50)
pen.penup()
pen.goto(x-12,y+55)
pen.write("CH₃",
        font=("Arial",12,"bold"))
for x,y in points:
 pen.penup()
 pen.goto(x-8,y-10)
 pen.write("C",
          align="center",
          font=("Arial",12,"bold"))
charges = ["δ−","δ+","δ−","δ+","δ−","δ+"]
for i,(x,y) in enumerate(points):
 r = math.sqrt(x*x+y*y)
```

```
 ux = x/r
 uy = y/r
 pen.penup()
 pen.goto(x+ux*25,y+uy*25)
 pen.write(charges[i],
           align="center",
           font=("Arial",10,"bold"))
pen.penup()
pen.goto(190,120)
pen.write("Energy Levels",
        font=("Arial",14,"bold"))
pen.goto(190,70)
pen.pendown()
pen.goto(290,70)
pen.penup()
pen.goto(190,30)
pen.pendown()
pen.goto(290,30)
pen.penup()

pen.goto(300,65)
pen.write("HOMO",
        font=("Arial",11,"normal"))
pen.goto(300,25)
pen.write("LUMO",
        font=("Arial",11,"normal"))
pen.goto(190,90)
pen.write("Gap = %.2f eV"%energy_gap,
        font=("Arial",10,"italic"))
pen.goto(-180,-180)
pen.write("Molecule : "+molecule,
        font=("Arial",13,"bold"))
pen.goto(-180,-205)
pen.write("HOMO : %.2f eV"%HOMO,
        font=("Arial",11,"normal"))
pen.goto(-180,-225)
pen.write("LUMO : %.2f eV"%LUMO,
        font=("Arial",11,"normal"))
pen.goto(-180,-245)
pen.write("Energy Gap : %.2f eV"%energy_gap,
        font=("Arial",11,"normal"))
pen.goto(-180,-265)
pen.write("Dipole Moment : %.2f D"%dipole,
        font=("Arial",11,"normal"))
pen.hideturtle()
```

```
screen.mainloop()
```



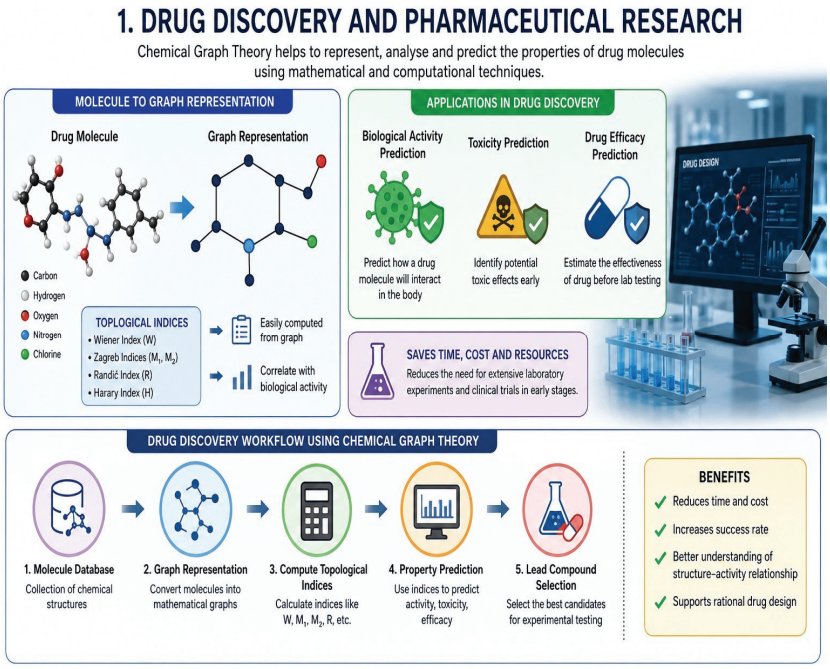
Applications of Chemical Graph Theory in Real Life

Chemical Graph Theory has emerged as an important interdisciplinary field that combines graph theory, chemistry and computational science to analyse molecular structures and predict their properties. By computing graph parameters, topological indices, and molecular descriptors, it provides valuable information for various scientific and industrial applications. Some major real-life application are as follows:

- **Drug Discovery and Pharmaceutical Research**

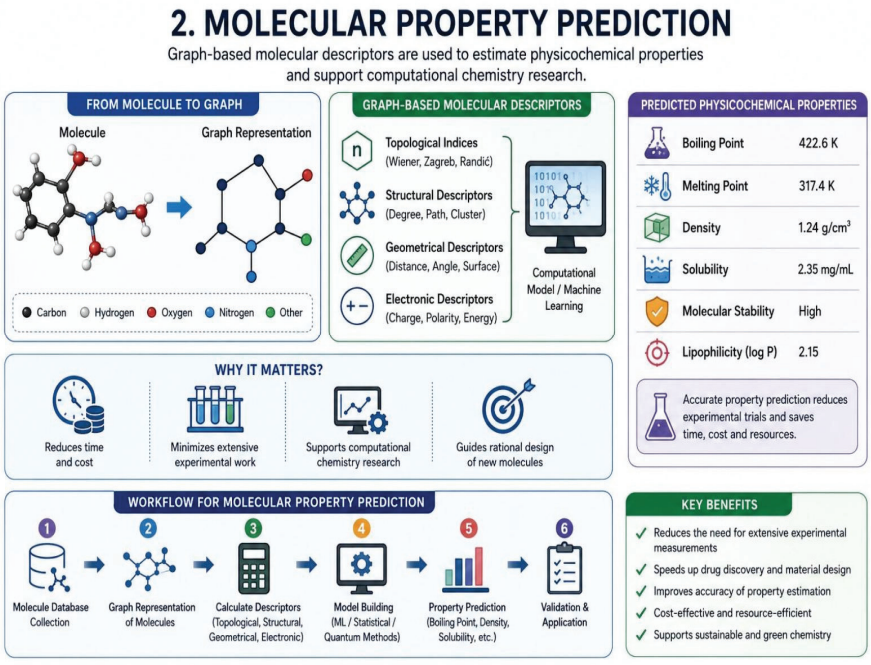
Chemical Graph Theory is extensively used to represent drug molecular as graph and analyse their structural characteristics.

Topological indices such as the Wiener, Zagreb, and Randic indices help predict biological activity, toxicity, and drug efficacy before laboratory testing.



➢ **Molecular Property Prediction**

Graph -based molecular descriptors are used to estimate physiochemical properties such as boiling point, melting point, density, solubility, molecular stability, and lipophilicity.These predictions reduce the need for extensive experimental measurement and support computational chemistry research.



➢ **Chemical Structure Analysis**

Degree, eccentricity, radius, diameter, and topological indices help analyse molecular connectivity, branching, and overall structural complexity.

Researchers use these parameters to compare molecular structures and classify chemical compounds efficiently.

➢ **Materials Science and Nanotechnology**

Chemical graph theory assists in studying advanced materials such as graphene, nanotubes, polymers, and molecular crystals.

Structural descriptors help predict mechanical strength, thermal stability, conductivity, and durability, supporting the design of new functional materials.

➢ **Environmental and Toxicity Assessment**

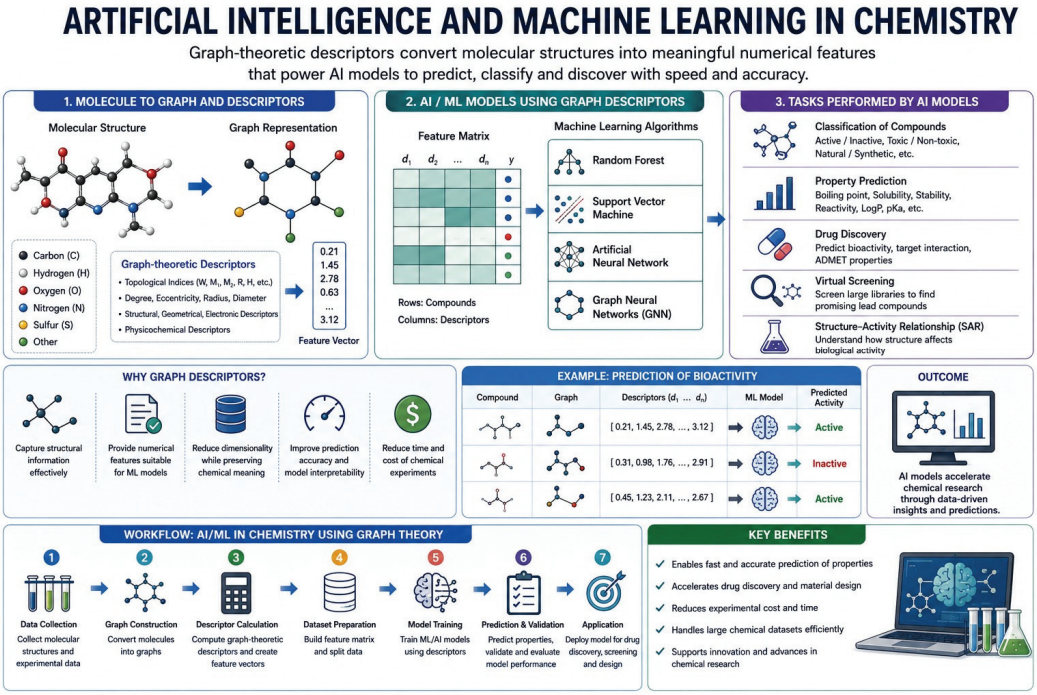
Molecular descriptors are used to predict the environmental behaviour and toxicity of chemical compounds.

This enables scientist to evaluate hazardous chemical, pesticides, and industrial pollutants before large-scale production or environmental release.

➢ **Artificial Intelligance and Machine Learning in Chemistry**

Graph-theoretical descriptors serve as numerical features, for machine learning algorithms.

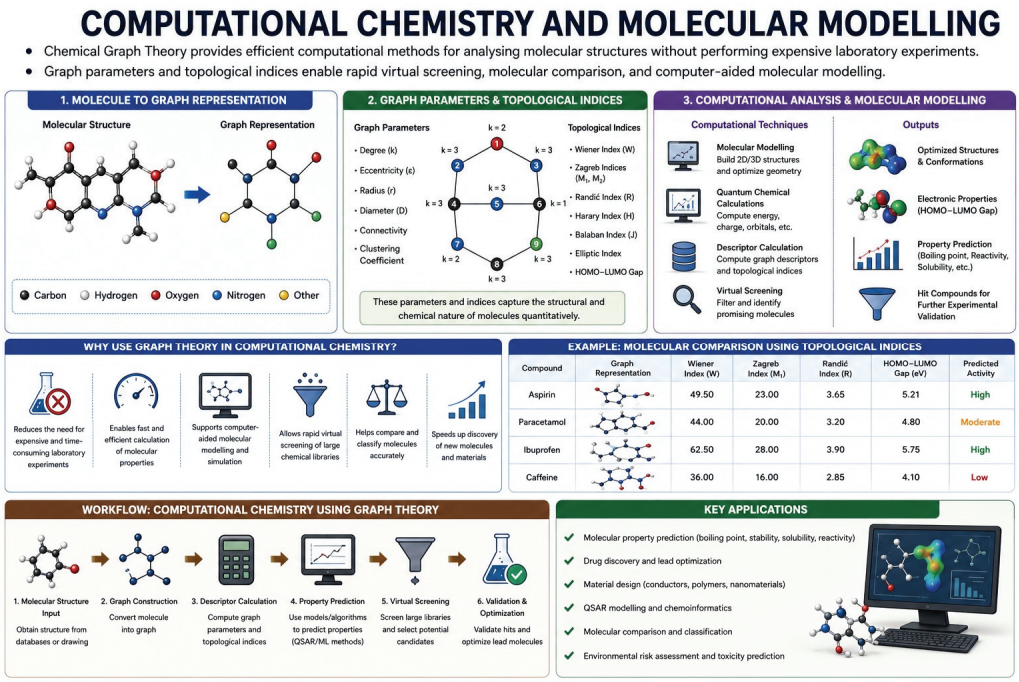
AI models use these descriptors to classify compounds, predict molecular properties,descover new drugs, and accelerate chemical research through data-driven analysis.



➢ **Computational Chemistry and Molecular Modelling**

Chemical graph theory provides efficient computational methods for analysing Molecular structures without performing expensive laboratory experiments.

Graph parameters and topological indices enable rapid virtual screening, molecular comparison, and computer-aided molecular modelling.



Conclusion

This study demonstrates the effective use of Chemical graph theory and python programming for computational modelling analysis,and visualization of molecular structures.Chemical compounds are represented as graph in which atoms correspond to vertices and chemical bonds to edges. Important graph parameters such as vertex degree,eccentricity, radius,and diameter,together with the first and second zagreb indices,wiener index,and randic index,are computed to describe molecular connectivity,branching,size,and structural characteristics.

The use of python IDEL and Turtle Graphics provides a simple,low-cost,and accessible framework for constructing and visualizing chemical graphs without requiring specialized cheminformatics software.Recent research confirms that topological indices continue to play an important role in QSPR and QSAR modelling, where they can be combined with statistical and machine-learning techniques for predicting molecular properties and biological activity.

Overall the study establishes a clear computational relationship between molecular structures,graph parameters,topological indices,and molecular descriptors .the proposed approach is particularly useful for education and provides a foundation for further research in chem-informatics,computational chemistry,drug discovery, and materials science. furture work can extend the framework by incorporating larger molecular datasets,additional topological indices,and machine-learning-based molecular property prediction.

Reference

[1] A Alqesmah A. Alwardi, cangule, in on the entire ABC index of graphs, proceedings of the Jangjeon mathematical society 23(1)(2020),39-51

[2] Abdurahim J. Qudsi muawanah and salwa.”Indeks Harmonika Randic index, dan Gutman dari Graf Koprima prime untuk Grup Bilangan Bulat Modulo”, J urnal Differensial vol 7 no (2025)

[3] Ahmed, W.,Ali,K.,Zaman, S., Ahmad,F.&Ashebo,M.A. Molecular insights into anti-Alzheimer drug through eccentricity-based predictive mathematical modeling using regression and QSPR analysis.j.Mol. 13, 1(2024)

[4] Akbar Ali,Tomislav Doslik,Zahid Raza -on Wiener-Type Topological Indices and Their Reciprocals(2025)

[5] Arockiaraj M., Greeni A.B., Kalaam A. R. A., Aziz T., Alharbi M. (2024). Mathematical modeling for prediction of physicochemical characteristics of cardiovascular drugs via modified reverse degree topological indices.Eur.Phys. J. E 47, 53.1 0.1140/epje/s10189-024-00446-3

[6] Dragan,F.F,.&Ducoffe,G.(2024) Matrix Graphs:Radius, Diameter, and all Eccentricities Journal:Algorithmica.86,2092-2129

[7] Hayat, S. & Asmat, F.Sharp bounds on the generalized multiplicative first Zagreb index of graph with application to QSPR modeling. Mathematics11(10), 2245 (2023)

[8] Heider Abdulsada,Faik Mayah -Topological Modelling of Chemichal Graphs Using Wiener and Zagreb Indices(2026)

[9] Martinez-Martinez, C.T., Mendez-Bermudez, J. M. Computational and analytical studies of the haharmonic index in Math. Comput. Chem 85, 395-426(2021)

[10] Niko Tratnik,Petra Zigret Pletersek.Zagreb Root indices of Graphs with chemical applications (2024)

[11] Sarkarai,D.&Desikan.k.(2024)computational analysis of diameter,Eccentricity based and Hyper diameter Eccentricity based indices for linear saturated monocarboxylic acid Journal:journal of the national science foundation of srilanka,52(3)