📘 Module-5: Root Finding and Symbolic Computing using SciPy and SymPy

🔹 Introduction

Root-finding and symbolic mathematics are crucial in engineering problem-solving. Python libraries like **SciPy** and **SymPy** allow us to:

- Find **roots of equations** using numerical methods such as **Bisection**, **Brent's**, and **Newton-Raphson**
- Perform **symbolic computation** such as derivatives, integrals, limits, simplification, and solving equations analytically

This module introduces how to solve civil engineering problems both **numerically (using SciPy)** and **symbolically (using SymPy)**.

💻 Python Programmes

🔹 1. Find root using Bisection method (`bisect`)

```
from scipy.optimize import bisect
def f(x): return x**3 - x - 2
root = bisect(f, 1, 2)
print("Root using Bisection:", root)
```

🔹 2. Find root using Brent's method (`brentq`)

```
from scipy.optimize import brentq
def f(x): return x**3 - x - 2
root = brentq(f, 1, 2)
print("Root using Brent’s method:", root)
```

🔹 3. Find root using Newton-Raphson method (`newton`)

```
from scipy.optimize import newton
def f(x): return x**3 - x - 2
def df(x): return 3*x**2 - 1
root = newton(f, x0=1.5, fprime=df)
print("Root using Newton-Raphson:", root)
```

🔹 4. Basic SymPy Setup and Symbol Declaration

```
from sympy import symbols
x = symbols('x')
```

```
print("Symbol defined:", x)
```

🔹 5. Find Derivative of a Function

```
from sympy import diff, symbols
x = symbols('x')
f = x**3 + 2*x**2 + x
df = diff(f, x)
print("Derivative of f:", df)
```

🔹 6. Find Definite and Indefinite Integrals

```
from sympy import integrate, symbols
x = symbols('x')
f = x**2
indef = integrate(f, x)
definite = integrate(f, (x, 0, 2))
print("Indefinite:", indef)
print("Definite (0 to 2):", definite)
```

🔹 7. Evaluate Limit of a Function

```
from sympy import limit, symbols
x = symbols('x')
expr = (x**2 - 1)/(x - 1)
l = limit(expr, x, 1)
print("Limit as x → 1:", l)
```

🔹 8. Simplify an Expression

```
from sympy import simplify, symbols
x = symbols('x')
expr = (x**2 - 1)/(x - 1)
simple = simplify(expr)
print("Simplified expression:", simple)
```

🔹 9. Substitute a Value into Expression

```
from sympy import symbols
x = symbols('x')
expr = x**2 + 3*x + 2
val = expr.subs(x, 2)
print("Expression value at x = 2:", val)
```

🔹 10. Solve Algebraic Equation Symbolically

```
from sympy import solve, symbols
```

```
x = symbols('x')
eq = x**2 - 5*x + 6
roots = solve(eq, x)
print("Roots of the equation:", roots)
```

🔹 11. Solve Simultaneous Equations

```
from sympy import symbols, Eq, solve
x, y = symbols('x y')
eq1 = Eq(x + y, 10)
eq2 = Eq(x - y, 4)
sol = solve((eq1, eq2), (x, y))
print("Solution:", sol)
```

🔹 12. Solve a First-order Differential Equation

```
from sympy import Function, dsolve, Derivative, Eq, symbols
x = symbols('x')
y = Function('y')
ode = Eq(Derivative(y(x), x), x + 2)
sol = dsolve(ode, y(x))
print("Solution of ODE:", sol)
```

🔹 13. Solve a Second-order Differential Equation

```
from sympy import Function, dsolve, Derivative, Eq, symbols
x = symbols('x')
y = Function('y')
ode = Eq(Derivative(y(x), x, x) - 3*Derivative(y(x), x) + 2*y(x), 0)
sol = dsolve(ode, y(x))
print("2nd Order ODE Solution:", sol)
```

🔹 14. Real-life Civil Engg Example: Beam Load Derivative

```
from sympy import symbols, diff
x = symbols('x')
M = -10*x**2 + 50*x  # Bending Moment Equation
V = diff(M, x)       # Shear Force is derivative of Moment
print("Shear Force (V):", V)
```

🔹 15. Area under Stress-Strain Curve Symbolically

```
from sympy import symbols, integrate
ε = symbols('ε')
σ = 200*ε  # Assume linear stress-strain curve
area = integrate(σ, (ε, 0, 0.002))  # Up to 0.2% strain
print("Area under curve (Strain Energy):", area)
```