Introduction to Numerical Methods and Errors

Objective of The Lecture:

The objective of this lecture is to introduce you to the basic concepts of **numerical methods** and explain their importance in solving real-world civil engineering problems. By the end of the lecture, you should understand:

1. What **numerical methods** are and why they are used in civil engineering.
2. Common types of **errors** that occur in numerical methods, such as round-off and truncation errors.
3. How to calculate **absolute** and **relative errors** to assess the accuracy of a numerical solution.
4. The role of programming tools like **MATLAB** or **Python** in implementing numerical methods.

Lecture Strecture:

Topic	Description
1. Introduction to Numerical Methods	Definition of numerical methods and their role in solving complex mathematical problems in engineering.
2. Importance of Numerical Methods in Civil Engineering	Explanation of how numerical methods are used in structural, geotechnical, and fluid mechanics problems.
3. Types of Errors in Numerical Methods	Overview of round-off errors, truncation errors, and propagation of errors, with examples.
4. Error Analysis: Absolute and Relative Errors	Explanation of how to calculate absolute and relative errors, with real-world examples.
5. Real-World Example: Deflection of a Simply Supported Beam	Example calculation of the deflection of a beam under a load, showing the exact and approximate solutions.
6. Introduction to MATLAB/Python	Brief introduction to MATLAB and Python for solving numerical problems, including basic commands and examples.

1. Introduction to Numerical Methods

What Are Numerical Methods?

Numerical methods are mathematical techniques used to find approximate solutions to problems that cannot be solved exactly using traditional algebraic methods. These methods involve **iterative** approaches, where a solution is gradually refined through a series of calculations until the result is close enough to the true value.

In engineering, many problems result in complex systems of equations, non-linear functions, or differential equations that are either too difficult or impossible to solve analytically. This is where numerical methods come into play — they allow engineers to obtain practical solutions that are "good enough" for engineering purposes.

Why Do We Use Numerical Methods?

In many real-world civil engineering problems, exact solutions are either impossible to find or too complicated to be useful in practice. For example:

- When analyzing the deflection of a complex structure under load, the governing equations may not have a simple, closed-form solution.
- When modeling fluid flow in pipes or groundwater seepage in soils, engineers often encounter complex differential equations that are too difficult to solve by hand.

Numerical methods allow us to overcome these challenges by providing an efficient way to obtain approximate solutions to such problems. The accuracy of these solutions can be adjusted depending on the needs of the project, making numerical methods highly flexible.

Types of Problems Solved by Numerical Methods

Numerical methods are used to solve various types of engineering problems, including but not limited to:

1. **Solving Non-linear Equations**:
   In many engineering problems, equations are not simple polynomials, and analytical methods cannot be used. Numerical methods like **Newton's Method** or the **Bisection Method** can be used to iteratively find roots of non-linear equations.

   Example: Finding the force required to displace a pile driven into soil involves solving a non-linear equation where forces depend on the non-linear behavior of the soil.

2. **Solving Systems of Linear Equations**:
   Large systems of linear equations often arise when analyzing structures using the **Finite Element Method (FEM)**. Numerical methods such as **Gaussian elimination** or **LU decomposition** are used to solve these systems.

Example: When designing a multi-story building, numerical methods are used to calculate the stresses and displacements of various structural elements.

3. **Numerical Differentiation and Integration**:
   Engineers often need to calculate the derivative or integral of a function, especially when dealing with physical processes like heat transfer, fluid flow, or structural deformation. **Numerical differentiation** approximates the derivative, while **numerical integration** estimates the area under a curve.

   Example: In fluid mechanics, numerical integration is used to estimate flow rates through channels, and numerical differentiation is used to calculate velocity or pressure gradients.

4. **Solving Ordinary and Partial Differential Equations (ODEs and PDEs)**:
   Many engineering problems are modeled by differential equations. For example, the bending of a beam or the flow of heat through a material can be described by differential equations. These equations are often too complex to solve analytically, so engineers use numerical methods such as **Euler's method**, **Runge-Kutta methods**, and the **Finite Difference Method (FDM)**.

   Example: The distribution of temperature in a concrete dam subject to solar heating can be modeled by solving a partial differential equation using numerical methods.

Advantages of Numerical Methods

1. **Ability to Handle Complex Problems**:
   Numerical methods can be applied to problems that are too difficult or impossible to solve exactly. For example, when designing a large suspension bridge, the equations governing the behavior of the structure are too complex for analytical solutions, but numerical methods can provide an approximate solution.
2. **Flexibility**:
   Numerical methods can be adapted to a wide variety of problems. Whether you're working with a small structure or a large-scale geotechnical analysis, you can apply numerical methods to get the solution you need.
3. **Improving Accuracy**:
   Numerical methods allow you to control the level of accuracy by increasing the number of iterations or refining the grid in numerical simulations. This is useful when you need a high level of precision for critical engineering projects.
4. **Efficient for Large-Scale Systems**:
   Large-scale systems, such as modeling the stresses in a high-rise building or simulating groundwater flow through a region, involve hundreds or thousands of equations. Numerical methods can handle these systems efficiently using computers.

Challenges of Numerical Methods

While numerical methods are extremely useful, they also come with certain challenges:

1. **Approximation**:
   Numerical methods provide **approximate** solutions, not exact ones. Depending on the method used and the problem's complexity, the accuracy of the solution can vary. Engineers must understand the level of precision required for their specific application.
2. **Computational Cost**:
   Some numerical methods, especially for large-scale problems or complex simulations, can be **computationally expensive**. This means that solving the problem might require significant time and computer resources, especially when high precision is needed.
3. **Convergence and Stability**:
   Not all numerical methods will always converge to a solution. In some cases, a numerical method might fail to find a solution or the solution might not be stable. Engineers must understand the limitations of each method and apply them appropriately.
4. **Error Propagation**:
   Errors introduced at early stages of numerical calculations can accumulate and become significant as the calculations proceed. Understanding how errors propagate is essential to obtaining reliable results.

2. Importance of Numerical Methods in Civil Engineering

Civil engineers often need to solve complex mathematical equations that describe the behavior of structures, fluids, or soils. These problems can be related to:

- **Structural Engineering**: Calculating stresses and deflections in bridges, buildings, and other structures.
- **Geotechnical Engineering**: Estimating the settlement of soil under the foundation of buildings or determining slope stability.
- **Fluid Mechanics**: Modeling how water flows through pipes or channels.

In all these cases, the mathematical models are often too complex to solve with simple algebra. This is where numerical methods come in — they allow us to solve these problems approximately, with a high level of accuracy.

3. Types of Errors in Numerical Methods

Whenever we use numerical methods, the result is an **approximation**, not the exact solution. This means there will always be some amount of **error** in the answer we calculate. Understanding and controlling these errors is important to ensure our results are accurate enough for engineering purposes.

1. Round-Off Errors:

- **Definition**: These errors occur because computers can only store numbers with a limited number of decimal places. When the number is too long, it gets "rounded off," introducing a small error.
- **Example**: Storing π as 3.14159 instead of its exact value creates a small error in calculations involving π.

2. Truncation Errors:

- **Definition**: Truncation errors occur when we approximate an infinite process (like an infinite series) with a finite number of steps.
- **Example**: When calculating an integral using a numerical method (like the Trapezoidal Rule), the exact result is approximated, and a small truncation error is introduced.

3. Propagation of Errors:

- **Definition**: As we perform a series of calculations, small errors can accumulate and grow. This is especially true when using iterative methods where the result of one step is used in the next step.

4. Error Analysis: Absolute and Relative Errors

Error analysis helps us understand how far our numerical solution is from the exact solution.

1. Absolute Error:

- **Definition**: The absolute error is the difference between the exact value and the approximate value.

- **Formula**:

$$\text{Absolute Error} = |\, x_{exact} - x_{approx} \,|$$

- **Example**: If the exact value is $x_{exact} = 2$ and the approximate value is $x_{approx} = 1.98$, the absolute error is:

$$\text{Absolute Error} = |\, 2-1.98 \,| = 0.02$$

2. Relative Error:

- **Definition**: The relative error compares the absolute error to the exact value, often expressed as a percentage.

- **Formula**:

$$\text{Relative Error} = \frac{|x_{exact} - x_{approx}|}{|x_{exact}|} \times 100$$

- **Example**: Using the same values from above ($x_{exact} = 2$, $x_{approx} = 1.98$), the relative error is:

$$\text{Relative Error} = \frac{0.2}{2} \times 100 = 1\ \%$$

In this case, the error is small and acceptable for most engineering applications.

5. Real-World Example: Deflection of a Simply Supported Beam

Let's apply what we've learned about errors to a real-world civil engineering problem.

Problem: Deflection of a Beam Under a Load

A simply supported beam of length **L=6 m** is subjected to a point load **P=10 kN** at the center of the beam. The beam has a modulus of elasticity **E=200 GPa** and a moment of inertia **I=0.0004** m^4.

We can calculate the exact deflection of the beam at its midpoint using the following formula:

$$\delta_{max} = PL^3/48EI$$

Step 1: Exact Calculation

- P=10 kN = 10,000 N
- L=6 m
- $E=200\times10^9$ N/m^2
- I=0.0004 m^4

Substituting the values into the formula:

$$\delta_{max} = 10{,}000\times6^3 \,/\, 48\times200\times10^9\times0.0004$$

Simplifying:

$$\delta_{max} = 2{,}160{,}000/3.84\times10^9 = 5.625\times10^{-4}\text{m}$$

Thus, the exact deflection is δ_{max}= 0.5625 mm.

Step 2: Introducing Truncation Error

Now let's introduce a **truncation error** by using an approximate value for the load, say P=9,500 N , instead of 10,000 N.

The approximate deflection is:

$$\delta_{approx} = 9{,}500\times6^3 \,/\, 48\times200\times10^9\times0.0004$$

This gives:

$$\delta_{approx} = 2{,}052{,}000/\, 3.84\times10^9 = 5.34375\times10^{-4}\text{m}$$

So, the approximate deflection is $\delta_{approx} = 0.5344$ mm

Step 3: Error Calculation

- **Absolute Error**:

$$\text{Absolute Error} = |0.5625 - 0.5344| = 0.0281 \text{ mm}$$

- **Relative Error**:

$$\text{Relative Error} = (\ 0.0281 / 0.5625\ ) \times 100 \approx 5\%$$

6. Introduction to MATLAB/Python

Throughout this course, **MATLAB** or **Python** will be to solve numerical problems like the one above. These are powerful tools for performing calculations, creating plots, and analyzing data.

Examples of Basic Commands:

- **MATLAB Example** (Solving an equation):

```
f = @(x) 3*x^2 - 4*x + 1;
root = fzero(f, 0);  % Finds the root near x = 0
disp(root);
```

- **Python Example** (Solving an equation):

```
from scipy.optimize import fsolve
def f(x):
  return 3*x**2 - 4*x + 1
root = fsolve(f, 0)  # Finds the root near x = 0
print(root)
```