MT308 Numerical Analysis

Girne American University
Faculty of Engineering
Spring 2026
Dr. Kamil Dimililer

Week 1: 9/13 February 2026

1

Numerical Analysis

- **Numerical Analysis** is a branch of mathematics that focuses on developing and analysing **algorithms** for obtaining **numerical solutions** to mathematical problems that are difficult or impossible to solve analytically.
- The primary objective of a **numerical analysis** course is to equip students with the knowledge and skills to solve mathematical problems that arise in engineering and science using **computational methods**.
- **Numerical methods** are the backbone of AI computation. Think of AI as the application, and numerical analysis as part of the infrastructure that enables it.
- Numerical analysis provides the **mathematical** and **computational tools** needed to implement, optimize, and understand **modern AI models**, especially in **machine learning** (ML) and **deep learning** (DL).

Week 1: 9/13 February 2026

2

Numerical Analysis vs ML/AI

- The core mathematical foundations of AI and machine learning are linear algebra, calculus, probability, and statistics.
- Numerical analysis is not a prerequisite for learning or applying AI techniques.
- Most AI users and practitioners can work effectively without formal training in numerical analysis.
- **However, numerical analysis is essential for a deep understanding of algorithms, improving efficiency and stability, and developing robust, advanced ML/AI models**.

3

Computational methods and Algorithms

- A **computational method** refers to the practical application of algorithms to solve mathematical, scientific, or engineering problems, often involving **numerical approximations**.
- An **algorithm** is a **step-by-step procedure** or **a set of rules for solving** a specific problem or performing a computation. It is an abstract, logical description of how a task should be carried out.
- Every computational method is implemented through one or more **algorithms.** Computational methods often involve choosing or designing appropriate algorithms.
- In practice, computational methods translate **theory** into **code** by using algorithms that are optimised for speed, stability, and resource usage

4

Where do we use computational methods?

Tool	Field	Key Numerical Methods
AutoCAD	General design	Geometry, matrix operations
ANSYS	Mechanical, thermal	FEM, CFD, modal analysis
MATLAB/Simulink	All	ODEs, PDEs, optimization
COMSOL	Multiphysics	FEM, nonlinear solvers
SolidWorks	Product design	FEM
Abaqus	Structural	Advanced FEM
OpenFOAM	Fluid dynamics	Finite volume, turbulence models
ETABS/SAP2000	Structural	FEM, modal/time history
Multisim/LTspice	Electrical	Circuit equations, integration
Adams	Dynamics	Multibody dynamics, ODEs

5

Key Aspects of Numerical Analysis

- **Approximation:** Exact solutions are often impossible to get, so numerical analysis seeks approximate solutions that are **"close enough"** to the real answers.
- **Error Analysis:** Understanding and controlling errors (such as **rounding** and **truncation**) is a crucial part of the field.
- **Algorithm Efficiency:** Numerical methods must be both **accurate** and **efficient**, especially for large-scale problems.
- **Stability and Convergence:** Methods must behave **predictably**. A **stable algorithm** does not magnify small errors, and a **convergent algorithm** approaches the correct solution as computation proceeds.

6

Reasons of Learning Numerical Analysis?

- Engineering Problems Rarely Have **Exact Solutions**
- All Engineering Software Uses **Numerical Methods**
- Numerical Methods Are the **Bridge** Between Theory and Computation
- Engineering Systems Are Often **Too Large** for Symbolic Solutions
- Understanding **Errors** and **Stability** is **Critical**

Numerical Analysis is not just about solving math problems; it's about applying engineering knowledge effectively in the real world.

Numerical analysis is important, especially for accurate, efficient, and reliable engineering design

7

Reference Books

1. Sauer, T. (2012). *Numerical analysis* (2nd ed.). Pearson.
2. Burden, R. L., & Faires, J. D. (2010). *Numerical analysis* (9th ed.). Brooks/Cole.
3. Mathews, J. H., & Fink, K. D. (2008). *Numerical methods using MATLAB* (3rd ed.). Pearson.
4. Scott, L. R. (2011). *Numerical analysis*. Princeton University Press.

8

Assessment

Attendance	5%
Assignments via E-learning	10%
Midterm Exam	25%
Quiz	15%
Final Exam	45%
Total	100%

Week 1: 9/13 February 2026

9

Course content

- Mathematical Preliminaries and Error Analysis
- Solution of Nonlinear Equations
- Systems of Equations
- Interpolation and Polynomial Approximation
- Numerical Differentiation and Integration
- Iterative Techniques and Approximation Theory

Throughout the semester, practice sessions will be held to improve students' skills.

Matlab/ GNU Octave examples will be used during the classes. Students may prefer to use different languages (C, C++, Python etc.) when they solve the given problems

Week 1: 9/13 February 2026

10

Programming Languages

Several programming languages are well-suited to numerical analysis.

MATLAB, Python, C, C++, R and **Julia** are popular choices for their performance and ability to handle complex numerical computations.

GNU Octave is a **free software** for scientific computing and numerical computation. (**https://octave.org/**)

The **Octave syntax** is largely compatible with MATLAB.

We are going to use **MATLAB** codes to teach how the methods can be implemented (**https://matlab.mathworks.com/**)

Students are **free to use any language** to solve the assignments in MT308

11

MATLAB (GNU Octave)

- **MATLAB** is a programming platform designed specifically for engineers and scientists.
- **MATLAB** is short for MATrix LABoratory
- Ideal for its user-friendly interface for **plotting** and **numerical computations**
- **MATLAB** is primarily an **interpreted language**.
- A **script file** (**m-file**), which is a simple text file containing MATLAB commands, can be used to run multiple commands.
- MATLAB **functions** are **reusable blocks** of code that perform specific tasks, accepting inputs and potentially returning outputs.
- MATLAB offers both **built-in functions** and **allows users to define their own**.

12

- In MATLAB, all variables are essentially arrays:
  - Scalars (1x1 arrays)
  - Vectors (1×N or N×1 arrays),
  - Matrices (M×N arrays),
  - Multidimensional arrays
- MATLAB supports other data types, such as cells, structs, tables, and objects.
- MATLAB stores all numeric variables as double-precision floating-point values (i.e. the 'double' data-type in C).
- No declaration is needed for variables. Assigning values creates them.
- MATLAB syntax is influenced by the C language, making it easier for C programmers to learn MATLAB.

13

Built-in Functions:

- MATLAB provides a wide variety of predefined functions for various tasks like mathematical operations (e.g., sin, cos, exp), data analysis, and more.

User-Defined Functions:

- Users can create their own functions for specific logic or algorithms, making the code more modular and readable:

```
function return_value = my_function (input1, input2)
    #Write your code here
    # Write your code here. Use # symbol before the comments
end
```

- To execute a function, you call it by its name, providing the necessary input arguments.

14

Help ?

The help command in both MATLAB and GNU Octave is used to display documentation for functions, toolboxes, or topics.

help function_name

- Examples:
  - help sin
  - help plot

There are various Online Interpreters for both Octave and MATLAB

You may create a student account to use the official Matlab's online interpreter

Some web pages to get help:

- https://docs.octave.org/latest/
- https://www.mathworks.com/help/matlab/index.html
- https://www.mathworks.com/help/index.html

15

Matrix Operations

Operator	Purpose	Description
*	Matrix multiplication	C = A*B is the linear algebraic product of the matrices A and B. The number of columns of A must equal the number of rows of B.
\	Matrix left division	x = A\B is the solution to the equation Ax = B. Matrices A and B must have the same number of rows.
/	Matrix right division	x = B/A is the solution to the equation xA = B. Matrices A and B must have the same number of columns. In terms of the left division operator, B/A = (A'\B')'.
^	Matrix power	A^B is A to the power B, if B is a scalar. For other values of B, the calculation involves eigenvalues and eigenvectors.
'	Complex conjugate transpose	A' is the linear algebraic transpose of A. For complex matrices, this is the complex conjugate transpose.

16

Array Operations

Operator	Purpose	Description
+	Addition	A+B adds A and B.
+	Unary plus	+A returns A.
-	Subtraction	A-B subtracts B from A
-	Unary minus	-A negates the elements of A.
.*	Element-wise multiplication	A.*B is the element-by-element product of A and B.
.^	Element-wise power	A.^B is the matrix with elements A(i,j) to the B(i,j) power.
./	Right array division	A./B is the matrix with elements A(i,j)/B(i,j).
.\	Left array division	A.\B is the matrix with elements B(i,j)/A(i,j).
.'	Array transpose	A.' is the array transpose of A. For complex matrices, this does not involve conjugation.

17

Operator Precedence

Note: Operators at the same precedence level are evaluated from left to right, except for the power operator (^), which evaluates from right to left.

The information on this slide is for reference only. This content will be learned through practical applications during the semester and does not need to be memorized..

Precedence Level	Operators	Description
1 (Highest)	()	Parentheses (grouping)
2	.' '	Transpose (non-conjugate .', complex conjugate ')
3	.^ ^	Power (element-wise .^, matrix ^)
4	+ -	Unary plus and minus
5	.* * ./ / .\ \	Multiplication and division (element-wise and matrix)
6	+ -	Addition and subtraction
7	:	Colon operator
8	< <= > >= == ~=	Relational operators
9	&	Element-wise AND
10	`	`
11	&&	Short-circuit AND
12	`	
13 (Lowest)	=	Assignment

18

Examples (Command Window)

```
>> 8/10
ans =
     0.8000
>> 5*ans
ans =
     4
>> r=8/10
r =
   0.8000
>> r
r =
   0.8000
>> s=20*r
s =
   16
```

```
>> 8 + 3*5
ans =
     23
>> 8 + (3*5)
ans =
     23
>>(8 + 3)*5
ans =
     55
>>4^212 8/4*2
ans =
     0
>>4^212 8/(4*2)
ans =
     3
```

```
>> 3*4^2 + 5
ans =
     53
>>(3*4)^2 + 5
ans =
     149
>>27^(1/3) + 32^(0.2)
ans =
     5
>>27^(1/3) + 32^0.2
ans =
     5
>>27^1/3 + 32^0.2
ans =
     11
```


19

Some useful Matlab Commands

clc	Clears the Command window.
clear	Removes all variables from memory.
clear v1 v2	Removes the variables **v1** and **v2** from memory.
who	Lists the variables currently in memory.
whos	Lists the current variables and sizes,
quit	Stops MATLAB.

The information is for reference only. This content will be learned through practical applications during the semester and does not need to be memorized..

Special Variables and Constants

ans	Temporary variable containing the most recent answer.
eps	Specifies the accuracy of floating-point precision.
i,j	The imaginary unit $\sqrt{-1}$.
Inf	Infinity.
NaN	indicates an undefined numerical result.
pi	The number π.

20