Welcome to **CME 192** MATLAB

Winter 2026

John Winnicki
Institute for Computational and Mathematical Engineering
Stanford University

Outline

About the course

- Schedule
- Descriptions and prerequisites
- Syllabus
- Mini Projects

MATLAB Fundamentals

- MATLAB online
- MATLAB IDE
- MATLAB Syntax, python equivalents
- Core MATLAB functionalities
- MATLAB livescripts
- MATLAB basic plotting capabilities
- General overview of toolboxes

Who uses MATLAB? Why should you be excited about this class?

Course Description

This short course runs for eight weeks/eight lectures and is offered in the Winter quarter during the academic year. It is intended for both students with prior programming experience who are expected to use MATLAB in math, science, or engineering courses, and for ambitious students with no prior programming experience. It consists of interactive lectures and application-based assignments. The goal of the course is to make students fluent in MATLAB and to provide familiarity with its wide array of features, in real-world settings. We will have a special emphasis on learning how to actually apply MATLAB outside of a classroom setting. The course introduces essential MATLAB programming concepts and data structures, and builds toward applied scientific computing workflows involving visualization, numerical linear algebra, simulation, machine learning, and toolbox-driven analysis in realistic problem settings.

About the Course

Schedule

Winter 2023, 4 weeks

- A one-credit course offered by ICME, in collaboration with MathWorks
- Time: 01/05/2026 - 03/13/2026, Thur 4:30 PM - 5:20 PM
- Room: Hewlett Teaching Center 102
- Instructor: John Winnicki (office hours by appointment)
- Materials:
    - All contents and announcements will be posted to course website and canvas.
    - No textbooks.
    - Lecture notes will be provided.
    - Bring in your own laptop (recommended).
  - Course materials credit to: John Winnicki, Xiran Liu (ICME alum), Matthew J. Zahr (ICME alum), Reza Fazel-Rezai (MathWorks), Hung Le (ICME), online resources provided by MathWorks

Description and Prerequisites

- Advanced MATLAB features, syntaxes, and toolboxes
- Various topics from scientific computing, machine learning, parallel computing, image/signal processing
- In-class examples and demos
- Two graded assignments for practice
  - These will use the techniques we cover in class and won't require any domain knowledge
- Selected topics:
  - Advanced plotting and 2D/3D visualizations, Numerical Linear Algebra, ODEs/PDEs, Big data and databases, Python/C++ interfaces and workflows, Statistics, Machine Learning, Optimization, Simulation/Modeling, Image/ Signal Processing, Parallel Processing
  - **Our emphasis will be on using MATLAB toolboxes to solve practical problems in these domains, rather than on the theoretical foundations behind the methods.**
  - Please let me know if there are any topics you would like to be covered!

Syllabus

Lecture	Topic
1	Course Introduction; MATLAB Setup/Capabilities; Why should you care?
2	Advanced plotting and visualizations in MATLAB
3	Numerical Linear Algebra, ODEs/PDEs, and Symbolic Math
4	Big data, python/c++ in MATLAB, Intro to machine learning
5	Statistics and Machine Learning in MATLAB
6	Optimization and Simulation/Modeling
7	Image Processing and Signal Processing
8	Parallel processing with multicore and GPU, Interactive Plotting

Graded Assignments

- **Goal: Obtain hands-on experience of advanced MATLAB features & tools**
- Each assignment will be a set of computational tasks on a real-world dataset leveraging tools in the MATLAB toolboxes covered in class.
- Work individually or in teams.
  - I will be sending out team matching forms for students looking to connect with others. I strongly recommend students to find teams!
- Assignment 1 due date: February 15
- Assignment 2 due date: March 15
- I can be flexible on due dates, but you need to email me with a valid excuse.
- Feel free to ask the instructor for any questions.

Questions?

MATLAB Access

MATLAB Individual Institution License for faculty, staff, and students:
https://uit.stanford.edu/service/softwarelic/matlab
Download to personal machines and activate

MATLAB Online (Recommended for this course)
https://www.mathworks.com/products/matlab-online.html
Sign in using your Stanford email.

MATLAB on Sherlock cluster:
https://www.sherlock.stanford.edu/docs/software/using/matlab

MATLAB Fundamentals

CME 192 Lecture 1

01/01/2025

Stanford University

What is MATLAB?

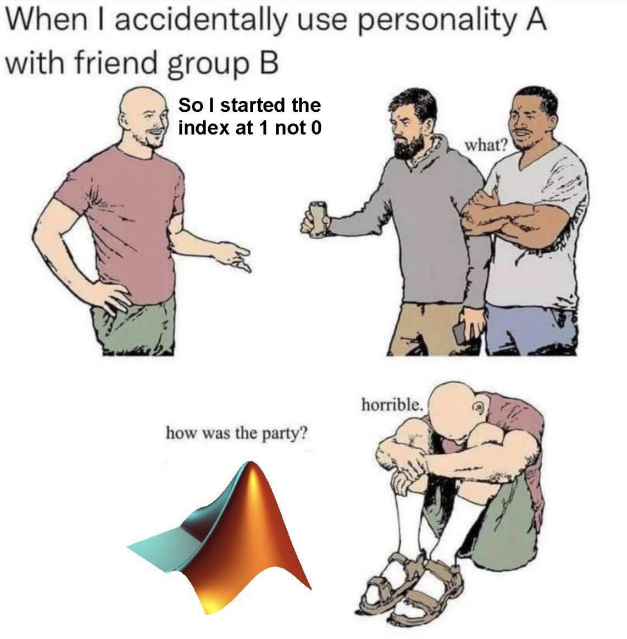
- **MATrix LABoratory (MATLAB)**

> “MATLAB® is a programming platform designed specifically for engineers and scientists to analyze and design systems and products that transform our world. The heart of MATLAB is the MATLAB language, a matrix-based language allowing the most natural expression of computational mathematics.”

- Highly optimized for matrix operations
- Originally written to provide easy access to matrix software: LINPACK (linear system package) and EISPACK (eigen system package)
- Basic element: array
- MATLAB language: highly interactive and interpreted

Comparing to other programming languages

- Development time is usually significantly reduced compared to compiled languages.
- Uses one-based indexing (vs. zero-based indexing)
- Uses columns-major ordering (vs. row-major for C/C++)
- Has a very large standard library of engineering and scientific functions (vs. the external packages in Python/C/C++)
- Emphasizes vectorized operations by default (vs. explicit loops common in other languages).

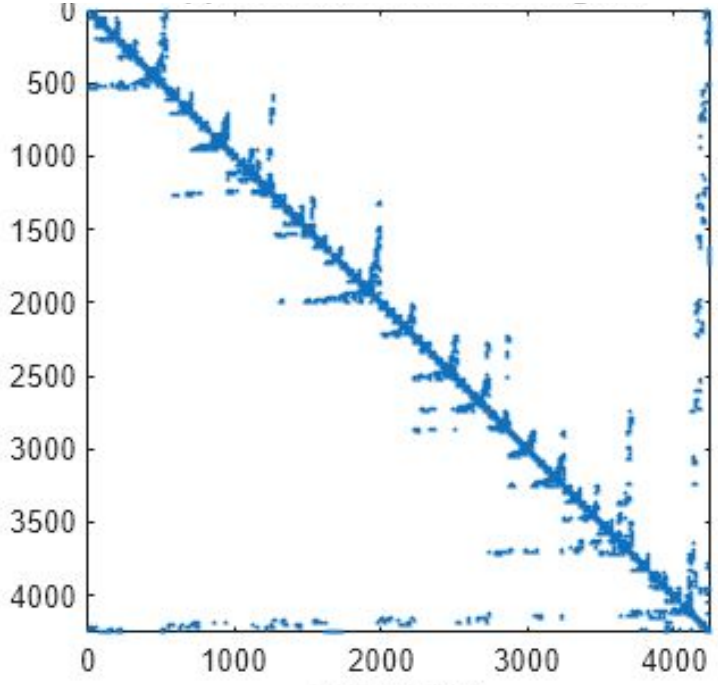


MATLAB Live Scripts

- Classic workflow uses the Editor; interactive documents use the Live Editor.
- Live Scripts let you mix code, output, formatted text, equations, and images in one place.
  - Code can be broken into sections that run independently for faster iteration.
  - Plots and output appear directly next to the code that generated them.
  - The utility of this will become clear once we start using this.
- Text, headings, links, and images can be added like in a regular Word document. Equations can be inserted via point-and-click or written in LaTeX.
- Everything—code, results, and annotations—is saved in one executable document.
- Basically the same thing as Jupyter Notebooks or R Notebooks.
- We will use Live Scripts throughout the course.

Data types and structures

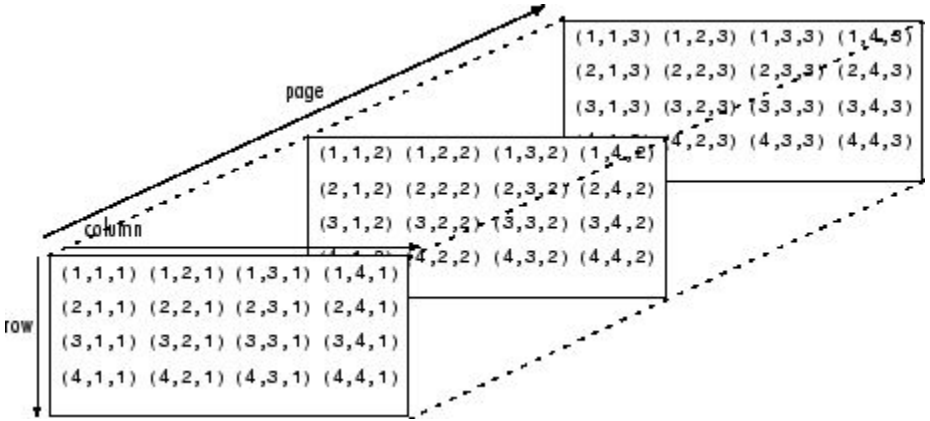
MATLAB Fundamentals



Numerical Arrays

All MATLAB variables are **arrays.**

- Scalars are a single-element array (singleton)
- Vectors are one-dimensional arrays
- Matrices are multidimensional arrays



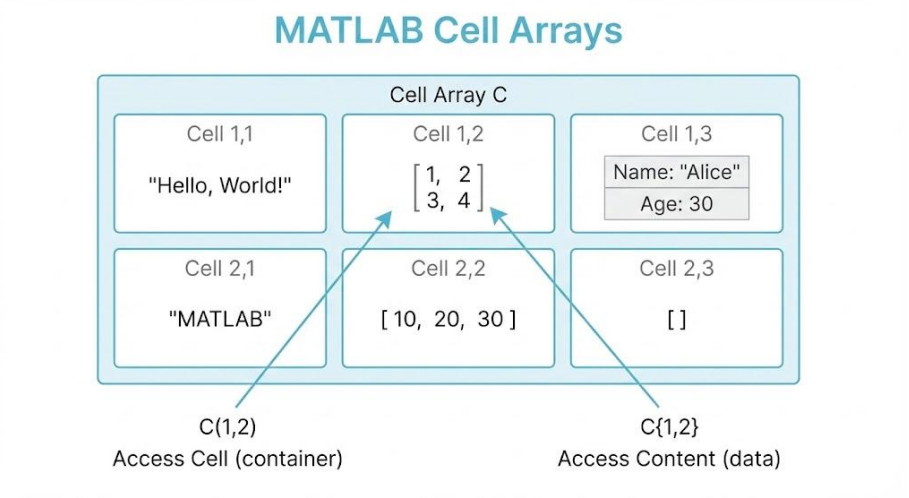
Things to keep in mind

- Arrays must have compatible sizes in any array operation
- Be careful with the array ordering during indexing and reshaping.
- Note the difference between element-wise array operations and matrix operations. (We will see more on this later.)
  - `A.*B` vs. `A*B`

Cell and Cell Arrays

Cell Arrays

- A **cell array** is a flexible container data type whose elements (“cells”) can hold any kind of data.
- Cells often store **lists of text**, **mixed text and numbers**, or **numeric arrays of different sizes/shapes**.
- They offer more generality than numeric arrays, at the cost of some memory and performance



Creating and Accessing Cell Arrays

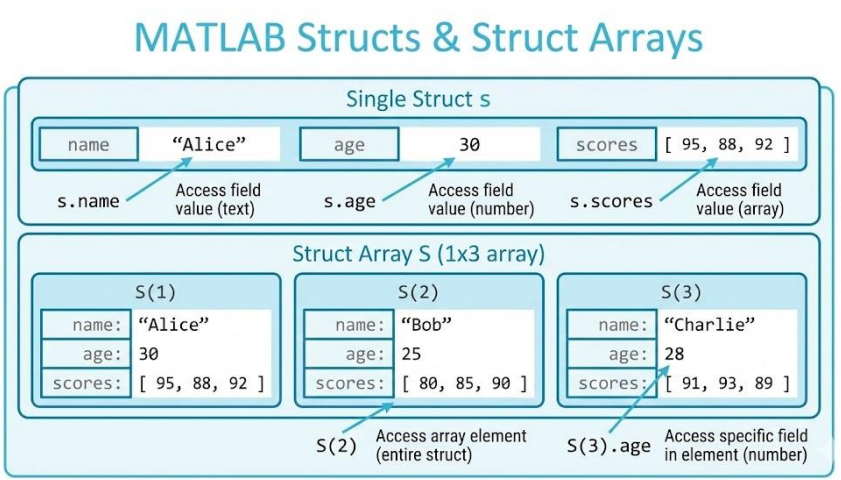
- Construct cell arrays using curly braces {} or the cell function
- Parentheses () index the cell itself: c(i) returns the i-th cell as a 1x1 cell array
- Curly braces {} return the contents of a cell: c{i} returns the contents of the ith cell (the actual data stored inside

Structs and Struct Arrays

- A **structure (struct)** is a flexible data type that groups related information into **named fields**.
- Each array can store **any MATLAB data type**, giving structs versatility similar to cell arrays.
- Unlike cell arrays, structs organize data using a **field-value** structure, making the relationship between pieces of data explicit.

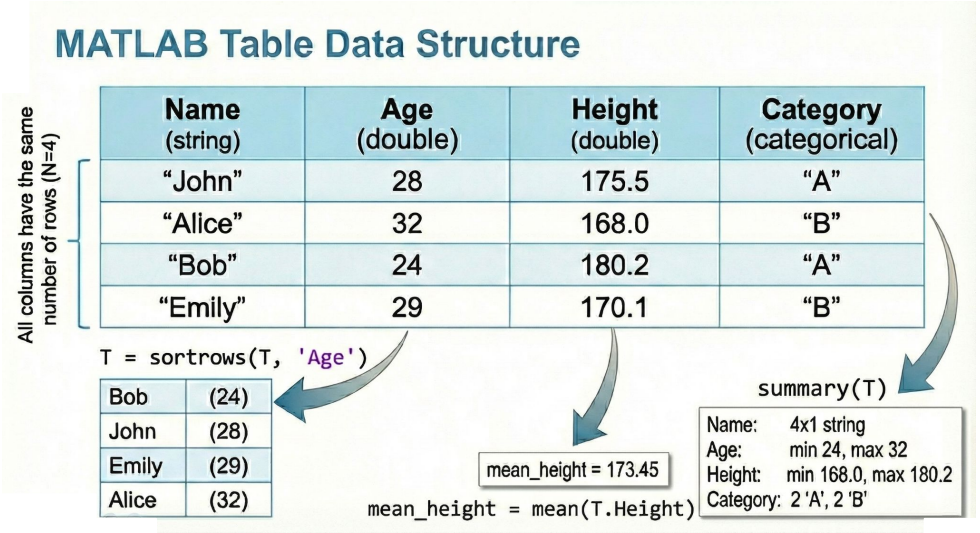
Creating and Accessing Structs

- Construct structs using the struct function or by assigning fields directly:
  - s = struct('name', 'Alice', 'age', 25)
  - s.score = 98
- Access data with dot notation: s.name returns the value stored in the name field.



Tables

- Many datasets are naturally tabular, with observations stored as rows and variables stored as columns.
- MATLAB introduced the **table** data type in **R2013b** to support this pattern cleanly and efficiently
- A table stores **column-oriented variables**, where each variable can have a different data type or size, but **all variables share the same number of rows**.
- Tables are conceptually similar to a **Pandas DataFrame** in Python or a **data frame** in R.
- Ideal for data manipulation, cleaning, and analysis workflows
    - Summarize or visualize data
    - Sort, join, or merge tables
    - Apply functions to variables or groups

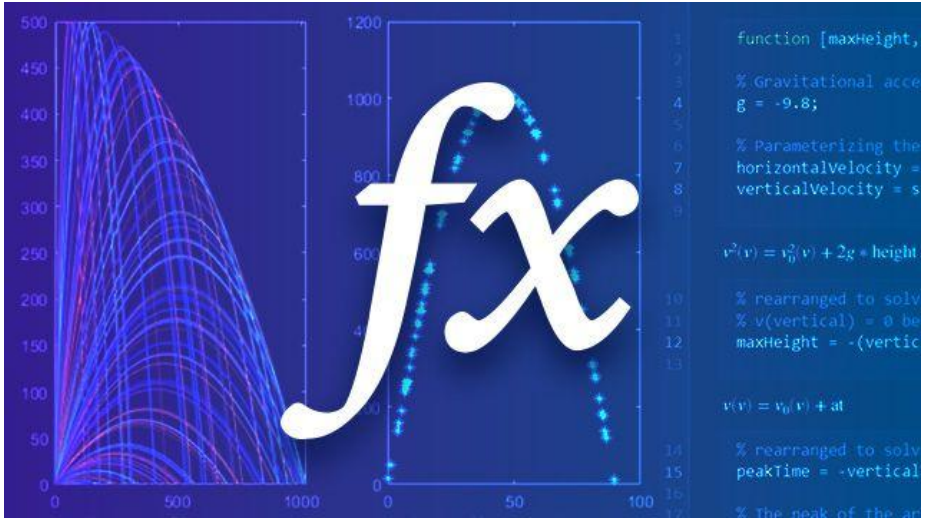


Time Table

- It’s a special kind of table especially useful in scientific computing.
- A specific time is associated with each row.
- Time is usually represented using datetime format (proleptic ISO calendar).
- One can use all of table’s functions on timetable, as well as time-specific functions.

Functions

MATLAB Fundamentals



Scripts vs. Functions

Scripts

- Run a sequence of MATLAB commands.
- Operate in the **base workspace** (no separate scope).
- Are parsed and loaded **every time** they run.

Functions

- Take inputs, perform computations, and return **outputs**.
- Execute in their own **local workspace**, isolated from the base workspace.
- Can share or persist data using **global, persistent**, evalin, or assignin when needed.
- Support multiple forms: **local functions, nested functions, private functions, anonymous functions**, and **class methods**.
- Are parsed and loaded **once**, then reused on subsequent calls.

Anonymous Functions

Functions that are not stored in a program file but associated with a variable.

- Stored directly in function handle
- Store expression and require variables
- Allow zero or more arguments
- Permit nested anonymous functions
- Allow a single executable statement

E.g., `sqr = @(x) x.^2;`

Local Functions

A given MATLAB file can contain multiple functions

The first function is the **main** function.

- callable from anywhere, provided it is in the search path.

Other functions in file are **local** functions.

- Only callable from main function or other local functions in same file
- It enables modularity (large number of small functions) without creating a large number of files.
- It is unfavorable from code reusability standpoint

Nested Functions

A nested function is a function completely contained within some parent function.

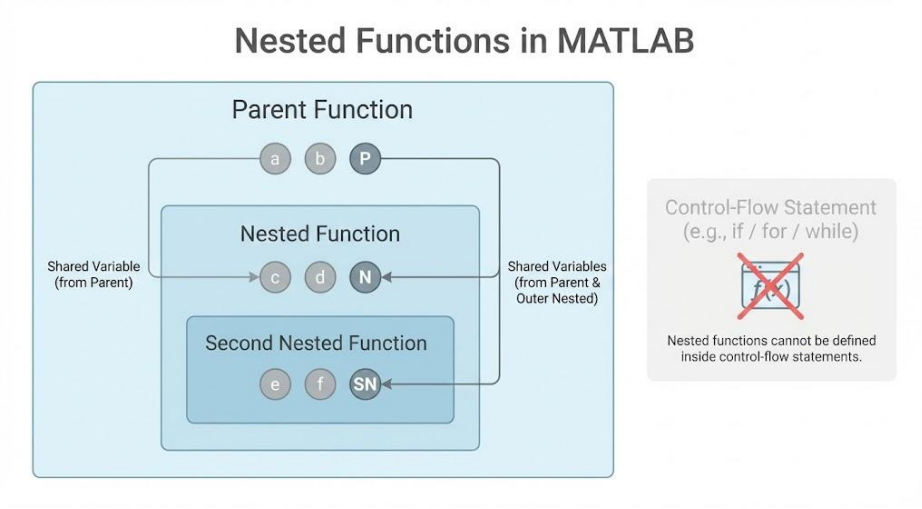
- Useful as an alternative to anonymous function that can't be confined to a single line
- Can't be defined within MATLAB control statements if/elseif/else, switch/case, for, while, or try/catch

Variables are sharable between parent and nested functions.

- If variable in nested function is not used in parent function, it remains local to the nested function.

Multiple levels of nesting is permitted. Nested functions are available from:

- Level immediately above
- Function nested at same level with same parent
- Function at any lower level



Private Functions

Private functions are useful for limiting the scope of a function.

- Designate a function as private by storing it in a subfolder named private
- Only available to functions/scripts in the folder immediately above the private subfolder

Variable Number of Inputs/Outputs

Use nargin to determine how many inputs were actually passed to a function.

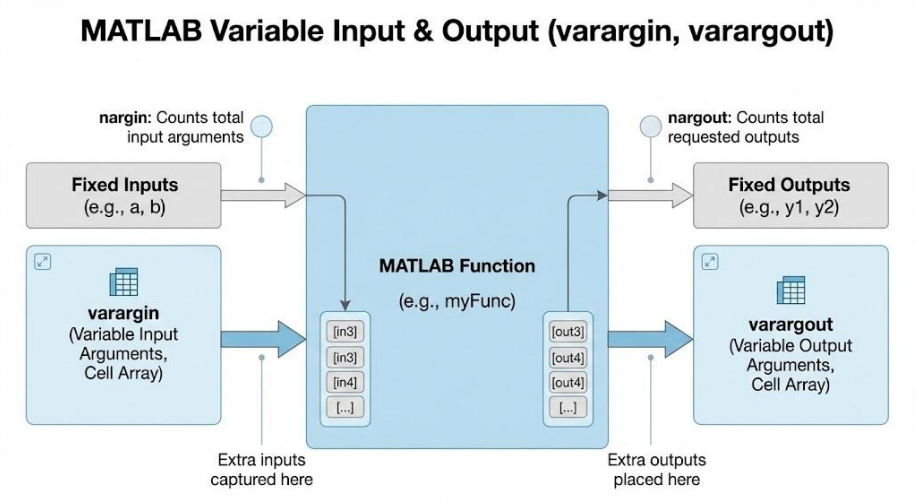
- Do not pass more inputs than the function header allows.

Use nargout to check how many outputs the caller requested.

- Do not request more outputs than the function defines.

MATLAB supports functions with flexible input or output lists:

- Use varargin as the final input argument and varargout as the final output argument when the number of inputs or outputs is variable.
- All arguments before varargin or varargout are matched positionally to the caller.
- Any remaining inputs or outputs are collected into cell arrays named varargin or varargout.



Function Handle (@)

A function handle is a reference to a MATLAB function stored in a variable.

It allows you to call the function indirectly, pass it as an input, or save it for later use.

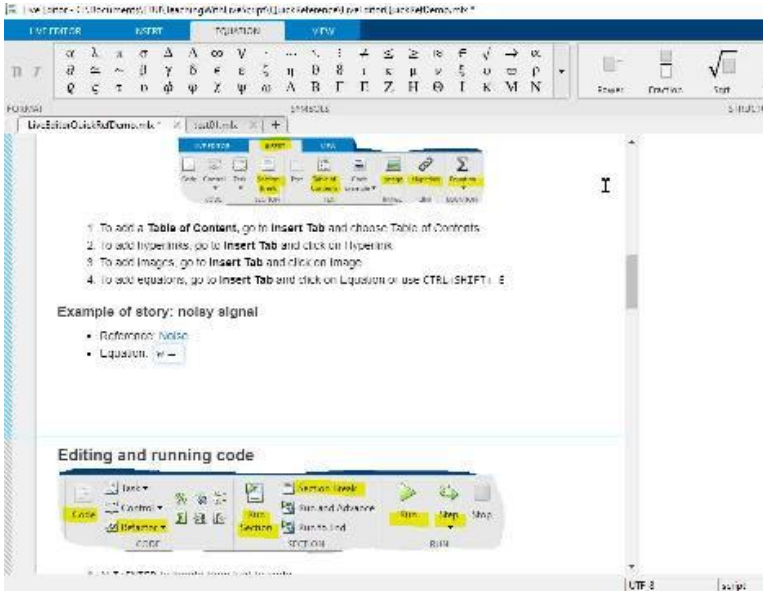
Key uses:

- Call a function outside its usual scope
- Pass functions as arguments to other functions
- Store or capture data for deferred execution
- Enable common numerical workflows such as:
  - Optimization
  - Solving nonlinear equations
  - Solving ODEs Numerical integration

Notes: Function handles must be scalars and cannot be indexed with parentheses. Example application: using a function handle for the trapezoidal rule in numerical integration.

Livescripts

Jump into using MATLAB now!



Livescripts

- MATLAB online
- Creating arrays, functions
- MATLAB basic plotting capabilities
- First look at toolboxes

Who uses MATLAB?

Why should you care?