Welcome to **CME 292**
Advanced MATLAB for Scientific Computing

WINTER 2023

Xiran Liu
Institute for Computational & Mathematical Engineering
Stanford University

Outline

About the course

- Schedule
- Description & prerequisite
- Syllabus
- Mini project

MATLAB Fundamentals

- What is MATLAB
- Data types and structures
- Table & timetable
- Functions

About the Course

Schedule

Winter 2023, 4 weeks

- A one-credit course offered by ICME, in collaboration with MathWorks
- Time: Jan 10 - Feb 2, Tue/Thur 3:00-4:20pm
- Room: Huang Engineering Center 203
- Instructor: Xiran Liu (office hours by appointment)
- Materials:
  - All contents & announcements will be posted to Canvas.
  - No textbooks.
  - Lecture notes will be provided.
  - Bring in your own laptop (recommended).
- Course materials credit to:

  Matthew J. Zahr (ICME alum), Reza Fazel-Rezai (MathWorks), Hung Le (ICME), online resources provided by MathWorks

Description & Prerequisites

- Advanced MATLAB features, syntaxes, and toolboxes
- Various topics from scientific computing
- In-class examples and demos
- A mini course project for practice & optional practice problems
- Selected topics:

  advanced graphics and animation, data management, code optimization, object-oriented programming, optimization, statistical and machine learning, deep learning, etc.
- Prerequisites:
  - Required: CME 192 (Introduction to MATLAB) or equivalent programming background in other languages
  - Recommended: basic knowledge of scientific computing

Syllabus

Lecture	Topic
1	Course Introduction; MATLAB Fundamentals
2	Graphics and Data Visualization
3	File Manipulation; Big Data Handling; Integration with Other Languages
4	Machine Learning with MATLAB
5	Applied Math with MATLAB
6	Object Oriented Programing; Efficient Code Writing
7	Advanced Tools for Images and Signals
8	Project Presentation; Wrap-Up & Additional Topics

Mini Project

- Goal:

  Obtain hands-on experience of advanced MATLAB features & tools
- Work individually.
- Topic choice:
  1. Pick one from suggested topics.

     We provide a list of examples. You may apply similar techniques solve a problem of your interest.
  2. Work on a topic of your own choice.

     You may work on some problem in your research or propose some new problem to work with using features and/or tools at or above the level of those we introduce in class.
  - If you are not sure about the topic, feel free to ask the instructor for suggestions.

Mini Project

- Project statement (due by Lecture 5): 1~2 paragraphs
  - Background of the problem
  - Introduction: data/model/application
- Project report (due at the end of Week 4): 2~4 pages including figures
  - Methods
  - Results: quantitative analysis and/or visualizations
  - Discussion/Conclusion: findings, challenges, takeaways, etc.
- Write-ups can be written in word, latex, or MATLAB live script, and should be handed in as pdf files.
- Presentation (Lecture 8): 5~8 minutes + Q&A
  - a few slides to introduce your problem and the methods, show the results, and discuss the takeaways.

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:
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 292 LECTURE 1

1/10/2023

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 & interpreted

Comparing to other programming languages:

- Its development time is usually significantly reduced compared to compiled languages.
- It uses one-based indexing (vs. zero-based indexing).
- It uses Fortran-like array ordering, i.e., columns-major (vs. row-major for C/C++)

Row-major order

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

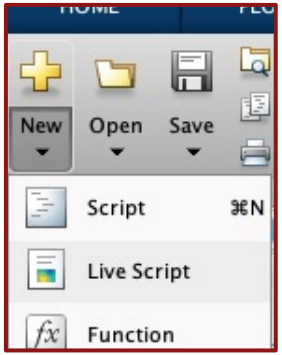
Column-major order

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Image from Wikipedia

Credit to AI

MATLAB Live Scripts



classic coding environment → Editor

interactive document environment → Live Editor

Live Script:

combine code, output, formatted text, equations, images, and interactive controls

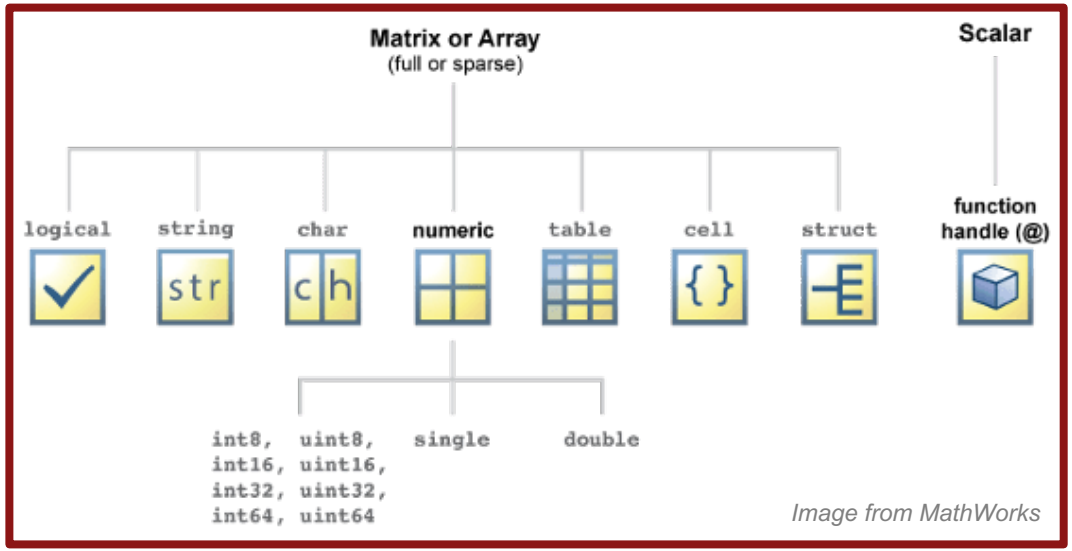
- Divide code into manageable sections that can be run independently.
- View output and visualizations next to the code that produced them.
- Enhance the code and results with formatted text, headings, images, and hyperlinks.
- Insert equations using the interactive editor or create them using LaTeX.
- Save code, results, and formatted text in a single executable document.

Like Jupyter Notebook for Python and R Notebook.

We will use Live Scripts throughout our lectures.

Data types and structures

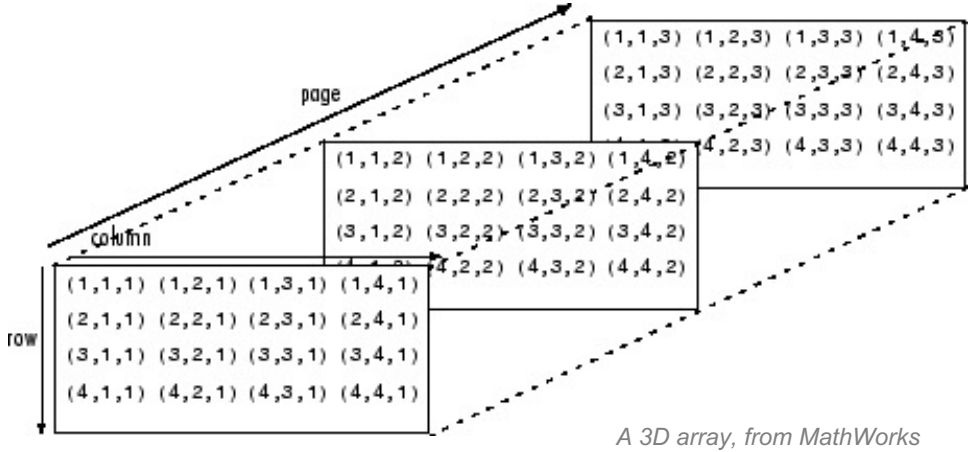
MATLAB Fundamentals



Numerical Arrays

All MATLAB variables are **arrays**.

- Scalars (a single element)
- Vectors (one-dimensional array)
- Matrices (multidimensional array, ≥2 dimensions)



A 3D array, from MathWorks

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`

Quiz

For the same matrix

```
M = reshape(linspace(11,18,8),[2,2,2]),
```

use two different ways to index into the all the odd entries of the matrix.

Quiz

For the same matrix

```
M = reshape(linspace(11,18,8),[2,2,2]),
```

use two different ways to index into the all the odd entries of the matrix.

Solution

```
M = reshape(linspace(11,18,8),[2,2,2]);
M(1:2:end)
M(mod(M,2)==1)
M(1,:,:)
```

Cell & Cell Arrays

A cell array is a data type with indexed data containers called cells.

Each cell can contain *any* type of data.

- Cell arrays commonly contain either lists of text, combinations of text and numbers, or numeric arrays of different sizes.
- It provides additional flexibility and generality over numeric array at the price of storage efficiency.

Construction and data access:

- Cell array can be constructed with `{}` or `cell`.
- Cell **containers** can be indexed using smooth parentheses `():`
  `c(i)` returns i-th cell of cell array `c`
- Cell **contents** can be indexed using curly braces `{}:`
  `c{i}` returns contents of i-th cell of cell array `c`

Struct & Struct Arrays

A structure array is a data type that groups related data using data containers called fields.

Each field can contain any type of data.

- Like cell arrays, struct arrays can hold arbitrary MATLAB data types
- Unlike cell arrays, each entry is associated with a field → Field-Value relationship

Construction and data access:

- Structure array can be constructed with `struct` or `<structName>.<fieldName> = <fieldValue>`
- Data in a field can be accessed using dot notation of the form `structName.fieldName`

Table

Many data structures naturally have a tabular form. Data entries can be organized as rows, each row containing the same number of fields.

MATLAB introduced tables in R2013b.

- Table is a data type suitable for column-oriented or tabular data that is often stored as columns in a text file or in a spreadsheet.
- Tables consist of rows and column-oriented variables.
- Each variable in a table can have a different data type and a different size, but each variable must have the same number of rows.

Similar to Pandas DataFrame in Python and data frame in R.

Good for data manipulation and analysis.

- Summarize information and plot
- Sort, combine, and perform calculations
- Apply functions

Variable Types

Data Type Name	Initial Value in Each Element
double, single	Double- or single-precision 0
doublenan, doubleNaN, singlenan, singleNaN	Double- or single-precision NaN
int8, int16, int32, int64	Signed 8-, 16-, 32-, or 64-bit integer 0
uint8, uint16, uint32, uint64	Unsigned 8-, 16-, 32-, or 64-bit integer 0
logical	0 (false)
categorical	<undefined> categorical value
datetime	NaT datetime value
duration	0 seconds, as a duration value
calendarDuration	0 days, as a calendarDuration value
string	"" (1-by-1 string with no characters)
cellstr	{''} (cell with 0-by-0 character array)
cell	{[]} (cell with 0-by-0 double array)
struct	Scalar structure with no fields
table	Table with no variables
timetable	Timetable with no variables and NaT for row times

Categorical arrays

- Categorical variables are values from a finite set of discrete categories.
- Easy to determine categories and type, modify categories, and count occurrences of categorical array elements by category

Timetable

- 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.

We will look at a hurricane dataset, provided in *hurricaneData.txt* (available on Canvas).

Quiz

Compute the mean windspeed on August 5th 1993 from the data.
**Hint:* use `varfun` or `retime`

Quiz

Compute the mean windspeed on August 5th 1993 from the data.
**Hint:* use `varfun` or `retime`

Solution

```
varfun(@mean, (TT(timerange("1993-08-05","days"),
  'Windspeed')))
retime(TT(timerange("1993-08-05","days"),
  'Windspeed'),'daily','mean')
```

Functions

MATLAB Fundamentals

Add two values together

C = ADDME(A) adds A to itself.

C = ADDME(A,B) adds A and B together.

See also SUM, PLUS.

```
1   function c = addme(a,b)
2
3   switch nargin
4       case 2
5           c = a + b;
6       case 1
7           c = a + a;
8       otherwise
9           c = 0;
10  end
```

Image from MathWorks

Scripts vs. Functions

Scripts

- Execute a series of MATLAB statements
- Use base workspace (does not have own workspace)
- Parsed and loaded into memory every execution

Functions

- Accept inputs, execute a series of MATLAB statements, and return outputs
- Local workspace is defined only during execution of function. `global, persistent` variables and `evalin, assignin` commands help share variables between workspaces or allow them to persist between function executions.
- Local, nested, private, anonymous, class methods
- Parsed and loaded into memory during first execution

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

Query number of inputs passed to a function by `nargin`

- Don't try to pass more than in function declaration

Determine number of outputs requested from function by `nargout`

- Don't request more than in function declaration

Sometimes, the input-output argument list length can be unknown or conditional.

- Use `varargin` as last function input and `varargout` as last function output for input/output argument lists to be of variable length
- All arguments prior to `varargin/varargout` will be matched one-to-one with calling expression
- Remaining input/outputs will be stored in a cell array named `varargin/varargout`

Function Handle (@)

A callable association to MATLAB function stored in variable

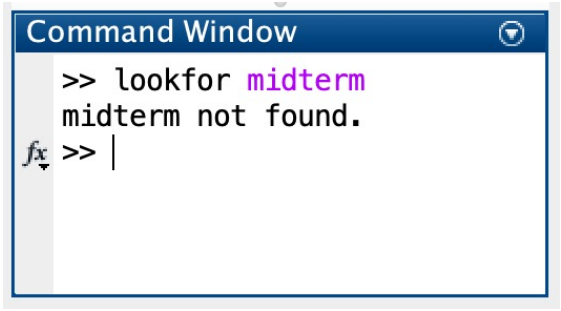
- Enable invocation of function outside its normal scope
- Invoke function indirectly
- Capture data for later use
- Enable passing functions as arguments
  - Optimization
  - Solution of nonlinear systems of equations
  - Solution of ODEs
  - Numerical Integration

Function handles must be scalars, i.e., can't be indexed with ( )

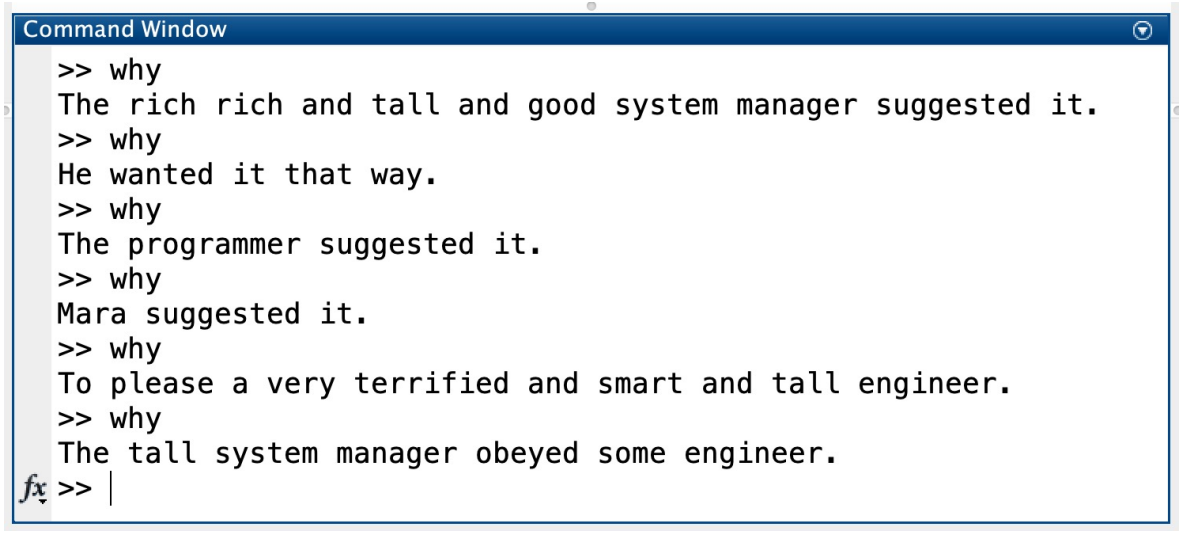
Example: trapezoidal rule for integration

Fun with MATLAB

Type `lookfor midterm` in Command Window.



Type `why` in Command Window. Do it multiple times.



Next Lecture

Graphics and Data Visualization

- Graphic handles
- Advanced plotting
- Plots for publications
- Animations