



EEE247: Programming-2

Lab

Your Interactive Engineering Workbook

Department of Electrical and Electronics Engineering

 Res. Asst. Muhterem Alper Kaplan

Week 1: The Matrix Vault & Spaceship Calibration

 **Mission Briefing:** Calibrate your navigation arrays and bypass the mainframe using matrix mathematics.

🎯 Main Goals of the Week

- Master basic MATLAB operations, vectors, matrices, and indexing.
- Understand the difference between the Command Window and Workspace.
- Learn to use element-wise operators versus standard matrix operators.

Learn MATLAB: Thinking in Grids for Embedded Data

Welcome to your new super-calculator! Unlike standard programming languages, MATLAB (Matrix Laboratory) is designed to handle massive grids of data effortlessly.

- **The Golden Rule of Math:** If you want to multiply two matrices using standard linear algebra (a scalar dot product), use `A * B`. But if you want to multiply corresponding elements together, you **must** use a period before the operator: `.*`, `./`, or `.^`.
- **Semicolons are your friends:** Adding a `;` at the end of a line stops MATLAB from shouting the answer back at you in the Command Window.
- **Rounding Data:** When working with float data, `round(6.66)` goes to the nearest integer (7), `floor(6.66)` forces it down (6), and `ceil(6.66)` forces it up (7).

Activity 1: Human Vectors

Let the variable A be a row matrix (2, 4, 0, -1, 3), and B be a column matrix whose five elements are 2, 5, 8, 3, -5, in that order. Calculate the quantity

$A \times (B + 1)$. Evaluate the following MATLAB expressions by hand and use MATLAB to check the answers: $2/2*3$ and $10/2 \setminus 5 - 3 + 2*4$.

 Write your MATLAB code here:

Activity 2: Spaceship Trajectory

Set up the vector and calculate the magnitude of this vector, (0,1,2,...,50). Create a vector x with the elements... 10, 8, 6, 4, 2, 0, -2, -4. Avoid rounding errors to stay in orbit: test the functions round(6/9+3*2), floor(3.9), and ceil(6/9+3*2).

 Write your MATLAB code here:

Activity 3: The Data Heist

Given the array $A = \begin{bmatrix} 2 & 4 & 1; & 6 & 7 & 2; & 3 & 5 & 9 \end{bmatrix}$, assign the first row of A to a vector called x_1 and assign the last 2 rows of A to an array called y. Let $x = \begin{bmatrix} 3 & 2 & 6 & 8 \end{bmatrix}'$ and $y = \begin{bmatrix} 4 & 1 & 3 & 5 \end{bmatrix}'$. Multiply each element in x by the corresponding element in y, calling the result "z". Add up the elements in z and assign the result to a variable called "w". Compute $x'^*y - w$ and interpret the result.

 Write your MATLAB code here:

☰ Independent Tasks (End of Lab Validation)

Task 1: Create vector $x = 1 : 0.5 : 10$. Compute $y = \sin(x) + \cos(2*x)$ and plot it.

Task 2: For $A = rand(3, 4)$, find the mean and standard deviation of each column.

Task 3: Generate 100 random numbers and count how many are above 0.5.

Summary Key Concepts

- **Element-wise vs. Matrix:** Always check if you need `.*`, `./`, or `.^` when math involves vectors.
- **Indexing:** MATLAB starts counting at 1, not 0.
- **Semicolons:** Use `;` to suppress huge outputs in the Command Window.

Week 2: Alien Biometrics & Zombie Outbreaks

🦋 **Mission Briefing:** Learn to plot dynamic environments, map biometric data, and simulate circuits to survive the outbreak!

🎯 Main Goals of the Week

- Master mathematical functions and element-wise operations.
- Learn how to format, label, and manipulate 2D figure windows.

🏠 Learn MATLAB: Visualizing the Unknown

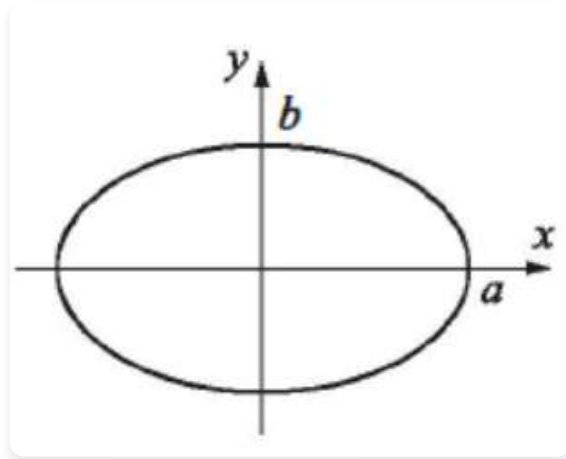
Engineering is useless if you can't communicate your results. The `plot(x, y)` function is your primary tool for visualizing 2D data.

- To add a new plot to an existing figure without erasing the old one, always use the **hold on** command!
- A graph without labels is just a squiggly line. Always add `xlabel('Time (s)')`, `ylabel('Voltage (V)')`, and `title()`.
- Remember: In MATLAB, the natural logarithm (base e) is written as **log(x)**, not `ln(x)`!

Q1: The UFO Landing Pad

The circumference of an ellipse can be approximated by:

$C = \pi[3(a + b) - \sqrt{(3a + b)(a + 3b)}]$. Calculate the circumference of an ellipse with $a = 16$ in. and $b = 11$ in.



 Write your MATLAB code here:

Q2: Decoding Alien Biometrics

Given a vector $t = 1 : 0.2 : 2$, write MATLAB commands to plot the function $f(t) = e^t(1 + \cos(3t))$ on the same figure as $g(t) = \ln(2 + t + t^2)$.

Use different line styles and include axis labels and a legend. Write down the expressions that will also correctly compute $\cos^{-1}(t) + \sin^{-1}(t)$, $\tan^{-1}(t)$, $\cot(t)$, and $\sec(t) + \cot(t) - 1$.

 Write your MATLAB code here:

Q3: The Zombie Outbreak Simulator

Plot the expression (determined in modeling the growth of the US population)

$$P(t) = 197,273,000 / (1 + e^{-0.0513(t-1913.25)}).$$

Use the date, in years AD, using $t = 1790$ to 2000. What population is predicted in the year 2020?

 Write your MATLAB code here:

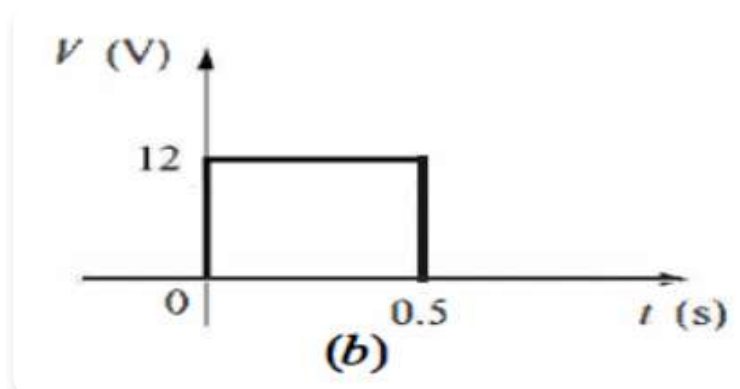
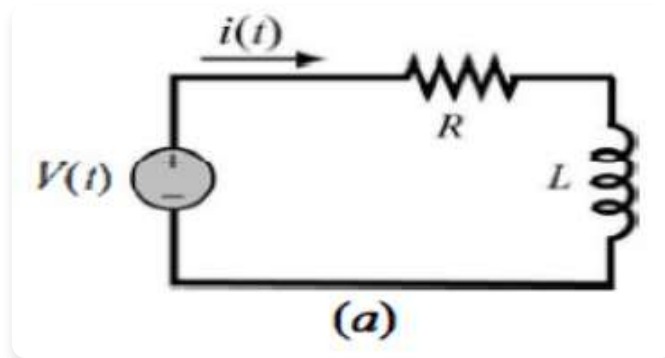
Q4: Defibrillator Power-Up

A resistor, $R = 4\ \Omega$, and an inductor, $L = 1.3\ H$, are connected in a circuit to a voltage source as shown in Figure (a) (an RL circuit). When the voltage source applies a rectangular voltage pulse with an amplitude of $V = 12\ V$ and a duration of $0.5\ s$, as shown in Figure (b), the current $i(t)$ in the circuit as a function of time is given by:

$$i(t) = \frac{V}{R}(1 - e^{(-Rt)/L}) \quad \text{for } 0 \leq t \leq 0.5\ s$$

$$i(t) = e^{-(Rt)/L} \frac{V}{R}(e^{(0.5R)/L} - 1) \quad \text{for } 0.5 \leq t\ s$$

Make a plot of the current as a function of time for $0 \leq t \leq 2\ s$.



 Write your MATLAB code here:

☰ Independent Tasks

Task 1 (Damped Oscillation): Plot $f(t) = te^{-t} \sin(3t)$ for $t = 0 : 0.1 : 10$. Find and mark the maximum value of $f(t)$ on the graph.

Task 2 (Exponential Comparison): Plot $y_1 = e^{-0.5t}$, $y_2 = e^{-0.1t}$, and $y_3 = e^{-t}$ for $t = 0 : 0.5 : 20$. Display them on the same graph with proper legends and line styles.

Task 3 (Subplots): Plot $\sin(t)$, $\sin(2t)$, and $\sin(3t)$ on the same axes for $t = 0 : 0.01 : 2\pi$, then plot them again using subplots.

Task 4 (Parametric Plot): Create a Lissajous curve with $x = \sin(3t)$ and $y = \sin(4t + \pi/3)$ for $t = 0 : 0.01 : 2\pi$.



🔑 Summary Key Concepts

- **Plot Customization:** Use `title`, `xlabel`, `ylabel`, and `grid on` to format engineering data clearly.
- **Visual Combinations:** Master `hold on` for overlaying data and `subplot` for side-by-side comparisons.

Week 3: Logical Indexing & Conditional Logic

▼ **Mission Briefing:** Use boolean masking to filter out sensor noise and build intelligent decision-making scripts.

🎯 Main Goals of the Week

- Master logical indexing and conditional filtering.
- Build robust `while` loops and interactive scripts.

🧠 Learn MATLAB: Scrubbing Real-World Data

Welcome to logical indexing! In other languages, you write long `for` loops to check every number in an array. MATLAB does this via logical masks.

- If you type `mask = (x > 2)`, MATLAB creates a boolean array. You can use this to instantly overwrite bad data: `x(x > 2) = 0`.
- To check if a number is a multiple of 3 (meaning the remainder is 0), use the `mod()` or `rem()` functions.
- Need a random sequence? `randperm(35)` creates a random permutation of integers from 1 to 35.

Q1: Filtering SIM800L Sensor Noise

Given $x = [3, 15, 9, 12, -1, 0, -12, 9, 6, 1]$, write MATLAB commands to perform each step: a) Set values greater than 10 to 0. b) Create a vector containing only positive elements of x . c) Set all elements of x less than its mean to 0.

 Write your MATLAB code here:

Q2: Valve Motor Logic

Create the vector $x = \text{randperm}(35)$ and then evaluate the following function using only logical indexing:

- $y(x) = 2$ if $x < 6$
- $y(x) = x - 4$ if $6 \leq x < 20$
- $y(x) = 36 - x$ if $20 \leq x \leq 35$

 Write your MATLAB code here:

Q3: Loop and User Input

Write a MATLAB script that repeatedly asks the user for a number in Fahrenheit and prints its Celsius equivalent until the user presses Enter without typing a number.

 Write your MATLAB code here:

☰ Independent Tasks

Task 1 (Filtering Data): Create a vector of 20 random integers between -10 and 20. Replace negative numbers with 0 and numbers above 10 with 10. Then compute the average of the modified vector.

Task 2 (Conditional Function): Given $x = -5 : 5$, define $f(x) = x^2$ if $x < 0$, and $f(x) = x$ if $x \geq 0$. Plot $f(x)$ versus x .

Task 3 (Input-Control Loop): Write a script that keeps asking for a number until the user enters a negative value. The script should display the sum of all positive numbers entered.

Task 4 (Even-Odd Counter): Generate a vector of 50 random integers between 1 and 100. Count how many are even and how many are odd. Display the result using `fprintf`.

🔑 Summary Key Concepts

- **Logic Arrays:** Using conditions like $x > 10$ creates a boolean array that acts as a mask to extract or overwrite data.
- **Loops:** Use `for` when you know the exact number of iterations; use `while` when checking for an ongoing condition.

Week 4: User-Defined Functions

 **Mission Briefing:** Move beyond simple scripts. Create dynamic, reusable tools for complex engineering calculations.

🎯 Main Goals of the Week

- Write modular, reusable functions with inputs and outputs.
- Parameterize scripts to accept dynamic engineering inputs.

⚙️ Learn MATLAB: Reusable Control Factories

Stop rewriting the same code! Functions are like little factories. You pass raw materials (inputs) in, and you get a finished product (outputs) out.

- **Syntax:** Which keyword starts a function definition in MATLAB? It's always **function**. (e.g., `function [out1, out2] = mySystem(in1, in2)`)
- **Workspace Isolation:** Variables created inside a function are destroyed when the function ends. This prevents your variables from clashing with the main Command Window.
- **Animation:** You can use a for loop combined with the **pause()** command to slow down execution and dynamically update a plot, simulating a live data feed!

Q1: Amateur Radio Calibrations

Write a function to do the following: a function that returns area of a triangle, a function that returns maximum value in a vector, a function for converting cartesian coordinates to polar coordinates, and a function that outputs first N prime numbers.

 Write your MATLAB code here:

Q2 & Q3: Radar Sweep & Dynamic Signals

Write the function `[x,y] = getCircle(center,r)` to get the x and y coordinates of a circle. The circle should be centered at center (2-element vector containing the x and y values of the center) and have radius r. Then, write a function to dynamically plot a sinusoidal signal. Plot must change dynamically with respect to time. Use sin or cos functions to generate signal and `pause()` to create animation.

 Write your MATLAB code here:

☰ Independent Tasks

Task 1 (Rectangle Properties): Write a MATLAB function `[area, perimeter] = rectProps(length, width)` that accepts the length and width of a rectangle and returns both its area and perimeter.

Task 2 (Vector Statistics): Write function `[meanVal, maxVal, minVal] = vectorStats(v)` that receives a numeric vector `v` and returns its mean, maximum, and minimum values.

Task 3 (Voltage Divider): Write function `Vout = voltageDivider(Vin, R1, R2)` that computes the output voltage V_{out} of a simple voltage divider circuit using the formula $V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$. Test it with $V_{in} = 12V$, $R_1 = 1k\Omega$, $R_2 = 3k\Omega$.

🔑 Summary Key Concepts

- **Function Headers:** The syntax function `[out1] = name(in1)` is critical for passing variables correctly.
- **Scope:** Variables inside a function cannot be seen in the main workspace unless explicitly passed back out.

Weeks 5 & 6: GUIDE & Advanced Simulink

🔧 **Mission Briefing:** Connect logic to visual interfaces and model complex physical systems using block diagrams.

🎯 Main Goals of the Week

- Build interactive graphical user interfaces (GUIs) using GUIDE.
- Simulate real-world RLC and inverter circuits in Simulink.

⚙️ Learn MATLAB: GUIs & System Simulation

Real users don't type code into a command window; they click buttons.

- **GUIDE:** To get the string value from an Edit Text box, you use `get(handles.edit1, 'String')`. To save changes to your interface data globally, use the `guidata(hObject, handles)` command.
- **Simulink:** Why write pages of differential equations when you can draw the circuit? Use the **Scope** block, which behaves like an oscilloscope, to view signals. Use the **Integrator** block to solve equations like $V_c = \frac{1}{C} \int i dt$.
- **Data Export:** To save Simulink data to the workspace as a matrix for easy plotting, change the "Save format" in the "To Workspace" block to **2D-Array (Array)**.

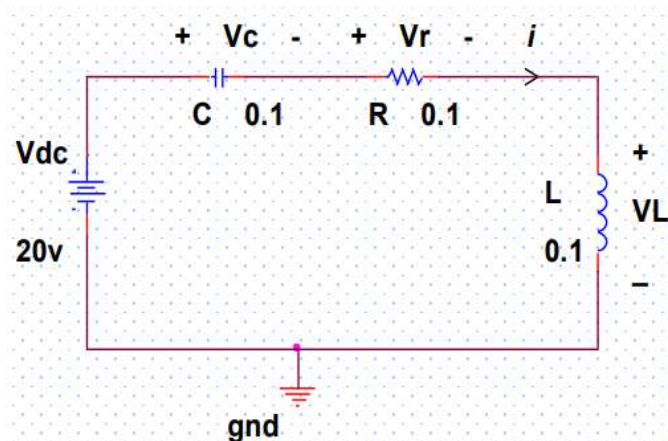
GUI: Remote Calculator (Week 5)

Write a GUI program that reads two numbers from two edit boxes and adds them if the user clicks the add button. Summation of two numbers has to be displayed in a static text box. Write a GUI program that plots a function (hardcoded into script) when the user clicks the plot button.

 Write your callback code here:

Simulink: Audio Amplifier Filter (Week 6)

Create a sine wave with amplitude of 10 and frequency of 60 Hertz using signal generator block. Run the simulation for 5 seconds. Add "To Workspace" block to your design and change save format to 2D-array. Next, create the RLC circuit in Simulink: 120 V 60 Hz, C_1 4.7 μF , L 650 mH, C_2 1.5 μF , and R 470 Ω .



$$V_{dc} - Ri - V_c = L \frac{di}{dt}$$

$$V_L = L \frac{di}{dt}$$

$$V_r = Ri$$

$$V_c = \frac{1}{C} \int i \, dt$$

$$i = C \frac{dV_c}{dt}$$

 Write your workspace I/O code here:

☰ Independent Tasks

Task 1 (GUI Clear): Add a "Clear" push button to your GUI that resets the edit boxes back to empty strings.

Task 2 (Simulink Data): In the data import/export settings, remove the tick next to the data point limit. Run the simulation for 5 seconds.

🔑 Summary Key Concepts

- **Handles Structure:** All data for the GUI is stored in the handles. Use `get` to retrieve data and `set` to change it.
- **I/O Blocks:** Use "To Workspace" to programmatically manipulate simulation results in your MATLAB scripts.

Week 7: File I/O & Data Inspection

 **Mission Briefing:** Engineering runs on data. Learn to read, clean, and export files like a professional data auditor.

🎯 Main Goals of the Week

- Read and write text and Excel files.
- Automate data auditing and handle tables.

Learn MATLAB: Processing the Logs

In the field, you'll often leave a microcontroller running for days to log network strength or valve states to an SD card.

- If you need to read a specific range (e.g., 'A1:B5') from an Excel file, use `readmatrix('file', 'Range', 'A1:B5')`.
- To quickly display the contents of a text file directly in the Command Window, use the `type filename` command.
- If your data has headers (like "Time", "Voltage"), use `readtable('data.xlsx')` so MATLAB organizes it neatly with column labels.

Q1: Workload Audit Data

Create an Excel file named `exampleFile.xlsx`. Read numeric data from the Excel file. Read a specific range of data from the Excel file. Read the second column from the Excel file.

 Write your MATLAB code here:

Q2-Q4: Corrupted Log Files & Tables

Open the badpoem.txt file, read the first line, and then close the file. Open the badpoem.txt file, read all the lines, and close the file. Create a sample text file that contains floating point numbers. Finally, build the table combining a vector 'x' from 0 to 1 in steps of 0.1 with another vector containing exponential values of all elements of x, and create a sample text file that contains these two vectors.

 Write your MATLAB code here:

Independent Tasks

Task 1 (Data Exporter): There is a text file named data.txt containing one numeric value per line. Create a table with the data and its square, then save it as results.xlsx.

Summary Key Concepts

- **Matrix vs Table:** `readmatrix()` drops text and only reads numbers; `readtable()` preserves mixed data structures and headers.
- **File Pointers:** Always close a file ID after use.

Weeks 8 & 9: Advanced Plotting & Symbolic Math

 **Mission Briefing:** Visualize complex data arrays in 3D and solve differential equations algebraically.

🎯 Main Goals of the Week

- Create categorical charts and dynamic 3D visualizations.
- Use `syms` and `dsolve` to map robotics dynamics algebraically.

📖 Learn MATLAB: Algebraic Dynamics

Week 8 takes us into the third dimension. You will use the **`surf()`** function to create dynamic 3D surface plots, mapping physical spaces like heat distribution across a custom PCB layout.

In Week 9, we stop using numbers. By typing `syms x y`, MATLAB treats `x` and `y` as algebraic symbols. **Real-World Application:** When designing Lyapunov-based control strategies, deriving the Jacobian matrices by hand is tedious. The Symbolic Math toolbox handles these derivatives automatically.

Week 8: Department Stats & 3D Models

The number of faculty members in each department at a certain College of Engineering is: ME 22, BM 45, CE 23, EE 33. Experiment with at least 3 different plot types to graphically depict this information. Plot Sinc function, where $\text{Sinc}(x) = \sin(x)/x$, and $-6\pi \leq x \leq 6\pi$. Generate a grid of equally spaced x values over the interval $0 \leq x \leq 5$ and make line plots of the functions x , x^3 , e^x all on the same graph. Use the cylinder function to create x , y , and z matrices and pass them to the surf function to get a surface plot.

 Write your MATLAB code here:

Week 9: System Dynamics ODEs

Create a symbolic function f with variables x and y by using `syms` and assign x^2*y expression to f . Solve the equation $ax^2 + bx + c = 0$ using the `solve` utility in MATLAB. Solve the equation $\frac{d^3y}{dx^3} = ax^2$ by using `dsolve` utility in Matlab. Solve this system of differential equations: $\frac{df}{dt} = 3f + 4g$ and $\frac{dg}{dt} = -4f + 3g$ with $f(0) = 0, g(0) = 1$.

 Write your MATLAB code here:

☰ Independent Tasks

Task 1: Create a 3D surface using `surf(peaks)`.

Task 2: Differentiate the symbolic expression $f = \sin(x)^2 + \cos(y)^2$ with respect to x , then y , and finally take the second derivative with respect to y .



🔑 Summary Key Concepts

- **Categorical Data:** Use `categorical({})` arrays to correctly label axes on bar and pie charts.
- **Symbolic Logic:** `syms` allows MATLAB to keep equations in algebraic form without requiring numerical inputs.

Week 10: Object-Oriented Programming (C++)

🔧 **Mission Briefing:** Transition your logic from MATLAB to C++ to handle memory efficiently on embedded microcontrollers.

🎯 Main Goals of the Week

- Understand fundamental OOP structures, classes, and methods.
- Transition concepts from MATLAB arrays to strict C++ arrays.

🛡️ Learn C++: Encapsulation for Hardware

When writing code for microcontrollers (like an ESP32), memory is extremely limited. We use Object-Oriented Programming (OOP) to protect that memory.

- **Real-World Application:** When flashing an OTA (Over-The-Air) update to an embedded device, you can't process all data packets at once. You must queue them safely in memory buffers like a Stack.
- **Encapsulation:** By declaring an array as `private`, no outside function can accidentally erase or overwrite your memory buffer. They **MUST** use your `public` functions to interact with the data safely!

OTA Firmware Packet Queue

Header of the Stack class is given below. Implement the class Stack and use it in the main program provided.

```
// Stack class declaration
const int SIZE = 5;
class Stack {
private:
    int s_array[SIZE];
    int top; // Top of Stack
    ...
}
```

 Write your C++ code here:

Independent Tasks

Task 1 (Buffer Test): In the `main()` program provided, write a loop that attempts to push 6 items into the Stack to verify that your `isfull()` check properly prevents a crash.

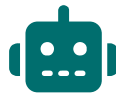
Summary Key Concepts

- **Data Security:** Encapsulation (private vs public) protects memory arrays from being accidentally overwritten by external functions.
- **LIFO Logic:** A Stack operates as Last-In, First-Out, unlike the queue structures you might use in data logging.

Semester Complete!

Congratulations on surviving MATLAB.

You have officially leveled up from simple calculator math
to building full-blown interactive GUIs and embedded
C++ logic.



See you next year in **EEE 352 Control Laboratory!**

*(May your PID controllers always converge and your cart-poles
never tip over.)*

Signing off,

>_ Res. Asst. Muhterem
Alper Kaplan

