Data Representation in MATLAB: From Raw Data to Insight

Background

In the modern computational landscape, scientists and engineers work with diverse data types—from numerical spreadsheets and audio recordings to digital images. The ability to seamlessly import, validate, and analyze this heterogeneous data is a foundational skill. This practical exercise guides you through the complete data handling pipeline in MATLAB, demonstrating how its core data structure—the array—unifies the representation of numbers, sound, and pictures. You will learn to move from raw data to meaningful insight through systematic inspection and visualization.

Core Concept: Whether it's a number in a cell, a sound wave sample, or a pixel's color, in MATLAB, it's all an array.

Part 1: Project Setup & Data Acquisition

Task - Setting Up Your Workspace

All good data analysis begins with organization. Your first task is to create a dedicated folder for this project and set it as MATLAB's Current Folder.

```
% Define the full path to a new folder on your Desktop
currentFolder = fullfile(getenv('USERPROFILE'), 'Desktop', 'CIEPracticals','Prac1_DataLab');

% Create the folder if it doesn't exist
if ~exist(currentFolder, 'dir')
    mkdir(currentFolder);
end
% Change the current directory to your new project folder
cd(currentFolder);
```

Question: Why is it important to set a Current Folder in MATLAB?

Part 2: Working with Spreadsheet Data

Spreadsheets are a common format for storing structured, tabular data. We will import this data and see how it can be represented and analyzed as arrays and tables.

Task 2.1 - Importing Spreadsheet Data

- Locate the provided `ElectricityData.xlsx` file.
- Copy and paste it into your current folder.
- Import the data into the MATLAB workspace using two different functions to see the difference.

```
% Import as a numeric matrix (ignores text headers
electricityMatrix = readmatrix('ElectricityData.xlsx');

% Import as a table (preserves headers and data types)
electricityTable = readable('ElectricityData.xlsx');
```

Task 2.2 - Validation & Inspection

Before analysis, we must check the data's integrity and structure.

```
 % Inspection Commands
 matrixSize = size(electricityMatrix)

 % Get dimensions of the matrix
 tableSize = size(electricityTable)

 % Get dimensions of the table
 tableSummary = summary(electricityTable)

 % Get a detailed summary of the table
 dataTypes = varfun(@class, electricityTable, 'OutputFormat', 'cell') % Check data types per
column

 % Check for missing values
 missingValues = ismissing(electricityTable)
```

Inspection Questions:

1. How many rows and columns of data were imported?
2. What is the fundamental difference between `electricityMatrix` and `electricityTable`?
3. Are there any missing values in the dataset?

Task 2.3 - Visualization

Visualization helps us see patterns and relationships in numerical data.

```
 % Create a bar chart of the total revenue per region
 figure;
 bar(electricityTable.Total);
 set(gca, 'XTickLabel', electricityTable.Region);
 title('Total Revenue by Region');
 xlabel('Region');
 ylabel('Total Revenue ($)');

 % Create a box plot to see the distribution of energy consumption in each sector
 figure;
 boxplot(electricityTable{:, 2:5}, 'Labels', electricityTable.Properties.VariableNames(2:5));
 title('Distribution of Electricity Consumption Across Sectors');
 ylabel('Energy Consumed (kWh)');
```

Interpretation Questions:

1. Which region generated the highest revenue?
2. Looking at the box plot, which sector shows the most consistent energy consumption across regions?

Part 3: Working with Audio Data

Audio is a one-dimensional signal representing how air pressure changes over time. In MATLAB, it becomes a single column vector (mono) or a two-column matrix (stereo).

Task 3.1 - Recording and Importing Audio

- Use your computer's voice recorder app to record a clear 3-5 second clip of your voice (e.g., saying "MATLAB is powerful!").
- Save the file as `myVoice.wav` in your project folder.
- Import the audio into MATLAB.

```
% Import the audio file
% y: the sampled audio data (the amplitude values)
% Fs: the sample rate (samples per second)
[myVoice, Fs] = audioread('myVoice.wav');
```

Task 3.2 - Validation & Inspection

Let's understand the properties of our audio signal.

```
% Inspection Commands
audioLength = length(myVoice)    % Total number of samples
audioDuration = audioLength / Fs % Duration of the audio in seconds
audioSampleRate = Fs             % Sampling frequency (Hz)
audioDataType = class(myVoice)   % Data type of the amplitude values

% Check for clipping (distortion caused by amplitude hitting max)
maxAmplitude = max(abs(myVoice))
```

Inspection Questions:

1. How long is your audio clip in seconds?
2. What is the sample rate? What is the practical implication of this number?
3. Is there any evidence of clipping in your recording (i.e., are any absolute amplitude values very close to 1.0)?

Task 3.3 - Visualization

Visualizing audio helps us "see" the sound.

```
% Create a time vector for the x-axis
timeVector = (0:length(myVoice)-1) / Fs;

% Plot the audio waveform
figure;
subplot(2,1,1);
plot(timeVector, myVoice);
title('Waveform of My Voice Recording');
xlabel('Time (seconds)');
ylabel('Amplitude');
grid on;

% Plot a histogram of the amplitude values
subplot(2,1,2);
histogram(myVoice, 50);
title('Distribution of Audio Amplitudes');
xlabel('Amplitude');
ylabel('Frequency');
```

Interpretation Questions:

- Does your waveform show a repeating pattern corresponding to the syllables of your speech?
- Is the amplitude histogram centered around zero and roughly symmetrical?

Part 4: Working with Image Data

A digital image is essentially a grid of colored points called pixels. A grayscale image is a 2D array where each value is a brightness. A color image is a 3D array (Height x Width x 3), where the third dimension holds the Red, Green, and Blue color channels.

Task 4.1 - Capturing and Importing an Image

- Use your phone or webcam to take a clear, well-lit picture of a single object.
- Transfer and save the file as `myImage.jpg` in your project folder.
- Import the image into MATLAB.

```
% Import the image file
myImage = imread('myImage.jpg');
```

Task 4.2 - Validation & Inspection

Let's dissect the structure of our image data.

```
% Inspection Commands
imageDimensions = size(myImage)  % Get the dimensions [rows, cols, channels]
imageDataType = class(myImage)   % Get the data type (e.g., uint8)

% Get basic statistics about pixel intensities
minPixel = min(myImage(:));
maxPixel = max(myImage(:));
meanPixel = mean(double(myImage(:))); % Convert to double for calculation
```

Inspection Questions:

1. Is your image color (3 dimensions) or grayscale (2 dimensions)?
2. What is the data type and range of the pixel values? (e.g., `uint8` means integers from 0 to 255).
3. What is the minimum, maximum, and average pixel intensity values in your image?

Task 4.3 - Visualization

Visualizing the image itself is primary but visualizing its pixel distribution reveals contrast and composition.

```
% Display the image
figure;
subplot(1,2,1);
imshow(myImage);
title('My Imported Image');

% Display the histogram of pixel intensities
subplot(1,2,2);
if ndims(myImage) == 3
    % For color images, let's look at the grayscale histogram
    imhist(rgb2gray(myImage));
else
    % For grayscale images
    imhist(myImage);
end

title('Pixel Intensity Histogram');
xlabel('Intensity Value');
ylabel('Number of Pixels');
```

Interpretation Questions:

1. Does the histogram of your image show a good range of contrasts (a spread-out distribution)?
2. Where are the pixel values concentrated? In the darks, mid-tones, or highlights? What does this say about your image's exposure?

Part 5: Final Synthesis & Report

Task 5.1 - Unified Workspace Review

In your MATLAB Workspace, you should now have variables for all three data types: `electricityTable` (spreadsheet), `myVoice` and `Fs` (audio), and `myImage` (image).

Final Reflection:

Despite their different origins and natures, all these data are now represented as arrays in your workspace. Briefly describe the "shape" and meaning of the data in each of these core variables.

Task 5.2 - Submission

Create a single MATLAB Live Script that executes all the tasks from this document. Your script should be well-commented and include:

1. Code: All the commands for setup, import, inspection, and visualization.
2. Answers: Correct answers to inspection/interpretation questions are provided as text in the Live Script
3. Outputs: The generated figures and answers to the inspection/interpretation questions embedded as text.
4. Summary: A final section (approx. 200 words) titled "Data Unification in MATLAB," discussing how the array structure serves as a universal data representation and reflecting on the insights gained from the validation and visualization steps for each data type.

The primary assessment is the MATLAB Live Script, which is evaluated based on the following criteria:

Criteria	Weight	Description
Technical Execution & Code	30%	All code runs without error. Proper use of import functions (`readtable`, `audioread`, `imread`) and inspection commands (`size`, `summary`, etc.). Code is well-commented.
Visualization & Output	40%	All required figures are generated and properly labeled. Correct answers to inspection/interpretation questions are provided as text in the Live Script.
Synthesis & Reflection	30%	The final summary demonstrates a clear understanding of how arrays unify different data types and reflects on insights gained from the validation and visualization steps.