Table of Contents

Vector making:

```
rowV = [1,2,3,4,5];
colV = [1;2;3;4;5];
matA = [1,2,3;4,5,6;7,8,9];
disp(rowV);
disp(colV);
disp(matA);

matB = ones(3,3);
disp(matB);

resultMat = matA + matB;
disp(resultMat);
```

```
     1     2     3     4     5

     1
     2
     3
     4
     5

     1     2     3
     4     5     6
     7     8     9

     1     1     1
     1     1     1
     1     1     1

     2     3     4
     5     6     7
     8     9    10
```

Defitions and uses of Arrays vs. Matrices:

Arrays can be shown in 3D structures and used in programming world. Matrices show 2D structures and used in mathematics world.

```
% Define a cell array with different data types (use curly braces "{}" for
the array designation)
C = {123, 'Hello, world!', [1 2 3; 4,5,6], true};
disp(C)


% Accessing data within the cell array (ex. access the second element in the
array)
str = C{2};
disp(['String value = ', str]);

% Acess a specific number within a matrix stored in the third cell of the
array
matarr = C{3}(2,3); %% Access the element in the 2nd row and
disp(['Matrix data: ', num2str(matarr)]);
```

```
    {[123]}    {'Hello, world!'}    {2x3 double}    {[1]}

String value = Hello, world!
Matrix data: 6
```

Table creation:

Define the data –

```
names = ["Alice", "Bob", "Charlie"]; % A string array
ages = [25, 28, 22]; % A column vector of numeric values
cities = ["New York", "Los Angeles", "Chicago"]; % A string array

% Create the table
peopleTable = table(names, ages, cities, 'VariableNames', {'Name', 'Age',
'City'});

%Display the table –
disp(peopleTable);

% Access all ages –
allAges = peopleTable.Age;

% Access the city of the second person –
secondCity = peopleTable.City(2);

% Display the accessed data –
disp(["All Ages:-", allAges]);
disp(["City of the second person:", secondCity])
```

```
                Name                        Age
City
    ____________________________    _____________
____________________________________

    "Alice"    "Bob"    "Charlie"    25    28    22    "New York"    "Los
Angeles"    "Chicago"
```

```
    "All Ages:-"    "25"    "28"    "22"

    "City of the second person:"    "Los Angeles"
```

Import and Export of data:

Create a matlab table Create a MATLAB table

```
T = table(["Alice"; "Bob"; "Charlie"], [30; 25; 22], ["New York"; "Chicago";
"Charleston"], ...
    'VariableNames', {'Name', 'Age', 'City'});


% Display the table to be exported:
disp(T)

% Export the table to a CSV file:
writetable(T, "Output.csv");

% The file 'output.csv' will be saved in the current workign directory
```

```
      Name       Age        City
    _________    ___    _____________
```

```
    "Alice"      30     "New York"
    "Bob"        25     "Chicago"
    "Charlie"    22     "Charleston"
```

Numerical operations in MATLAB:

Define the coefficients:

```
a = 2;
b = -4;
c = -6;

% Calculate the discriminant:
D = b^2 - 4*a*c;

% Calculate the roots:
x1 = (-b + sqrt(D))/(2*a);
x2 = (-b - sqrt(D))/(2*a);

% Display the roots:
disp(["Root_1 = ", (x1)]);
disp(["Root_2 = ", (x2)]);
```

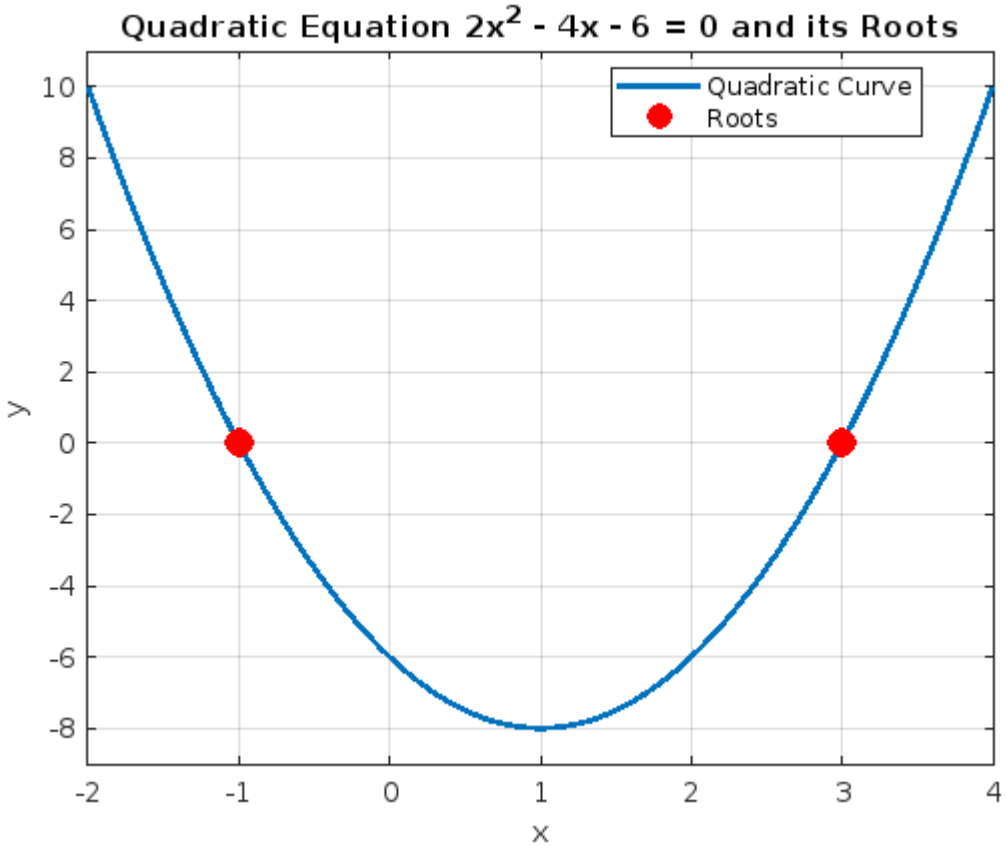
```
    "Root_1 = "    "3"

    "Root_2 = "    "-1"
```

Visualisation of the quadratic equation:

Coefficients of the quadratic eqn:

```
a = 2;
b = -4;
c = -6;

% Roots of the eqn:
D = b^2 - 4*a*c;
x1 = (-b + sqrt(D))/(2*a);
x2 = (-b - sqrt(D))/(2*a);

% Generate x values for plotting:
x = linspace(min(x1,x2)-1, max(x1,x2)+1, 400); % Range

% Calculate corresponding y values for the quadratic eqn:
y = a*x.^2 + b*x + c;

% Plot the quadratic eqn:
figure; % Create a new figure window
plot(x,y, 'LineWidth', 2);
hold on; % Keep the plot for adding more elements
```

```
% Highlight the roots on the plot:
plot(x1, 0, 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r'); % Root 1
plot(x2, 0, 'ro', 'MarkerSize', 10, 'MarkerFaceColor', 'r'); % Root 2

% Enhance the plot:
title('Quadratic Equation 2x^2 - 4x - 6 = 0 and its Roots');
xlabel('x');
ylabel('y');
grid on; % Add grid lines for better readability
legend('Quadratic Curve', 'Roots', 'Location','best');

% Adjust the axis for better visualisation:
axis([min(x) max(x) min(y) - 1 max(y) + 1]);
hold off; % Release the plot
```



Published with MATLAB® R2024b