Matrices and Vectors - Summary

1. Vectors

A vector is a one-dimensional array of numbers.

- Row vector: [1 2 3]

- Column vector: [1; 2; 3]

Vectors represent quantities with direction and magnitude.

2. Matrices

A matrix is a two-dimensional array of numbers.

Example:

[1 2 3;

 4 5 6]

This is a 2x3 matrix (2 rows, 3 columns).

3. Common Operations

- Addition/Subtraction: [1 2] + [3 4] = [4 6]

- Scalar Multiplication: 2 * [1 2] = [2 4]

- Matrix Multiplication: A(2x3) * B(3x1) = C(2x1)

- Transpose: [1 2 3]' = [1; 2; 3]

- Element-wise: [1 2] .* [3 4] = [3 8]

- Dot Product: [1 2] . [3 4] = 1*3 + 2*4 = 11

- Norm: ||v|| = sqrt(v1^2 + v2^2 + ... + vn^2)

4. Accessing Elements, Rows, Columns

- A(i,j): element at row i, column j

- A(i,:): entire row i

- A(:,j): entire column j

Example: A = [1 2 3; 4 5 6; 7 8 9]

A(2,3) = 6, A(1,:) = [1 2 3], A(:,2) = [2; 5; 8]

5. Operations on Rows/Columns

Matrices and Vectors - Summary

- Multiply column 2 by 2: A(:,2) = A(:,2)*2
- Sum: sum(A,1) => sum by column, sum(A,2) => by row
- Replace: A(2,:) = [0 0 0]; A(:,3) = [1; 1; 1]

6. MATLAB Code Examples

```
A = [1 2 3; 4 5 6; 7 8 9];
val = A(2,3);          % 6
row1 = A(1,:);          % [1 2 3]
col3 = A(:,3);         % [3; 6; 9]
A(:,2) = A(:,2) * 2;
A(1,:) = [10 11 12];
sum_row = sum(A, 2);   % Sum of rows
mean_col = mean(A, 1); % Mean of columns
```

2D and 3D Plotting in MATLAB

1. 2D Plotting in MATLAB

2D plotting represents the relationship between two variables (x and y).

Basic Command:

plot(x, y)

Example:

x = 0:0.1:2*pi;

y = sin(x);

plot(x, y);

title('Sine Wave');

xlabel('x'); ylabel('sin(x)');

grid on;

Line Styles and Colors

plot(x, y, 'r--o')

'r' = red, '--' = dashed, 'o' = circle markers

Plotting Multiple Functions

plot(x, sin(x), 'b', x, cos(x), 'g');

legend('sin(x)', 'cos(x)');

Stem Plot (Discrete Data)

x = 1:5;

y = [3 5 2 6 4];

stem(x, y);

title('Stem Plot');

Bar Chart

bar([3 6 4 2]);

2D and 3D Plotting in MATLAB

Multiple Subplots in One Figure

```
subplot(2,1,1);
plot(x, sin(x));
title('Sine');

subplot(2,1,2);
plot(x, cos(x));
title('Cosine');
```

Filled Area: fill()

```
x = [1 2 3 4];
y = [2 3 1 5];
fill(x, y, 'g');
```

2. 3D Plotting Basics

plot3(x, y, z) plots lines in 3D space.

Example:

```
t = 0:0.1:10;
x = sin(t); y = cos(t); z = t;
plot3(x, y, z);
xlabel('X'); ylabel('Y'); zlabel('Z');
grid on;
```

Surface and Mesh Plot

```
[X, Y] = meshgrid(-5:0.5:5);
Z = sin(sqrt(X.^2 + Y.^2));
mesh(X, Y, Z);
surf(X, Y, Z);
```

2D and 3D Plotting in MATLAB

Contour Lines (Level Curves)

contour(X, Y, Z);

General Tips

hold on - plot multiple items

axis equal - equal axis scale

grid on - show grid

legend - describe plots

Solving Equations in MATLAB

1. Solving Linear Equations (Ax = b)

Example:

```
A = [2 3; 4 1];
b = [8; 10];
x = A\b;
% Output: x = [1; 2]
```

2. Solving a Single Symbolic Equation

```
syms x
eq = x^2 - 5*x + 6 == 0;
sol = solve(eq, x);
% sol = 2, 3
```

3. Solving Multiple Symbolic Equations

```
syms x y
eq1 = x + y == 5;
eq2 = x - y == 1;
sol = solve([eq1, eq2], [x, y]);
sol.x, sol.y
```

4. Solving Nonlinear Equations Numerically (fzero)

```
f = @(x) x^3 - x - 1;
root = fzero(f, 1);
% root is approximately 1.3247
```

5. Solving Systems Numerically (fsolve)

```
f = @(v)[v(1)^2 + v(2)^2 - 4; v(1)*v(2) - 1];
x0 = [1, 1];
sol = fsolve(f, x0);
```

Solving Equations in MATLAB

6. Solving Polynomial Equations with roots()

coeff = [1 2 -3 1];

r = roots(coeff);

% r = [1; -3; -1] (example)

Tips

- Use double() to convert symbolic to numeric.

- Use pretty() or disp() for display.

- Use assume(x, 'real') to simplify.

Relational and Logical Operators in MATLAB

---

1. Relational Operators

Relational operators are used to compare values or arrays. They return logical results (1 for true, 0 for false).

Operator	Description	Example	Result
`==`	Equal to	`5 == 5`	`1 (true)`
`~=`	Not equal to	`3 ~= 2`	`1 (true)`
`>`	Greater than	`7 > 2`	`1 (true)`
`<`	Less than	`3 < 4`	`1 (true)`
`>=`	Greater than or equal to	`6 >= 6`	`1 (true)`
`<=`	Less than or equal to	`2 <= 3`	`1 (true)`

Example with arrays:

```
A = [1 2 3];
B = [3 2 1];
R = A > B
% Output: [0 0 1]
```

---

2. Logical Operators

Logical operators are used with logical values (true/false or 1/0).

Operator	Description	Example	Result
`&`	Logical AND	`true & false`	`false`
`	`	Logical OR	`true
`~`	Logical NOT	`~true`	`false`
`xor`	Exclusive OR	`xor(1, 0)`	`1 (true)`

Example with arrays:

```
A = [true false true];
B = [false false true];
R = A & B
% Output: [0 0 1]
```

---

3. Combining Relational and Logical Operations

You can combine both relational and logical operators for condition checks:

```
x = 10;
y = 5;
(x > 5) & (y < 10)
% Output: 1 (true)
```

---

4. Short-Circuit Logical Operators

These are used only with **scalar** (single) values:

Operator	Description
`&&`	Short-circuit AND
`	

Example:

```
a = 4;
b = 2;
(a > 1) && (b < 5)
% Output: true
```

Control Flow in MATLAB: if, for, while

---

1. if Statement

The `if` statement is used to execute code based on a logical condition.

Syntax:

```
if condition
    statements
elseif another_condition
    statements
else
    statements
end
```

Example:

```
x = 10;
if x > 0
    disp('Positive number');
elseif x == 0
    disp('Zero');
else
    disp('Negative number');
end
```

---

2. for Loop

The `for` loop is used to repeat a block of code a specific number of times.

Syntax:

```
for index = start:step:end_value
    statements
end
```

Example:

```
for i = 1:2:5
    disp(['Value is: ', num2str(i)]);
end
% Output:
% Value is: 1
% Value is: 3
% Value is: 5
```

---

3. while Loop

The `while` loop repeats a block of code **as long as a condition remains true**.

Syntax:

```
while condition
    statements
end
```

Example:

```
x = 1;
while x <= 5
    disp(['x = ', num2str(x)]);
    x = x + 1;
end
```

---

4. Comparison Summary

Feature	`if`	`for`	`while`
Purpose	Conditional execution	Loop with known iterations	Loop while condition is true
Repeats	Once if true, else skipped	As many times as specified	Until the condition becomes false
Use Case	Decisions	Iterating over fixed range	Unknown number of repetitions

Mathematical Concepts: Laplace, Limit, Integral, Derivative

---

1. Laplace Transform

The Laplace transform is a mathematical tool used to transform time-domain functions into the frequency domain.
It simplifies the solution of differential equations, especially in control systems and circuit analysis.

Example:

```
syms t s
f = exp(-2*t);  % Function f(t)
F = laplace(f, t, s);  % Laplace Transform
% Result: F(s) = 1 / (s + 2)
```

---

2. Limit

The limit is a concept used to describe the behavior of a function as its variable approaches a certain value.
It is essential in calculus, especially for defining derivatives and integrals.

Example:

```
syms x
f = (sin(x) / x);  % Function f(x)
limit(f, x, 0)  % Limit as x approaches 0
% Result: 1
```

---

3. Integral

Integration is the reverse process of differentiation and is used to calculate the area under curves, among other things.
It is a fundamental concept in calculus.

Example:

```
syms x
f = x^2;  % Function f(x)
integral_value = int(f, x);  % Compute the integral
% Result: (x^3) / 3 + C
```

---

4. Derivative

The derivative represents the rate of change of a function.
It measures how a function changes as its input changes. Derivatives are key to understanding the behavior of functions in physics, engineering, and other fields.

Example:

```
syms x
f = x^2 + 3*x;  % Function f(x)
derivative_value = diff(f, x);  % Compute the derivative
% Result: 2x + 3
```