Matrix and Array Operations: Arithmetic operations on matrices and vectors come in two forms: "matrix sense" based on the normal rules of linear algebra and "array sense" carried out element by element. For scalars both types of operations produce the same results.

"Matrix sense" operations are: + addition, – subtraction, * multiplication, / division, and ∧ exponentiation.

"Array sense" operations are .* multiplication, ./ division, and .∧ exponentiation.

Relational Operators: MATLAB’s relational operators are:

- == equal
- ~= not equal
- < less than
- > greater than
- <= less than or equal
- >= greater than or equal

Logical Operators: Some logical operators are: "&" the logical "and", "|" the logical "or", and "~" the logical "not".

Flow Control: MATLAB has four flow structures: the if statement, the for loop, the while loop, and the switch statement.

If Statement: The simplest form of the if statement is:

if expression
    statements
end

where the statements are executed if the expression is true.
Example:

```matlab
if x > y
    temp = y;
    y = x;
    x = temp;
end
```

Statements to be executed only if expression is false can be placed after else:

```matlab
if rem(n,2) == 1  % Remainder modulo 2
    n = 3*n + 1;
else
    n = n/2;
end
```

For Loop: The for loop has the form:

```
for variable = expression
    statements
end
```

Usually expression is a vector of the form i:s:j. The statements are executed with variable equal to each element of expression in turn. For example:

```matlab
s = 0
for i = 1:25
    s = s + 1/i;
end
s
```

computes the first 25 terms of the harmonic series.

Multiple for loops can be nested:

```matlab
n = 5;
A = eye(n);
for j = 2:n
    for i = 1:j-1
        A(i,j) = i/j;
        A(j,i) = i/j;
    end
end
```
While Loop: The **while** loop has the form:

```
while expression
    statements
end
```

The *statements* are executed as long as the *expression* is true. The following example approximates the smallest nonzero floating point number.

```
x = 1;
while x > 0
    xmin = x;
    x = x/2;
end
xmin
```

Switch statement: The **switch** statement consists of "**switch** expression" followed by a list of "**case** expression statements", optionally ending with "**otherwise** statements" and followed by **end**. The **switch** expression is evaluated and the statements following the first matching **case** expression are executed. If none of the cases produces a match then the statements following **otherwise** are executed. For example:

```
switch p
    case 1
        y = sum(abs(x));
    case 2
        y = sqrt(x'*x);
    case inf
        y = max(abs(x));
    otherwise
        error(' p must be 1, 2, or inf.');
end
```

This example evaluates the p-norm of a vector x.