Control Structures

Dr. Mihail

October 16, 2018

Overview

So far in this course, MATLAB programs consisted of a **ordered sequence** of mathematical operations, functions, etc. In other words, these programs have had a **sequential structure**. We will learn two other structures:

- selection structures
- repetition (loop) structures

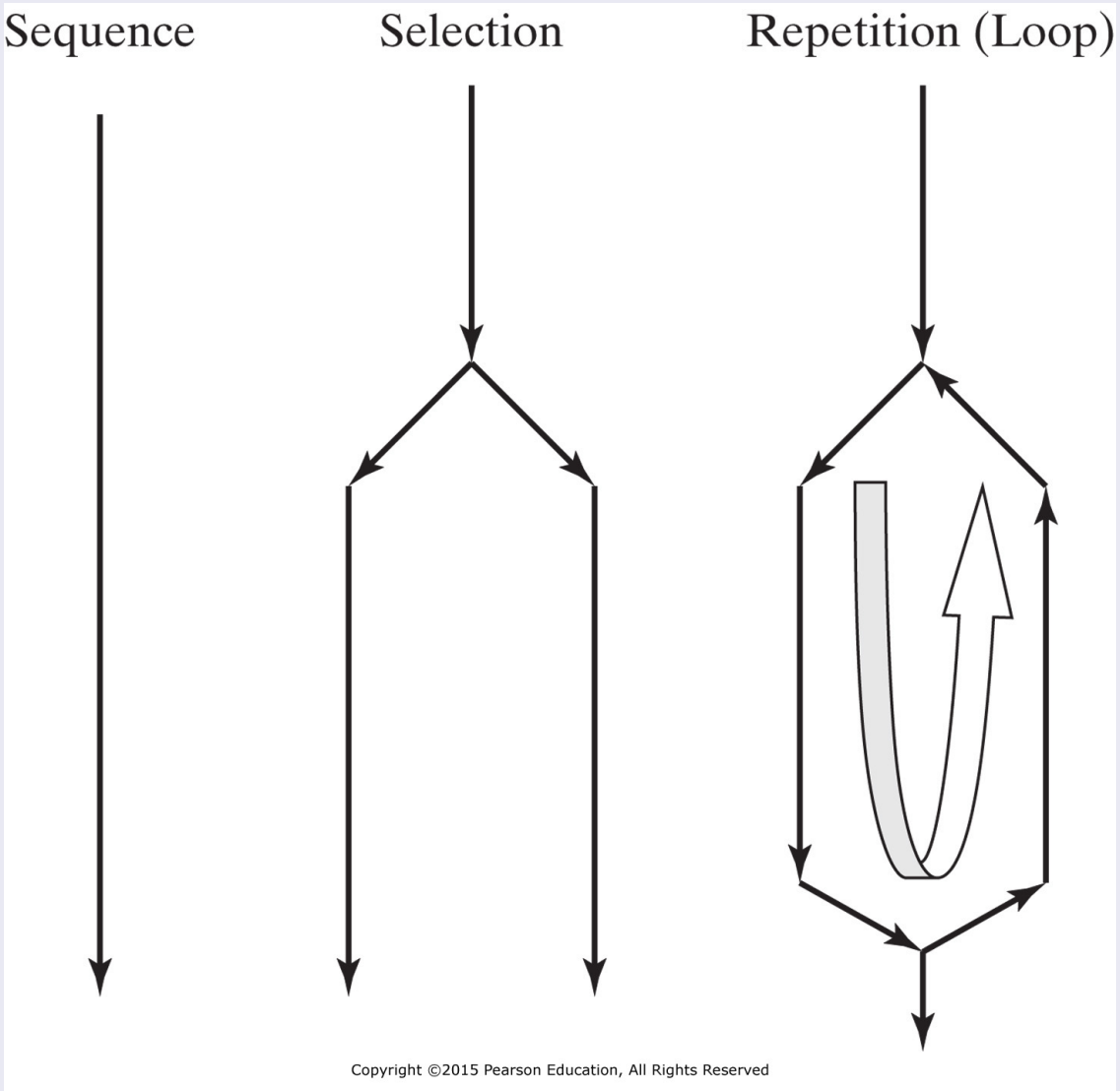
Selection and Repetition

Definition: a selection structure allows execution of a set of commands when some selection criteria is true, **or** a second set of commands when the selection criteria is false.
The selection criteria consists of one or more **logical conditions** that can evaluate to EITHER true or false (never both). The evaluation of these conditions often involves the use of **relational** and **logical** operators.

Definition: a repetition structure (or loop) allows repeated (zero, one or more times) execution of a set of commands. The number of times a loop is executed can depend on:

- a counter OR
- result of a logical condition

Selection and Repetition



Relational Operators

Definition: relational operators define relations between two entities of the same size (matrices, vectors and scalars). Relations include **numerical equality** and **inequality**.

Table 5.1 Relational Operators

Relational Operator	Interpretation
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
~=	not equal to

Relational Operators

Definition: relational operators define relations between two entities of the same size (matrices, vectors and scalars). Relations include **numerical equality** and **inequality**.

Table 5.1 Relational Operators

Relational Operator	Interpretation
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
~=	not equal to

Relational Operators

The result of equality or inequality tests are either true (numeric value of 1) of false (numeric value of 0). This data type is called **boolean**.
Example:

```
>> a = [1 2 3 4 5];
>> b = [1 3 1 6 3];
>> a>b
ans =
     0     0     1     0     1
>> a==b
ans =
     1     0     0     0     0
```

Logical Operators

One can combine boolean values with logical operators:

Table 5.2 Logical Operators

Logical Operator	Interpretation
&	and
~	not
|	or

Selection and Repetition

Examples in Class

Please take notes!

Truth Tables

AND (&)

A	B	(	A	&	B	)
$\top$	$\top$		$\top$	$\top$	$\top$	
$\top$	F		$\top$	F	F	
F	$\top$		F	F	$\top$	
F	F		F	F	F	

Truth Tables

OR (|)

A	B	(	A	|	B	)
$\top$	$\top$		$\top$	F	$\top$	
$\top$	F		$\top$	$\top$	F	
F	$\top$		F	$\top$	$\top$	
F	F		F	$\top$	F	

Truth Tables

NOT

A	$\sim$	A
$\top$	F	$\top$
F	$\top$	F

Truth Tables

Expression: (A & (∼ B | C))

A	B	C	(	A	&	(	∼	B	|	C	)	)
⊤	⊤	⊤		⊤	⊤		*F*	⊤	⊤	⊤		
⊤	⊤	*F*		⊤	⊤		*F*	⊤	⊤	*F*		
⊤	*F*	⊤		⊤	*F*		⊤	*F*	*F*	⊤		
⊤	*F*	*F*		⊤	⊤		⊤	*F*	⊤	*F*		
F	⊤	⊤		*F*	*F*		*F*	⊤	⊤	⊤		
F	⊤	*F*		*F*	*F*		*F*	⊤	⊤	*F*		
F	*F*	⊤		*F*	*F*		⊤	*F*	*F*	⊤		
F	*F*	*F*		*F*	*F*		⊤	*F*	⊤	*F*		

Operator Precedence

1. Parentheses ()
2. Transpose, power, complex conjugate transpose, matrix power
3. Unary plus, unary minus, logical negation
4. Multiplication (.*), division (./), matrix multiplication (*), matrix division (/)
5. Addition (+), subtraction (-)
6. Colon operator (:)
7. Less than, less than or equal to, greater than, greater than or equal to, equal to, not equal to
8. Element-wise AND
9. Element-wise OR

Operator Precedence

MATLAB always gives the & operator precedence over the | operator. Although MATLAB typically evaluates expressions from left to right, the expression $a|b\&c$ is evaluated as $a|(b\&c)$. It is a good idea to use parentheses to explicitly specify the intended precedence of statements containing combinations of & and |.

Selection Structures

Two selection structures:

- `find` command
- family of `if` structures

find

`find(x)` returns a vector composed of the indices of the nonzero elements of `x`. These can be used in further commands, and often eliminates the need for `if` or loop structures.

Trivial example:

```
>> x = [1 2 3 0 0 1 2 3];
>> find(x)

ans =

     1     2     3     6     7     8
```

find

`find(x)` returns a vector composed of the indices of the nonzero elements of `x`. These can be used in further commands, and often eliminates the need for `if` or loop structures.

Another example:

```
>> x = [1 2 3 0 0 1 2 3];
>> y = (x==0);
>> find(y)

ans =

     4     5
```

find

```
>> A = magic(8)

A =

    64     2     3    61    60     6     7    57
     9    55    54    12    13    51    50    16
    17    47    46    20    21    43    42    24
    40    26    27    37    36    30    31    33
    32    34    35    29    28    38    39    25
    41    23    22    44    45    19    18    48
    49    15    14    52    53    11    10    56
     8    58    59     5     4    62    63     1
```

find

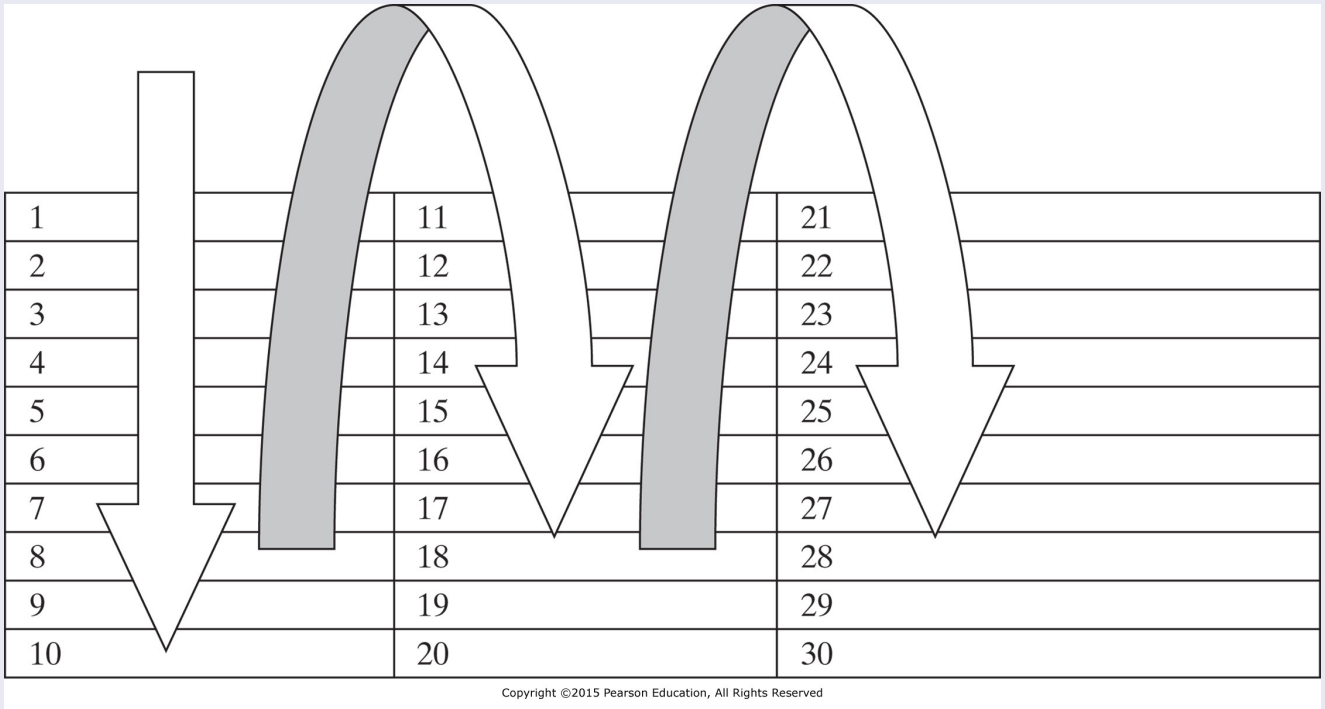
```
>> A = magic(8);
>> indices = find(A>50)

indices =

     1
    10
    16
    18
    24
    25
    31
    33
    39
    42
    48
    56
    57
    63
```

Linear indices

Linear indices. Example: 10 by 3 matrix



Linear indices

```
>> A = magic(3)
A =

     8     1     6
     3     5     7
     4     9     2

>> A(:)

ans =

     8
     3
     4
     1
     5
     9
     6
     7
     2
```

one-way if statements

Syntax

```
if condition
  statements
end
```

Example:

```
x = input('Please enter positive x: ');
if(x<0)
  fprintf('You have entered a negative number!\n');
end
```

two-way if statements

Syntax

```
if condition
  statements when condition is true
else
  statements when condition is false
end
```

Example:

```
x = input('Please enter x between 10 and 20 : ');
if( (x>=10)&(x<=20) )
  fprintf('Good input!\n');
else
  fprintf('Bad input!\n');
end
```

multi-way if statements

Syntax

```
if condition
  statements when condition is true
elseif another_condition
  statements when another_condition is true
elseif yet_another_condition
  statements when yet_another_condition is true
else
  statements when all conditions above are false
end
```

multi-way if statements

Example

```
if temerature>100
  disp('Too hot-equipment malfunctioning');
elseif temperature>90
  disp('Normal operating temperature');
elseif temperature>50
  disp('Temperature below desired operating range');
else
  disp('Too cold-turn off equipment');
end
```

while loops

Syntax

```
while condition
  statements when condition is true
end
```

Example: counting with a counter

```
c = 1;
while(c<=10)
  fprintf('%d\n', c);
  c = c + 1;
end
```

for loops

Syntax

```
for index = sequence
  body
end
```

Example: evaluate $s = 1 + 2 + 3 + ... + n$ for $n = 10$:

```
n = 10;
s = 0;
for i = 1:n
  s = s + i;
end
fprintf('s = %d', s);
```