Introduction to MATLAB

A. Prof./ Ahmed Nagib Elmekawy

Mechanical Engineering department, Alexandria University, Egypt

Spring 2023

Chapter 4

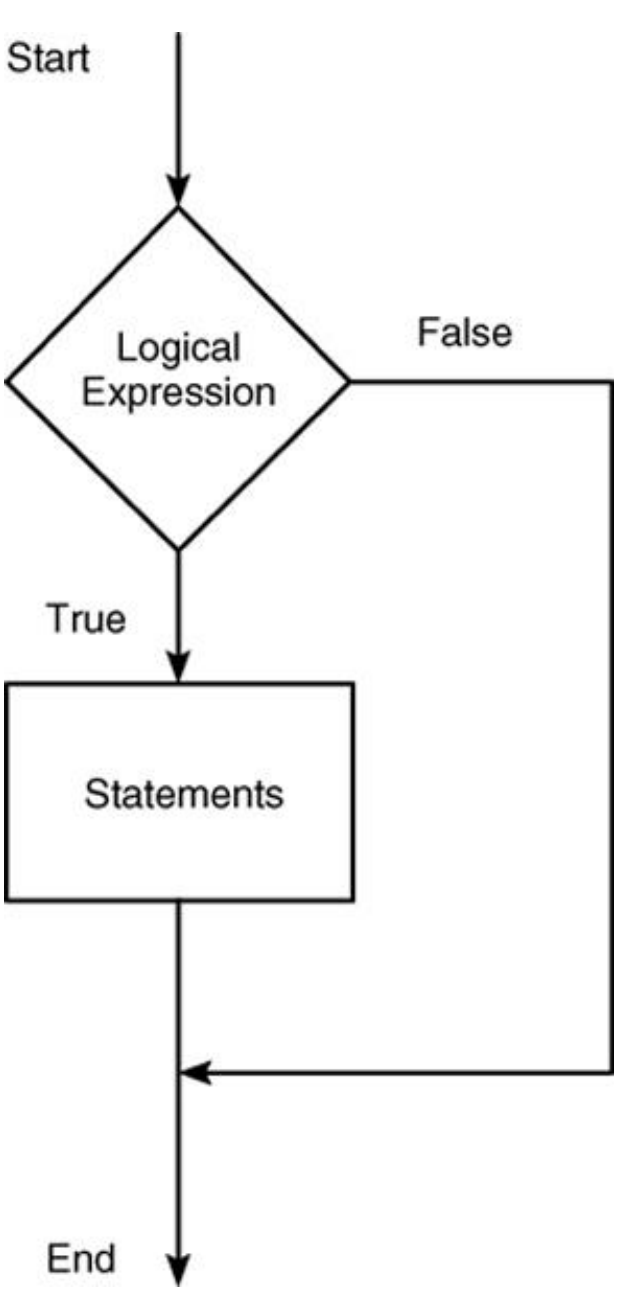
Decision making and looping functions

(If, for and while functions)

Flowcharts are useful for developing and documenting programs that contain conditional statements, because they can display the various paths (called “branches”) that a program can take, depending on how the conditional statements are executed.

Flowchart representation of the if statement.

```
if logical expression
        statements
end
```



Relational operators

Operator	Meaning
<	Less than.
<=	Less than or equal to.
>	Greater than.
>=	Greater than or equal to.
==	Equal to.
~=	Not equal to.

Relational operators Examples

```
>> 5>8
ans =
     0
>> a=5<10
a =
     1
```

Checks if 5 is larger than 8.

Since the comparison is false (5 is not larger than 8) the answer is 0.

Checks if 5 is smaller than 10, and assigns the answer to a.

Since the comparison is true (5 is smaller than 10) the number 1 is assigned to a.

Relational operators Examples

```
>> y=(6<10)+(7>8)+(5*3==60/4)
```

Using relational operators in math expression.

Equal to 1 since 6 is smaller than 10.

Equal to 0 since 7 is not larger than 8.

Equal to 1 since 5*3 is equal to 60/4.

```
y =
     2
```

```
>> b=[15 6 9 4 11 7 14]; c=[8 20 9 2 19 7 10];
```

Define vectors b and c.

```
>> d=c>=b
```

Checks which c elements are larger than or equal to b elements.

```
d =
     0     1     1     0     1     1     0
```

Assigns 1 where an element of c is larger than or equal to an element of b.

```
>> b == c
```

Checks which b elements are equal to c elements.

```
ans =
     0     0     1     0     0     1     0
```

```
>> b~=c
```

Checks which b elements are not equal to c elements.

```
ans =
     1     1     0     1     1     0     1
```

```
>> f=b-c>0
```

Subtracts c from b and then checks which elements are larger than zero.

```
f =
     1     0     0     1     0     0     1
```

Relational operators Examples

```
>> A=[2 9 4; -3 5 2; 6 7 -1]
A =
     2     9     4
    -3     5     2
     6     7    -1
>> B=A<=2
```

Define a 3×3 matrix A.

Checks which elements in A are smaller than or equal to 2. Assigns the results to matrix B.

```
B =
     1     0     0
     1     0     1
     0     0     1
```

Relational operators Examples

```
>> r = [8 12 9 4 23 19 10]
r =
     8    12     9     4    23    19    10
>> s=r<=10
s =
     1     0     1     1     0     0     1
>> t=r(s)
t =
     8     9     4    10
>> w=r(r<=10)
w =
     8     9     4    10
```

Define a vector r.

Checks which r elements are smaller than or equal to 10.

A logical vector s with 1s at positions where elements of r are smaller than or equal to 10.

Use s for addresses in vector r to create vector t.

Vector t consists of elements of r in positions where s has 1s.

The same procedure can be done in one step.

Relational operators Examples

```
>> 3+4<16/2
ans =
     1
>> 3+(4<16)/2
ans =
    3.5000
```

+ and / are executed first.

The answer is 1 since 7 < 8 is true.

4 < 16 is executed first, and is equal to 1, since it is true.

3.5 is obtained from 3 + 1/2.

Relational operators Order

Precedence	Operation
1 (highest)	Parentheses (if nested parentheses exist, inner ones have precedence)
2	Exponentiation
3	Logical NOT (~)
4	Multiplication, division
5	Addition, subtraction
6	Relational operators (>, <, >=, <=, ==, ~=)
7	Logical AND (&)
8 (lowest)	Logical OR (|)

The arithmetic operators +, -, *, /, and \ have precedence over the relational operators. Thus the statement

```
z = 5 > 2 + 7
```

is equivalent to

```
z = 5 >(2+7)
```

and returns the result `z = 0`.

We can use parentheses to change the order of precedence; for example, `z = (5 > 2) + 7` evaluates to `z = 8`.

The logical Class

When the relational operators are used, such as

```
x = (5 > 2)
```

they create a *logical* variable, in this case, `x`.

Logical is a first-class data type and a MATLAB class, and so logical is now equivalent to other first-class types such as character and cell arrays.

Logical variables may have only the values 1 (true) and 0 (false).

Just because an array contains only 0s and 1s, however, it is not necessarily a logical array. For example, in the following session `k` and `w` appear the same, but `k` is a logical array and `w` is a numeric array, and thus an error message is issued.

```
>>x = -2:2; k = (abs(x)>1)
k =
    1    0    0    0    1
>>z = x(k)
z =
   -2    2
>>w = [1,0,0,0,1]; v = x(w)
??? Subscript indices must either be real
  positive... integers or logicals.
```

Logical operators:

Logical operator	Name	Description
& Example: A&B	AND	Operates on two operands (A and B). If both are true, the result is true (1); otherwise the result is false (0).
| Example: A|B	OR	Operates on two operands (A and B). If either one, or both, are true, the result is true (1); otherwise (both are false) the result is false (0).
~ Example: ~A	NOT	Operates on one operand (A). Gives the opposite of the operand; true (1) if the operand is false, and false (0) if the operand is true.

Logical operators Examples:

```
>> 3&7
```
3 AND 7.

```
ans =
     1
```
3 and 7 are both true (nonzero), so the outcome is 1.

```
>> a=5|0
```
5 OR 0 (assign to variable a).

```
a =
     1
```
1 is assigned to a since at least one number is true (nonzero).

```
>> ~25
```
NOT 25.

```
ans =
     0
```
The outcome is 0 since 25 is true (nonzero) and the opposite is false.

```
>> t=25*((12&0)+(~0)+(0|5))
```
Using logical operators in a math expression.

```
t =
    50
```

Logical operators Examples:

```
>> x=[9 3 0 11 0 15]; y=[2 0 13 -11 0 4];
```

Define two vectors x and y.

```
>> x&y
ans =
     1     0     0     1     0     1
```

The outcome is a vector with 1 in every position where both x and y are true (nonzero elements), and 0s otherwise.

```
>> z=x|y
z =
     1     1     1     1     0     1
```

The outcome is a vector with 1 in every position where either or both x and y are true (nonzero elements), and 0s otherwise.

```
>> ~(x+y)
ans =
     0     0     0     1     1     0
```

The outcome is a vector with 0 in every position where the vector x + y is true (nonzero elements), and 1 in every position where x + y is false (zero elements).

The `if` Statement

The `if` statement's basic form is

```
if logical expression
   statements
end
```

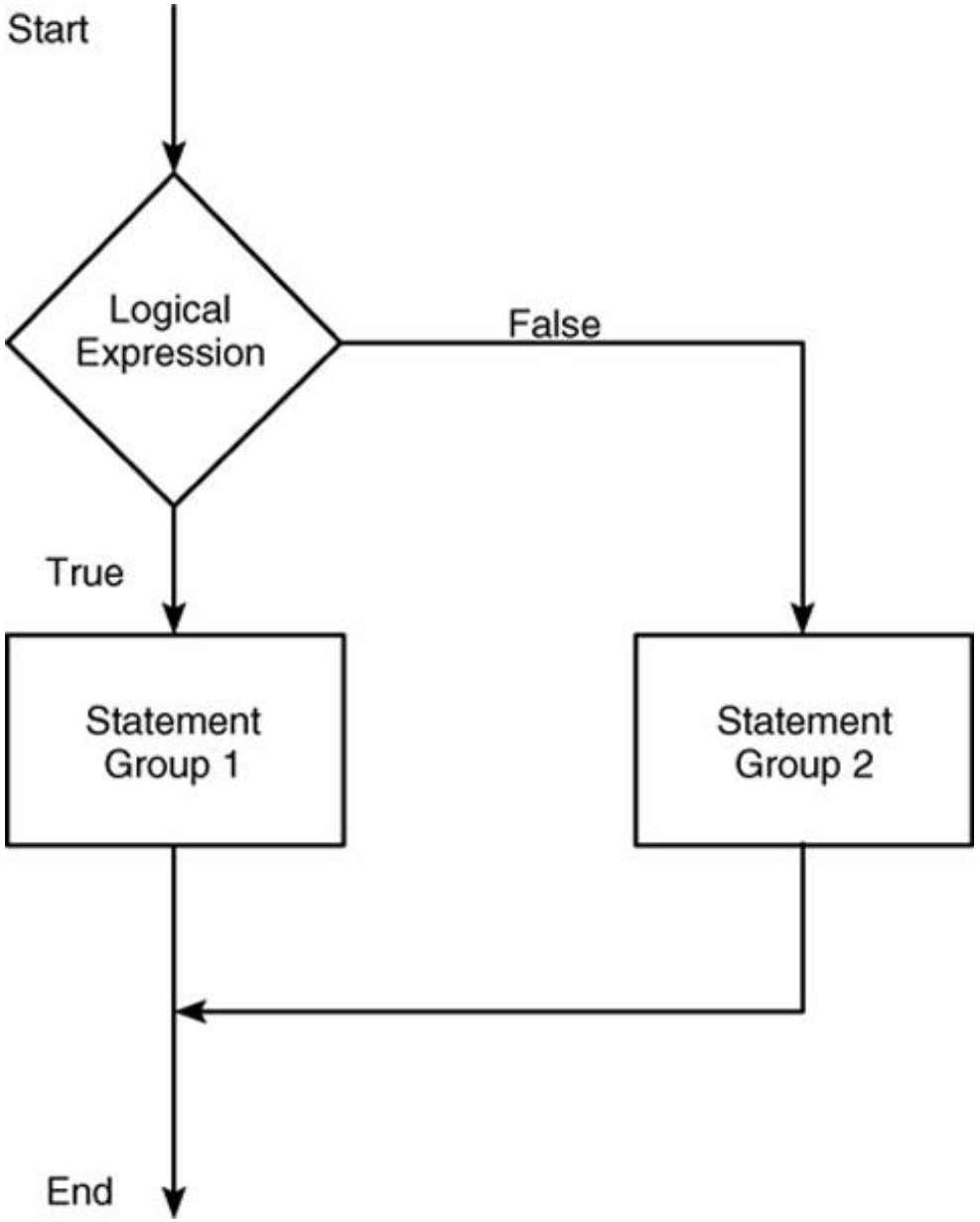
Every `if` statement must have an accompanying end statement. The end statement marks the end of the *statements* that are to be executed if the *logical expression* is true.

The `else` Statement

The basic structure for the use of the `else` statement is

```
if logical expression
  statement group 1
else
  statement group 2
end
```

Flowchart of the else structure.



When the test, if *logical expression,* is performed, where the logical expression may be an *array,* the test returns a value of true only if *all* the elements of the logical expression are true!

Examples:

```
if a < b
if c >= 5
if a == b
if a ~= 0
if (d<h)&(x>7)
if (x~=13)|(y<0)
```

All the variables must have assigned values.

For example, if we fail to recognize how the test works, the following statements do not perform the way we might expect.

```
x = [4,-9,25];
if x < 0
   disp('Some of the elements of x are
negative.')
else
   y = sqrt(x)
end
```

When this program is run it gives the result

```
y =
  2   0 + 3.000i   5
```

Instead, consider what happens if we test for `x` positive.

```
x = [4,-9,25];
if x >= 0
  y = sqrt(x)
else
  disp('Some of the elements of x are
negative.')
end
```

When executed, it produces the following message:

```
Some of the elements of x are negative.
```

The test if `x < 0` is false, and the test if `x >= 0` also returns a false value because `x >= 0` returns the vector `[1,0,1]`.

The statements

```
if logical expression 1
   if logical expression 2
    statements
  end
end
```

can be replaced with the more concise program

```
if logical expression 1 & logical expression 2
  statements
end
```

Instead, consider what happens if we test for `x` positive.

```
clc; clear
x = [4,9,25];
y=[4 6 8];
if x >= 0
   if y>=0
      z = x+y
   end
else
disp('Some of the elements of x and/or y are
negative.')
end
```

When executed, it produces the following message:

```
z =   8   15   33
```

Instead, consider what happens if we test for `x` positive.

```
clc; clear
x = [4,9,25];
y=[4 6 8];
if x >= 0 & y>=0
  z = x+y
else
disp('Some of the elements of x and/or y are
negative.')
end
```

When executed, it produces the following message:

```
z =   8   15   33
```

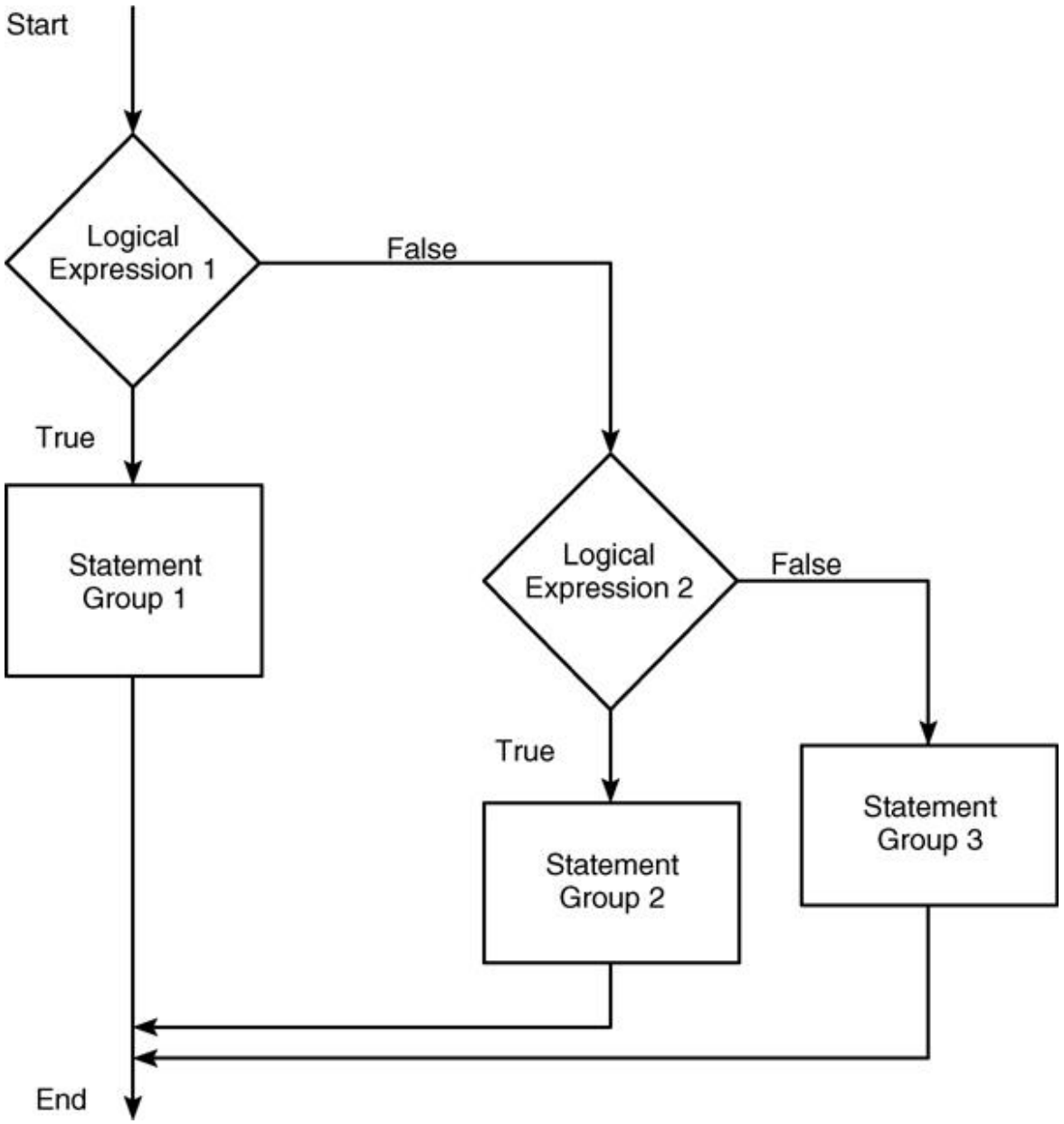
The `elseif` Statement

The general form of the `if` statement is

```
if logical expression 1
     statement group 1
elseif logical expression 2
     statement group 2
else
     statement group 3
end
```

The `else` and `elseif` statements may be omitted if not required. However, if both are used, the `else` statement must come after the `elseif` statement to take care of all conditions that might be unaccounted for.

Flowchart for the general `if-elseif-else` structure.



For example, suppose that `y = 2*x` for `x > 10`, `y =sqrt(x)` for `0 <= x <= 10`, and `y = exp(x) - 1` for `x < 0`. The following statements will compute `y` if `x` already has a scalar value.

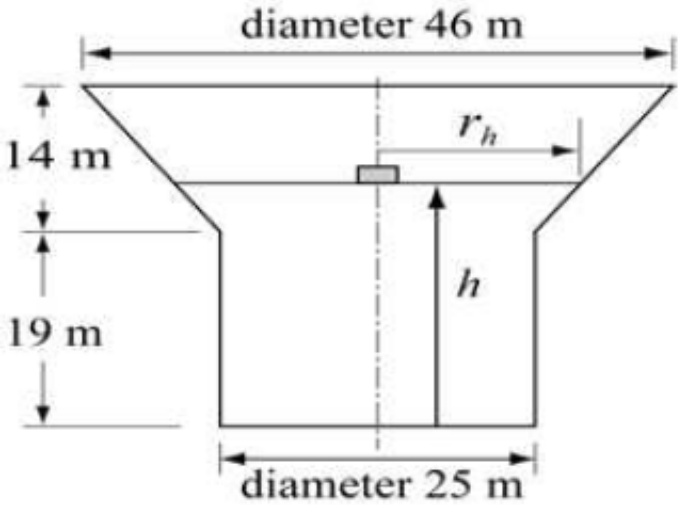
```
x = [4,9,25];
if x > 10
  y = 2*x;
elseif x >= 0
  y = sqrt(x)
else
  y = exp(x) - 1
End
```

When executed, it produces the following message:

```
y =
 2     3     5
```

Sample Problem: Water level in water tower

The tank in a water tower has the geometry shown in the figure (the lower part is a cylinder and the upper part is an inverted frustum of a cone). Inside the tank there is a float that indicates the level of the water. Write a MATLAB program that determines the volume of the water in the tank from the position (height h) of the float. The program asks the user to enter a value of h in m, and as output displays the volume of the water in m^3.



Sample Problem: Water level in water tower

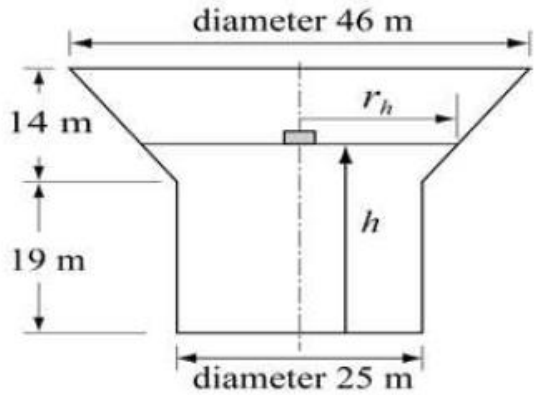
Solution

For $0 \le h \le 19$ m the volume of the water is given by the volume of a cylinder with height h: $V = \pi 12.5^2 h$.

For $19 < h \le 33$ m the volume of the water is given by adding the volume of a cylinder with $h = 19$ m, and the volume of the water in the cone:

$$V = \pi 12.5^2 \cdot 19 + \frac{1}{3}\pi(h - 19)(12.5^2 + 12.5 \cdot r_h + r_h^2)$$

where $r_h = 12.5 + \frac{10.5}{14}(h - 19)$.



Sample Problem: Water level in water tower

```
% The program calculates the volume of the water in the
water tower.
h=input('Please enter the height of the float in meter  ');
if h > 33
   disp('ERROR. The height cannot be larger than 33 m.')
elseif h < 0
   disp('ERROR. The height cannot be a negative number.')
elseif h <= 19
   v = pi*12.5^2*h;
   fprintf('The volume of the water is %7.3f cubic meter.\n',v)
else
   rh=12.5+10.5*(h-19)/14;
   v=pi*12.5^2*19+pi*(h-19)*(12.5^2+12.5*rh+rh^2)/3;
   fprintf('The volume of the water is %7.3f cubic meter.\n',v)
end
```

Sample Problem: Water level in water tower

The following is the display in the Command Window when the program is used with three different values of water height.

```
Please enter the height of the float in meter  8
The volume of the water is 3926.991 cubic meter.

Please enter the height of the float in meter  25.7
The volume of the water is 14114.742 cubic meter.

Please enter the height of the float in meter  35
ERROR. The height cannot be larger than 33 m.
```

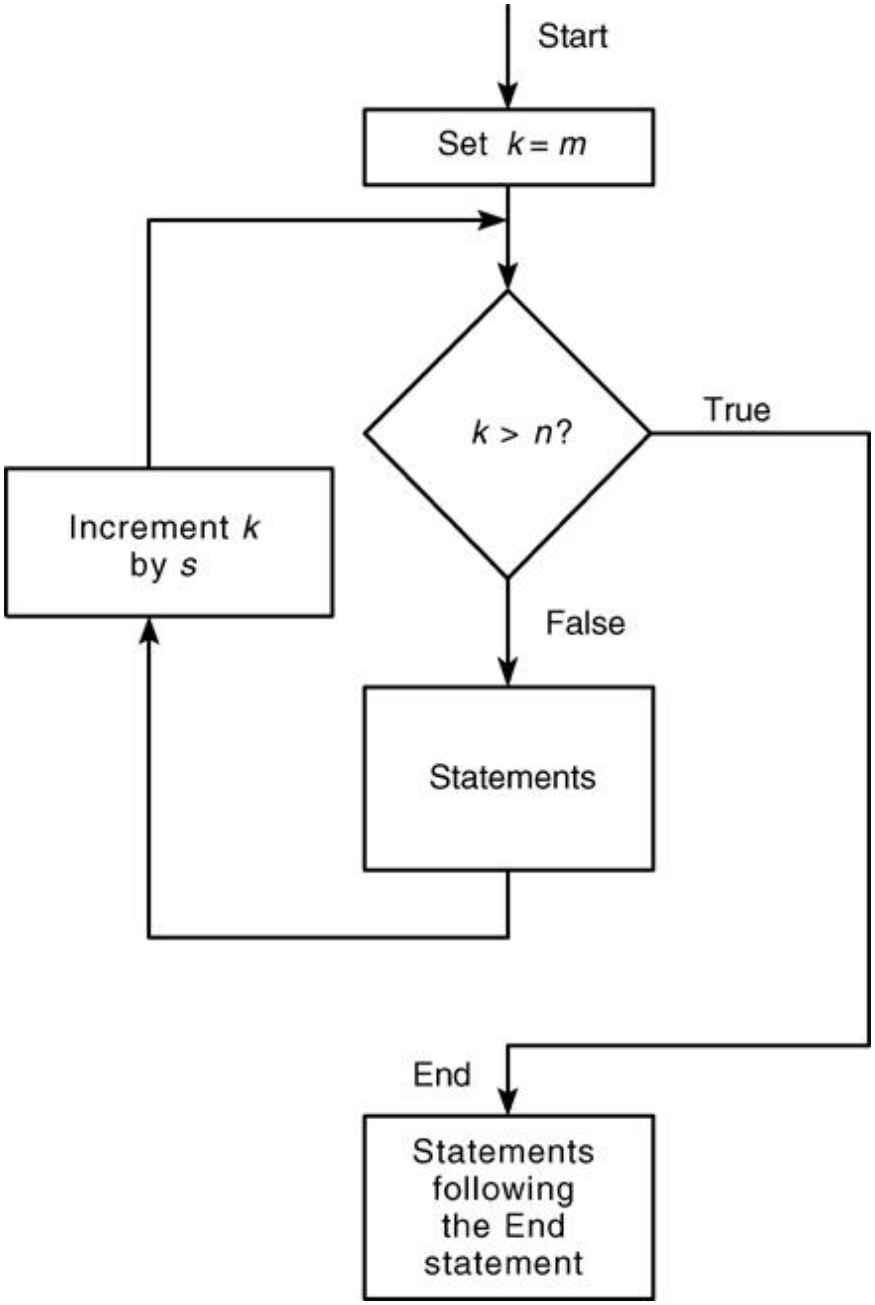
for Loops

A simple example of a `for` loop is

```
for k = 5:10:35
  x = k^2
end
```

The *loop variable* `k` is initially assigned the value 5, and `x` is calculated from `x = k^2`. Each successive pass through the loop increments `k` by 10 and calculates x until k exceeds 35. Thus `k` takes on the values 5, 15, 25, and 35, and `x` takes on the values 25, 225, 625, and 1225. The program then continues to execute any statements following the end statement.

Flowchart of a for Loop.



Note the following rules when using for loops with the loop variable expression `k = m:s:n`:

- The step value `s` may be negative.
  Example: `k = 10:-2:4` produces `k` = 10, 8, 6, 4.
- If `s` is omitted, the step value defaults to one.
- If `s` is positive, the loop will not be executed if `m` is greater than `n`.
- If `s` is negative, the loop will not be executed if `m` is less than `n`.
- If `m` equals `n`, the loop will be executed only once.
- If the step value s is not an integer, round-off errors can cause the loop to execute a different number of passes than intended.

For example, the following code uses a continue statement to avoid computing the logarithm of a negative number.

```
x = [10,100,1000];
for k = 1:length(x)
  y(k) = sqrt(x(k));
end
y
```

The result is `y = 3.16, 10, 31.6`.

`while` Loops

The `while` loop is used when the looping process terminates because a specified condition is satisfied, and thus the number of passes is not known in advance. A simple example of a while loop is

```
x = 5;
while x < 25
   disp(x)
   x = 2*x - 1;
end
```

The results displayed by the `disp` statement are 5, 9, and 17.

`while` Loops

The `while` loop is used when the looping process terminates because a specified condition is satisfied, and thus the number of passes is not known in advance. A simple example of a while loop is

```
x = 1;
while x < 25
   disp(x)
   x = 2*x - 1;
end
```

The while loop will be running forever, why?

The typical structure of a while loop follows.

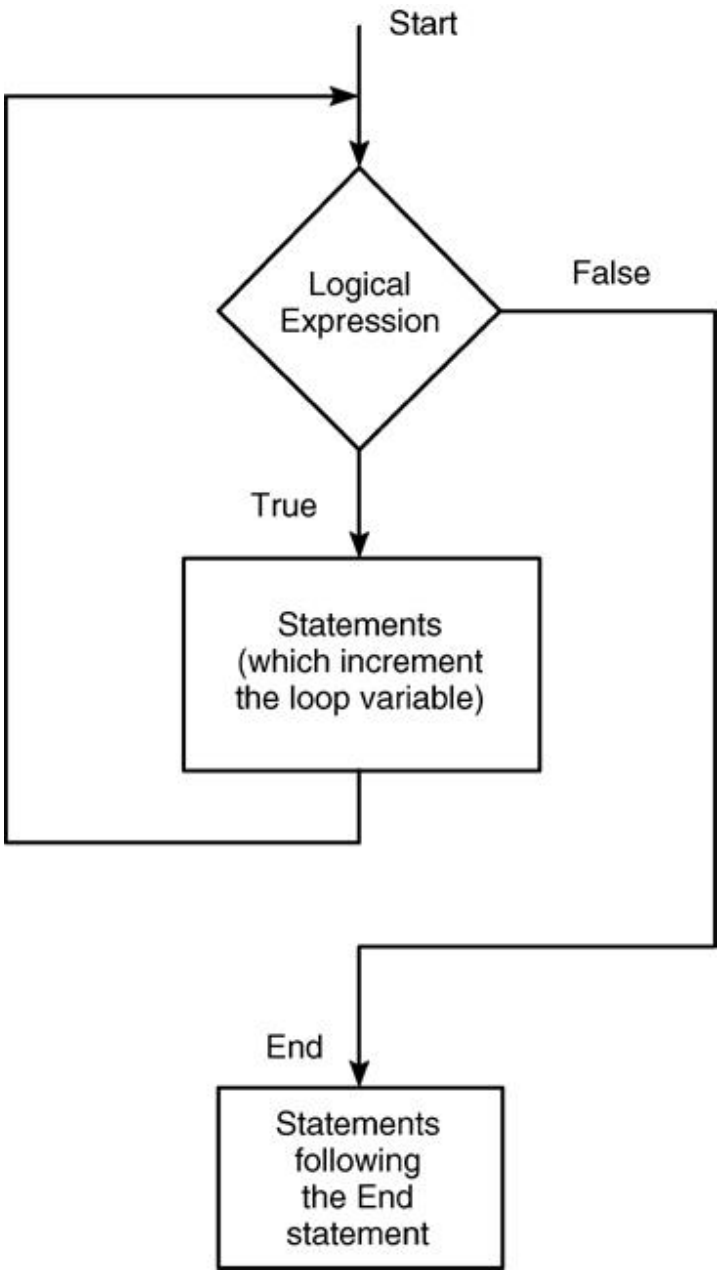
```
while logical expression
    statements
end
```

For the `while` loop to function properly, the following two conditions must occur:

1. The loop variable must have a value before the while statement is executed.

2. The loop variable must be changed somehow by the *statements.*

Flowchart of the while loop.



Sample Problem: Flight of a model rocket

The flight of a model rocket can be modeled as follows. During the first 0.15 s the rocket is propelled upward by the rocket engine with a force of 16 N. The rocket then flies up while slowing down under the force of gravity. After it reaches the apex, the rocket starts to fall back down. When its downward velocity reaches 20 m/s, a parachute opens (assumed to open instantly), and the rocket continues to drop at a constant speed of 20 m/s until it hits the ground. Write a program that calculates and plots the speed and altitude of the rocket as a function of time during the flight.

Sample Problem: Flight of a model rocket

Solution

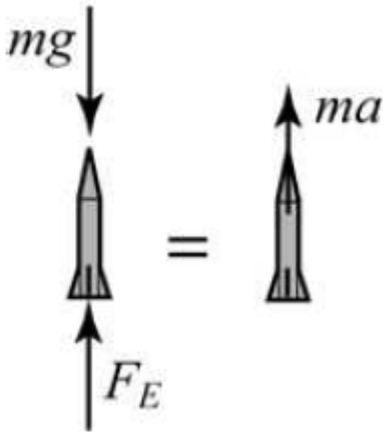
The rocket is assumed to be a particle that moves along a straight line in the vertical plane. For motion with constant acceleration along a straight line, the velocity and position as a function of time are given by:

$$v(t) = v_0 + at \quad \text{and} \quad s(t) = s_0 + v_0 t + \frac{1}{2}at^2$$

where v_0 and s_0 are the initial velocity and position, respectively. In the computer program the flight of the rocket is divided into three segments. Each segment is calculated in a `while` loop. In every pass the time increases by an increment.

Sample Problem: Flight of a model rocket

Segment 1: The first 0.15 s when the rocket engine is on. During this period, the rocket moves up with a constant acceleration. The acceleration is determined by drawing a free body and a mass acceleration diagram (shown on the right). From Newton's second law, the sum of the forces in the vertical direction is equal to the mass times the acceleration (equilibrium equation):



$$+\uparrow \Sigma F = F_E - mg = ma$$

Solving the equation for the acceleration gives:

$$a = \frac{F_E - mg}{m}$$

The velocity and height as a function of time are:

$$v(t) = 0 + at \quad \text{and} \quad h(t) = 0 + 0 + \frac{1}{2}at^2$$

where the initial velocity and initial position are both zero. In the computer program this segment starts at $t = 0$, and the looping continues as long as $t < 0.15$ s. The time, velocity, and height at the end of this segment are t_1, v_1, and h_1.

Sample Problem: Flight of a model rocket

Segment 2: The motion from when the engine stops until the parachute opens. In this segment the rocket moves with a constant deceleration g. The speed and height of the rocket as functions of time are given by:

$$v(t) = v_1 - g(t - t_1) \quad \text{and} \quad h(t) = h_1 + v_1(t - t_1) - \frac{1}{2}g(t - t_1)^2$$

In this segment the looping continues until the velocity of the rocket is –20 m/s (negative since the rocket moves down). The time and height at the end of this segment are t_2 and h_2.

Segment 3: The motion from when the parachute opens until the rocket hits the ground. In this segment the rocket moves with constant velocity (zero acceleration). The height as a function of time is given by $h(t) = h_2 - v_{chute}(t - t_2)$, where v_{chute} is the constant velocity after the parachute opens. In this segment the looping continues as long as the height is greater than zero.

Sample Problem: Flight of a model rocket

```
m=0.05; g=9.81; tEngine=0.15; Force=16; vChute=-20; Dt=0.01;
clear t v h
n=1;
t(n)=0; v(n)=0; h(n)=0;
% Segment 1
a1=(Force-m*g)/m;
while t(n) < tEngine & n < 50000
    n=n+1;
    t(n)=t(n-1)+Dt;
    v(n)=a1*t(n);
    h(n)=0.5*a1*t(n)^2;
end
v1=v(n); h1=h(n); t1=t(n);
% Segment 2
while v(n) >= vChute & n < 50000
    n=n+1;
    t(n)=t(n-1)+Dt;
    v(n)=v1-g*(t(n)-t1);
```

The first `while` loop.

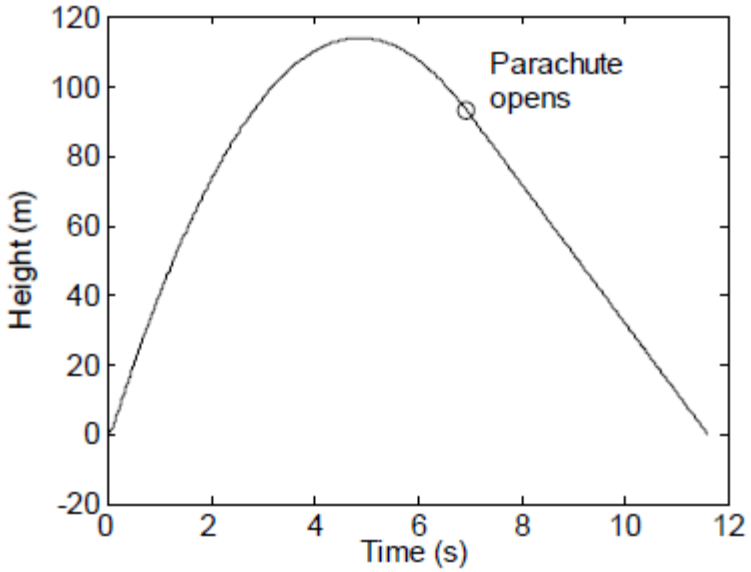
The second `while` loop.

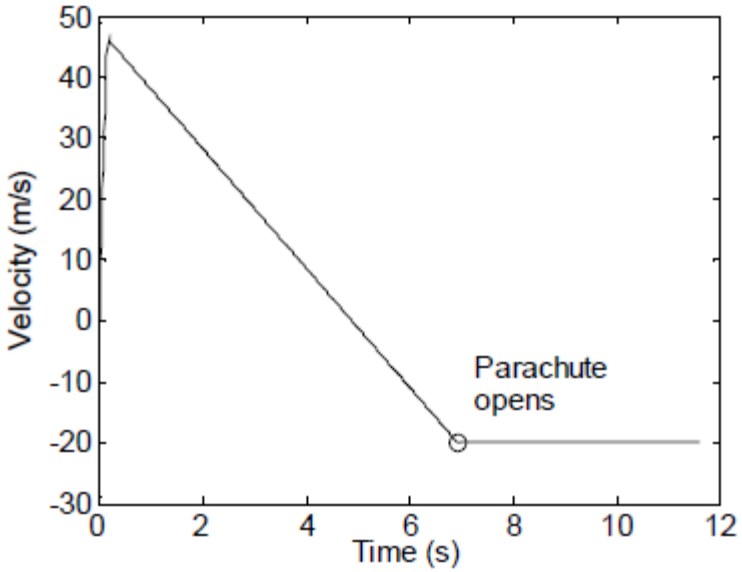
Sample Problem: Flight of a model rocket

```
    h(n)=h1+v1*(t(n)-t1)-0.5*g*(t(n)-t1)^2;
end
v2=v(n); h2=h(n); t2=t(n);
% Segment 3
while h(n) > 0 & n < 50000
   n=n+1;
    t(n)=t(n-1)+Dt;
    v(n)=vChute;
    h(n)=h2+vChute*(t(n)-t2);
end
subplot(1,2,1)
plot(t,h,t2,h2,'o')
subplot(1,2,2)
plot(t,v,t2,v2,'o')
```

The third `while` loop.

Sample Problem: Flight of a model rocket





Sample Problem: Colebrook equation - try and error

$$\frac{1}{\sqrt{f}} = -2\log\left(\frac{\epsilon}{3.7\,D_h} + \frac{2.51}{Re\,\sqrt{f}}\right)$$

Example 5.3

Water flows through a 20-in pipe at 5700 gal/min. Calculate the friction factor using Colebrook–White equation. Assume 0.375-in. pipe wall thickness and an absolute roughness of 0.002 in. Use specific gravity of 1.00 and viscosity of 1.0 cSt. What is the head loss resulting from friction in 2500 ft of pipe?

Solution

First we calculate the Reynolds number from Eqn (5.17) as follows:

$$R = 3{,}160 \times 5{,}700/(19.25 \times 1.0) = 935{,}688$$

The flow is fully turbulent and the friction factor f is calculated using Eqn (5.21) as follows:

$$1/\sqrt{f} = -2\log_{10}\left[\left(0.002/\left(3.7 \times 19.25\right)\right) + 2.51/\left(935{,}688\sqrt{f}\right)\right]$$

This implicit equation for f must be solved by trial and error.

First assume a trial value of $f = 0.02$. Substituting in equation above, we get a successive approximations for f as follows:

$$f = 0.0133,\ 0.0136 \text{ and } 0.0136$$

Therefore the solution is $f = 0.0136$.

Using Eqn (5.12),

$$\text{velocity} = 0.4085\left(5700\right)/19.25^2 = 6.28 \text{ ft/s}$$

Using Eqn (5.19), head loss from friction is

$$h = 0.0136 \times \left(2{,}500 \times 12/19.25\right) \times 6.28^2/64.4 = 12.98 \text{ ft}$$