

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

دانشگاه شهید رجایی
دانشکده فنی مهندسی
گروه مکانیک خودرو



کاربرد نرم افزار در مهندسی و صنعت خودرو
MATLAB
بخش: برنامه نویسی (Programming)

مدرس: شهاب داودی
davoudi@gmail.com
davoudi.110mb.com

davoudi.110mb.com

Flow Control

MATLAB has several flow control constructs:

- ▶ if statements
- ▶ switch statements
- ▶ for loops
- ▶ while loops
- ▶ continue statements
- ▶ break statements

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

3

if

- ▶ The if statement evaluates a logical expression and executes a group of statements when the expression is *true*.
- ▶ The optional `elseif` and `else` keywords provide for the execution of alternate groups of statements.
- ▶ An `end` keyword, which matches the `if`, terminates the last group of statements.
- ▶ The groups of statements are delineated by the four keywords – no braces or brackets are involved.

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

4

davoudi@gmail.com

if

```
if expression
    statements
end
```

- ▶ MATLAB evaluates the `expression` and, if the evaluation yields logical 1 (true) or a nonzero result, executes one or more MATLAB commands denoted here as `statements`.
- ▶ When you are nesting `ifs`, each `if` must be paired with a matching `end`.
- ▶ When using `elseif` and/or `else` within an `if` statement, the general form of the statement is

```
if expression1
    statements1
elseif expression2
    statements2
else
    statements3
end
```

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

5

Control Structures

- ▶ Expressions, relations (`==`, `>`, `|`, `&`, functions, etc.)

```
if/while expression statements end
```

- ▶ Use comma to separate expression from statements if on same line

```
if a == b & isprime(n), M = inv(K); else M = K; end
```

- ▶ for variable = expression statements end

```
for i=1:2:100, s = s / 10; end
```

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

6

if

```
if A > B
    'greater'
elseif A < B
    'less'
elseif A == B
    'equal'
else
    error('Unexpected situation')
end
```

switch and case

- ▶ The `switch` statement executes groups of statements based on the value of a variable or expression.
- ▶ The keywords `case` and `otherwise` delineate the groups.
- ▶ Only the first matching case is executed.
- ▶ There must always be an `end` to match the switch.

davoudi.110mb.com

switch and case

- ▶ The general form of the switch statement is

```
switch switch_expr
    case case_expr
        statement, ..., statement
    case {case_expr1, case_expr2, case_expr3, ...}
        statement, ..., statement
    otherwise
        statement, ..., statement
end
```

`case case_expr` compares the value of the expression `switch_expr` declared in the preceding switch statement with one or more values in `case_expr`, and executes the block of code that follows if any of the comparisons yield a true result.

switch and case

```
A=2
switch A
    case 0
        b=0
    case 1
        b=1
    case 2
        b=2
    otherwise
        b=333
end
```

davoudi@gmail.com

for

- ▶ The `for` loop repeats a group of statements a fixed, predetermined number of times. A matching `end` delineates the statements.

```
for n = 3:32
    r(n) = rank(magic(n));
end
r
```

- ▶ It is a good idea to indent the loops for readability, especially when they are nested.

```
for i = 1:m
    for j = 1:n
        H(i,j) = 1/(i+j);
    end
end
```

for

- ▶ The general format is

```
for variable = expression
    statement
    ...
    statement
end
```

- ▶ The columns of the `expression` are stored one at a time in the variable while the following statements, up to the `end`, are executed.
- ▶ In practice, the `expression` is almost always of the form `scalar : scalar`, in which case its columns are simply scalars.
- ▶ The scope of the `for` statement is always terminated with a matching `end`.

for

Assume `k` has already been assigned a value. Create the Hilbert matrix, using zeros to preallocate the matrix to conserve memory:

```
k=5
a = zeros(k,k) % Preallocate matrix
for m = 1:k
    for n = 1:k
        a(m,n) = 1/(m+n -1);
    end
end
```

► Step `s` with increments of `-0.1`
`for s = 1.0:-0.1: 0.0,...`, end

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

13

while

► The while loop repeats a group of statements an indefinite number of times under control of a logical condition (Repeatedly execute statements while condition is true).
► A matching end delineates the statements.

```
while expression
    statements
end
```

► while repeats statements an indefinite number of times. The statements are executed while the real part of expression has all nonzero elements. expression is usually of the form:

```
expression rel_op expression
```

where rel_op is ==, <, >, <=, >=, or ~=.

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

14

davoudi.110mb.com

while

► Here is a complete program, illustrating while, if, else, and end, that uses interval bisection to find a zero of a polynomial.

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
    x = (a+b)/2;
    fx = x^3-2*x-5;
    if sign(fx) == sign(fa)
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
x
```

► The result is a root of the polynomial $x^3 - 2x - 5$, namely $x = 2.09455148154233$

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

15

While expression and Nonscalar Expression

The cautions involving matrix comparisons apply to the if and while statement.

Given matrices A and B,

```
A =      1      0
      2      3
      B =      1      1
              3      4
```

Expression	EvaluatesAs	Because
<code>A < B</code>	false	<code>A(1,1)</code> is not less than <code>B(1,1)</code> .
<code>A < (B + 1)</code>	true	Every element of A is less than that same element of B with 1 added.
<code>A & B</code>	false	<code>A(1,2)</code> is false, and B is ignored due to short-circuiting. →??
<code>B < 5</code>	true	Every element of B is less than 5.

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

16

davoudi@gmail.com

continue

► continue passes control to the next iteration of the for or while loop in which it appears, skipping any remaining statements in the body of the loop.

► In nested loops, continue passes control to the next iteration of the for or while loop enclosing it.

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

17

continue

► The example below shows a continue loop that counts the lines of code in the file magic.m, skipping all blank lines and comments. A continue statement is used to advance to the next line in magic.m without incrementing the count whenever a blank line or comment line is encountered.

```
fid = fopen('magic.m','r');
count = 0;
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line) | strcmp(line,'% ',1)
        continue
    end
    count = count + 1;
end
disp(sprintf('%d lines',count));
```

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

18

break

► The `break` statement lets you exit early from a `for` or `while` loop. In nested loops, `break` exits from the innermost loop only.

► Here is an improvement on the example from the previous section. Why is this use of `break` a good idea?

```
a = 0; fa = -Inf;
b = 3; fb = Inf;
while b-a > eps*b
    x = (a+b)/2;
    fx = x^3-2*x-5;
    if fx == 0
        break
    elseif sign(fx) == sign(fa)
        a = x; fa = fx;
    else
        b = x; fb = fx;
    end
end
x
```

© 2007 Shahab Davoudi, davoudi@gmail.com, davoudi.110mb.com

19

davoudi.110mb.com

davoudi@gmail.com