Selection Structures
Chapter 8
8.4 Selection Structures

• Most of the time the **find** function should be used instead of an **if**

• However, there are certain situations where **if** is the appropriate process to use
Structured Programming

Format of if statement:

```
if Logical Expression
    Statements
    ...
end
```

Format of if else statement:

```
if Logical Expression
    Statement 1
else
    Statement 2
end
```
Simple if

if *comparison statements*
end

For example:

if G<50
    disp('G is a small value equal to:')
    disp(G);
end
G = 30;
if G < 50
    disp('G is a small value equal to: ')
    disp(G)
end
If statements

• Easy to interpret for scalars
• What does an if statement mean if the comparison includes a matrix?
  • The comparison is only true if it is true for every member of the array
Consider this bit of code

G=[30,55,10]
if G<50
    disp(‘G is a small value equal to:’)
disp(G);
end

The code inside the if statement is not executed, because the comparison is not true!!
This statement is false because at least one of the elements in G has a value \( \geq 50 \).

Therefore the code inside the if statement does not execute.
This statement is true because all of the elements in G are < 70

The output would have been cleaner if we had suppressed line 1 by adding a semicolon to the code.
The if/else structure

• The simple if triggers the execution of a block of code if a condition is true
• If it is false that block of code is skipped, and the program continues without doing anything
• What if instead you want to execute an alternate set of code if the condition is false?
Flow chart of an if/else structure

True

Comparison

False

Block of code to execute if the comparison is true

Block of code to execute if the comparison is false
Structured Programming

Example: *iftest1.m*

```
% Program to test the if statement #1

X=input('Enter value for x:');
if X>=0
    Y=sqrt(X);
    disp(['The squareroot is',num2str(Y)])
end
```

```
>>iftest1
Enter value for x: 9
The squareroot is 3.0000
```

```
>>iftest1
Enter value for x: -2
```

- **Initialization**
- **Input X**
- **If** $X\geq 0$
  - **True**
    - **Calculate** $\sqrt{X}$
    - **Display Result**
  - **False**
- **End of script**
Structured Programming

Example: iftest2.m

```matlab
% Program to test the if statement #2
X=input('Enter value for x:');
if X>=0
    Y=sqrt(X);
    disp(['The squareroot is ',num2str(Y)]);
else
    disp('x is negative: there is no real result')
end
```

```
>>iftest2
Enter value for x: 3
The squareroot is 3.0000
>>iftest2
Enter value for x: -2
x is negative: there is no real result
>>
```
Use an if structure to calculate a natural log

• Check to see if the input is positive
  • If it is, calculate the natural log
  • If it isn’t, send an error message to the screen
```
x = input('To calculate a natural log, enter a value for x: ')
if x > 0
    y = log(x)
else
    disp('The input to the log function must be positive')
end
```
Interactions in the Command Window

To calculate a natural log, enter a value for x: 8

To calculate a natural log, enter a value for x: -10

To calculate a natural log, enter a value for x: 0

\[ x = 0 \]

The input to the log function must be positive

>>
The if/else/elseif structure

• Use the elseif for multiple selection criteria
• For example
  • Write a program to determine if an applicant is eligible to drive
Structured Programming

Format of **if elseif else** statement:

```plaintext
if Logical Expression
   Statements 1
elseif Logical Expression
   Statements 2
....
else
   Statements 3
end
```
Start

if age<16

True

Sorry – You’ll have to wait

elseif

age<18

True

You may have a youth license

elseif

age<70

True

You may have a standard license

else

Drivers over 70 require a special license

End
disp('Are you eligible to drive?')
age = input('Enter your age: ')
if age<16
    disp('Sorry - You''ll have to wait')
elseif age<18
    disp('You may have a youth license')
elseif age<70
    disp('You may have a standard license')
else
    disp('Drivers over 70 require a special license')
end
Always test your programs – making sure that you’ve covered all the possible calculational paths.
% Program to test the if statement #3
X=input('Enter value for x:');
if X>0
    disp('x is positive');
elseif X<0
    disp('x is negative');
else
    disp('x equal 0');
end

>>iftest3
Enter value for x: 3
x is positive
>>iftest3
Enter value for x: -2
x is negative
Problem:
• Pick a random number $N$ such that $-2 < N < 2$

Calculate $B = \sqrt{\log(N)}$

• If $N$ is positive, calculate $A = \log(N)$
• If $A$ is positive, calculate $B = \sqrt{A}$

% nested "if statements" example

```matlab
N = rand(1)*4-2;
if N>=0
    A = log(N);
    if A>0
        B = sqrt(A);
    end
end
```

Use indentation (Tab key)
As a general rule…

- **If structures** work well for scalars
- **For vectors or arrays use a find function or..**
- Combine if structures with a repetition structure
- Repetition structures are introduced in the next chapter
switch/case

• This structure is an alternative to the if/else/elseif structure
• The code is generally easier to read
• This structure allows you to choose between multiple outcomes, based on some criterion, which must be exactly true
When to use switch/case

- The criterion can be either a scalar (a number) or a string.
- In practice, it is used more with strings than with numbers.
Structured Programming

The “SWITCH” structure

```
switch variable
  case test1
    Statement 1
  case test2
    Statement 2
  ....
  otherwise
    Statement n
end
```
Suppose you want to determine what the airfare is to one of three cities.

```matlab
city = input('Enter the name of a city : ','s')
switch city
    case 'Boston'
disp('$345')
    case 'Denver'
disp('$150')
    case 'Honolulu'
disp('Stay home and study')
otherwise
disp('Not on file')
end
```
Enter the name of a city: Boston

city = Boston

Enter the name of a city: Denver

city = Denver

Enter the name of a city: Honolulu

city = Honolulu

Enter the name of a city: Washington

city = Washington

Not on file

>>
Remember... You tell the **input** command to expect a string by adding ‘s’ in the second field.
% program to test switch

A=input('Your choice [1,2 3] ? ');

switch A
    case 1
        disp('Choice 1')
    case 2
        disp('Choice 2')
    case 3
        disp('Choice 3')
    otherwise
        disp('Wrong choice')
end

>> Testswitch
Your choice [1,2 3] ? 1
Choice 1

>> Testswitch
Your choice [1,2 3] ? 2
Choice 2

>> Testswitch
Your choice [1,2 3] ? 3
Choice 3

>> Testswitch
Your choice [1,2 3] ? 7
Wrong choice
The menu function is often used in conjunction with a switch/case structure.

This function causes a menu box to appear on the screen with a series of buttons defined by the programmer.

```python
input = menu('Message to the user', 'text for button 1', 'text for button 2', etc)
```
• Because the input is controlled by a menu box, the user can’t accidentally enter a bad choice
• This means you don’t need the otherwise portion of the switch/case structure
city = menu('Select a city from the menu: ','Boston','Denver','Honolulu')

switch city
    case 'Boston'
        disp('$345')
    case 'Denver'
        disp('$150')
    case 'Honolulu'
        disp('Stay home and study')
end

Note that the otherwise portion of the switch/case structure wasn’t used
When you run this code a menu box appears.

Instead of entering your choice from the command window, you select one of the buttons from the menu.
If I select Honolulu...

city = 
  3
Stay home and study
>>
Example 1

Write a script `example.m` to find roots of a second order equation

\[ ax^2 + bx + c = 0. \]

When the script is executed it will

- ask the user enter the coefficients \( a, b, c \)
- calculate discriminant
- calculate the roots and display the case according to sign of discriminant.
Example 2

Write a script that allows a user to enter a string containing a day of a week ("Sunday", "Monday" etc) uses a switch construct to convert the day to its corresponding number, where Monday is the first day of the week. Print out the resulting day number. Also be sure to handle the case of an illegal day name.
Summary

- Sections of computer code can be categorized as
  - sequences
  - selection structures
  - repetition structures
Summary – Sequence

• Sequences are lists of instructions that are executed in order
Summary – Selection Structure

• Selection structures allow the programmer to define criteria (conditional statements) which the program uses to choose execution paths
Summary – Repetition Structures

• Repetition structures define loops where a sequence of instructions is repeated until some criterion is met (also defined by conditional statements).
Summary – Relational Operators

- MATLAB uses the standard mathematical relational operators:
  - <
  - <=
  - >
  - >=
  - ==
  - ~=

Recall that = is the assignment operator, and cannot be used for comparisons.
Summary – Logical Operators

• MATLAB uses the standard logical operators
  • && and
  • || or
  • ~ not
  • xor exclusive or
Summary – Logical Functions

- The **find** command is unique to MATLAB, and should be the primary logical function used in your programming.
- It allows the user to specify a condition using both logical and relational operators, which is then used to identify elements of a matrix that meet the condition.
The family of if structures allows the programmer to identify alternate computing paths dependent upon the results of conditional statements.

- if
- else
- elseif
Summary switch/case

- Similar to the if/elseif/else structure
- Commonly used with menu