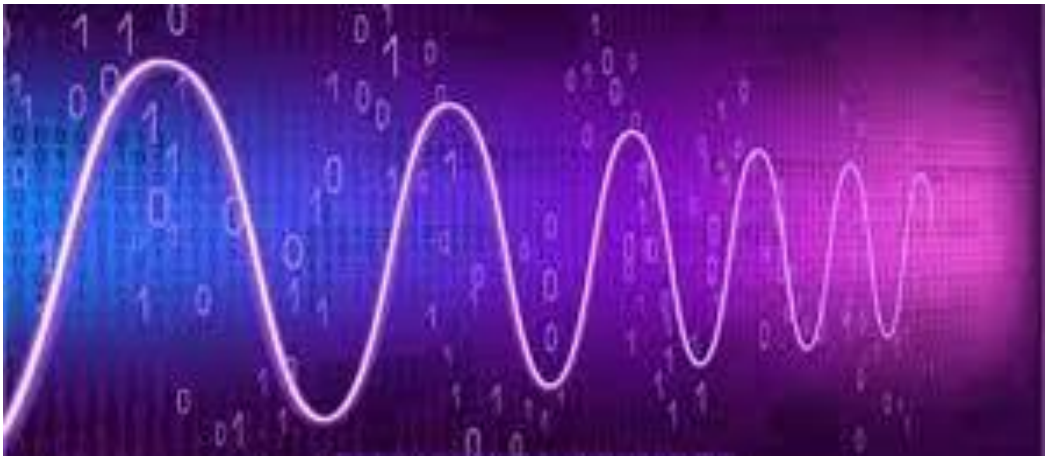
DIGITAL SIGNAL PROCESSING LAB MANUAL 7

Dr. Ahsan Latif, Ms. Anosh Fatima



WINTER SEMESTER 2022

UNIVERSITY OF AGRICULTURE, FAISALABAD (UAF)

MATLAB Fundamentals

TASK

- Open New Script from Toolbar
- Write code (all examples and Exercises) in Editor Window
- Click Run from Tool Bar to see output in command window.
- Save all MATLAB Files in Separate folder called DSP Labs
- Write name of file: your group number and lab number. E.g., G2Lab3
- Click add to the path
- Check results (output)
- Submit MATLAB code to TA online. Solve each exercise in separate script.
- ALL Code must be properly commented for submission, explaining each each step/line of exercises.

Loops and Conditional Statements

Loops and Conditional Statements

Control flow and branching using keywords, such as `if`, `for`, and `while`

MATLAB Language Syntax

`if, elseif, else`	Execute statements if condition is true
`for`	for loop to repeat specified number of times
`parfor`	Parallel for loop
`switch, case, otherwise`	Execute one of several groups of statements
`try, catch`	Execute statements and catch resulting errors
`while`	while loop to repeat when condition is true
`break`	Terminate execution of for or while loop
`continue`	Pass control to next iteration of for or while loop
`end`	Terminate block of code, or indicate last array index
`pause`	Stop MATLAB execution temporarily
`return`	Return control to invoking function

Loops and Conditional Statements

Within any script, you can define sections of code that either repeat in a loop or conditionally execute. Loops use a `for` or `while` keyword, and conditional statements use `if` or `switch`.

Loops are useful for creating sequences. For example, create a script named `fibseq` that uses a `for` loop to calculate the first 100 numbers of the Fibonacci sequence. In this sequence, the first two numbers are 1, and each subsequent number is the sum of the previous two, $F_n = F_{n-1} + F_{n-2}$.

```
N = 100;
f(1) = 1;
f(2) = 1;

for n = 3:N
    f(n) = f(n-1) + f(n-2);
end
f(1:10)
```

When you run the script, the `for` statement defines a counter named `n` that starts at 3. Then, the loop repeatedly assigns to `f(n)`, incrementing `n` on each execution until it reaches 100. The last command in the script, `f(1:10)`, displays the first 10 elements of `f`.

```
ans =
     1     1     2     3     5     8    13    21    34    55
```

Conditional statements execute only when given expressions are true. For example, assign a value to a variable depending on the size of a random number: `'low'`, `'medium'`, or `'high'`. In this case, the random number is an integer between 1 and 100.

```
num = randi(100)
if num < 34
   sz = 'low'
elseif num < 67
   sz = 'medium'
else
   sz = 'high'
end
```

The statement `sz = 'high'` only executes when `num` is greater than or equal to 67.

Exercise 1

1. Calculate and display the first 20 numbers of the fibonacci sequence when first two numbers are 1, using loop.
2. Calculate and display the first 15 numbers of the fibonacci sequence when first two numbers are 1 and 2, using loop.

Loop Control Statements

With loop control statements, you can repeatedly execute a block of code. There are two types of loops:

- `for` statements loop a specific number of times, and keep track of each iteration with an incrementing index variable.

  For example, preallocate a 10-element vector, and calculate five values:

  ```
  x = ones(1,10);
  for n = 2:6
      x(n) = 2 * x(n - 1);
  end
  ```

- `while` statements loop as long as a condition remains true.

  For example, find the first integer `n` for which `factorial(n)` is a 100-digit number:

  ```
  n = 1;
  nFactorial = 1;
  while nFactorial < 1e100
      n = n + 1;
      nFactorial = nFactorial * n;
  end
  ```

Each loop requires the `end` keyword.

It is a good idea to indent the loops for readability, especially when they are nested (that is, when one loop contains another loop):

```
A = zeros(5,100);
for m = 1:5
    for n = 1:100
        A(m, n) = 1/(m + n - 1);
    end
end
```

You can programmatically exit a loop using a `break` statement, or skip to the next iteration of a loop using a `continue` statement. For example, count the number of lines in the help for the `magic` function (that is, all comment lines until a blank line):

```
fid = fopen('magic.m','r');
count = 0;
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line)
       break
    elseif ~strncmp(line,'%',1)
       continue
    end
    count = count + 1;
end
fprintf('%d lines in MAGIC help\n',count);
fclose(fid);
```

Conditional Statements

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an `if` statement. For example:

```
% Generate a random number
a = randi(100, 1);

% If it is even, divide by 2
if rem(a, 2) == 0
    disp('a is even')
    b = a/2;
end
```

`if` statements can include alternate choices, using the optional keywords `elseif` or `else`. For example:

```
a = randi(100, 1);

if a < 30
    disp('small')
elseif a < 80
    disp('medium')
else
    disp('large')
end
```

Alternatively, when you want to test for equality against a set of known values, use a `switch` statement. For example:

```
[dayNum, dayString] = weekday(date, 'long', 'en_US');

switch dayString
   case 'Monday'
      disp('Start of the work week')
   case 'Tuesday'
      disp('Day 2')
   case 'Wednesday'
      disp('Day 3')
   case 'Thursday'
      disp('Day 4')
   case 'Friday'
      disp('Last day of the work week')
   otherwise
      disp('Weekend!')
end
```

For both `if` and `switch`, MATLAB® executes the code corresponding to the first true condition, and then exits the code block. Each conditional statement requires the `end` keyword.

In general, when you have many possible discrete, known values, `switch` statements are easier to read than `if` statements. However, you cannot test for inequality between `switch` and `case` values. For example, you cannot implement this type of condition with a `switch`:

```
yourNumber = input('Enter a number: ');

if yourNumber < 0
    disp('Negative')
elseif yourNumber > 0
    disp('Positive')
else
    disp('Zero')
end
```

Exercise 2

Write a MATLAB code to display first 15 natural numbers using loop.

1. Write a MATLAB code to calculate and display sum of first 10 natural numbers using loop.
2. Write a MATLAB code to calculate and display average of first 20 natural numbers using loop.
3. Write a MATLAB code to calculate and display square of first 10 natural numbers using loop.
4. Display the first 10 even numbers, using loop.
5. Display the first 20 odd numbers, using loop.
6. Write a MATLAB code to calculate and display sum of first 30 odd numbers using loop.
7. Write a MATLAB code to calculate and display sum of first 20 even numbers using loop.

Exercise 3

Write a MATLAB code for following programs

1. Take random number (out of 100) in variable ‘marks’ using ‘randi’ function and assign grade accordingly.
   Grade A when marks are equal and above 85
   Grade B when marks are between 70 and 84

Grade C when marks are between 55 and 69
Grade D when marks are between 45 and 54
Grade F when marks are below 45.

2. Display pattern like right angle triangle using asterik and loop.

*
**
***
****

3. Take random number (out of 20) in variable ‘num’ using ‘randi’ function and tell whether it is even or odd.
4. Take random number (out of 50) in variable ‘num’ using ‘randi’ function and tell whether it is prime number or not.
5. Write first 10 natural numbers in reverse order using loop.
6. Take 2 random numbers (out of 100) in variables ‘num1’ and ‘num2’ using ‘randi’ function and tell which one is maximum among both.
7. Take 2 random numbers (out of 100) in variables ‘num1’ and ‘num2’ using ‘randi’ function and tell which one is minimum among both.
8. Take 3 random numbers (out of 100) using ‘randi’ function and tell which one is maximum among them.
9. Take random numbers (out of 100) in variable ‘num’ using ‘randi’ function and tell if its divisible by 5 or not.
10. Take random numbers (out of 100) in variable ‘num’ using ‘randi’ function and tell if its divisible by 3 or not.
11. Take random numbers (out of 2022) in variable ‘year’ using ‘randi’ function and tell if its leap year or not.
12. Take random numbers (out of 12) in variable ‘month’ using ‘randi’ function and tell its number of days.
13. Take random numbers (out of 8) in variable ‘sides’ using ‘randi’ function and tell its shape.
    Zero sides means no shape
    One side means a line
    2 side means no shape
    3 sides means triangle
    4 sides means rectangle and so on.