

Worksheet 5

SUMMARY Using MATLAB for Fun and Profit
CURRENT READING Recktenwald (Chapter 2), pp. 15-84; Mathews & Fink, pp. 638-645; Moler (Chapter 1)

WARM UP

Write down (in your own words) the meaning of the following terms:
ALGORITHM :

PSEUDOCODE :

Introduction to MATLAB

MATLAB is an interactive numerical computing environment. It allows both command-line instructions, and programs, which are placed in files ending with `.m`.

Our goal is to take the next few classes to become introduced to, and proficient with, using MATLAB interactively.
We will be using files from the `nmm` toolbox, which should be found in `S:\Math Courses\Math370\Spring2009` directory. Of particular interest to us will be the `data`, `interact` and `program` directories. The programs used with the Mathews & Fink text are in the `mathews` directory.

MATLAB **Help**

You can use the command `help` *command* to get information on the command *command*. MATLAB is **not case-sensitive**.

You can use the command `lookfor` *string* to search the list of MATLAB commands for occurrences of the word *string*.

Scripts

Scripts are just files which contain sequences of interactive MATLAB commands. Scripts do not have input or output parameters. Variables used in scripts affect the variables in the MATLAB variable space.

Functions

Functions are MATLAB subprograms similar to subroutines found in programming languages `C` or `Fortran`. Functions can use both global variables and local variables. Functions can have multiple inputs and outputs.

Functions have features scripts do not have. Scripts have no advantages over functions. Use functions, not scripts!

EXAMPLE

Look at the files `tanplot.m`, `threesum.m`, `addmult.m` and `twosum.m`. Which of these are **script** files and which of these are **function** m-files?

tanplot.m

```
theta = linspace(1.6,4.6);
tandata = tan(theta);
plot(theta,tandata);
xlabel('\theta     (radians)')
ylabel('tan(\theta)');
grid on;
axis([min(theta) max(theta) -5 5]);
```

Check One: ☐ **SCRIPT** or ☐ **FUNCTION**
SAY WHAT THIS PROGRAM DOES:

twosum.m

```
function twosum(x,y)
% twosum  Add two matrices and print the result
x+y
```

Check One: ☐ **SCRIPT** or ☐ **FUNCTION**
SAY WHAT THIS PROGRAM DOES:

threesum.m

```
function s = threesum(x,y,z)
% threesum  Add three variable and returns the result
s = x+y+z;
```

Check One: ☐ **SCRIPT** or ☐ **FUNCTION**
SAY WHAT THIS PROGRAM DOES:

addmult.m

```
function [s,p] = addmult(x,y)
% addmult  Compute sum and product of two matrices
s = x+y;
p = x*y;
```

Check One: ☐ **SCRIPT** or ☐ **FUNCTION**
SAY WHAT THIS PROGRAM DOES:

easyplot.m

```
D = load('xy.dat');
x = D(:,1); y =D(:,2);
plot(x,y)
xlabel('x axis')
ylabel('y axis')
title('Plot of generic x-y data')
```

Check One: ☐ **SCRIPT** or ☐ **FUNCTION**
SAY WHAT THIS PROGRAM DOES:

Let's run each one and also look at them and insure that we understand what each one does. **NOTE: you must have the directory in which the m-files appear in your path in order to run them.**

The **Machine Precision** is the number ϵ_m which makes the following statement on a computer to be TRUE:

$$1 + \epsilon_m = 1$$

Consider the following algorithm to compute ϵ_m, the machine precision:

```
LET epsilon = 1
LET COUNTER = 0
LET MAXCOUNTER = 100
WHILE COUNTER < MAXCOUNTER
   LET B = 1 + EPSILON
   IF (B EQUALS 1) QUIT PROGRAM
   LET EPSILON = EPSILON/2  % halve epsilon each iteration
   LET COUNTER = COUNTER + 1 % update thecounter
END WHILE
OUTPUT (EPSILON, COUNTER)
```

Exercise

Find the machine precision of your calculator.

Implications

When designing an algorithm one should NOT USE the logical construct **Are** x **and** y **equal?** but instead **Are** x **and** y **close?** or **Is** $x - y$ **small enough**?

Here is how the "MACHINE PRECISION" ALGORITHM would be implemented in MATLAB (type in your own `myeps.m` script and see what happens)

```
epsilon = 1;
it = 0;
maxit = 100;
while it < maxit,
   b = 1 + epsilon;
   if b == 1 break; end
   epsilon = epsilon/2;
   it = it + 1;
end
fprintf('epsilon = %12.8e in %d steps',epsilon,it);
```

NOTE the machine precision ϵ_m for MATLAB is found in the built-in function `eps`.

Example

From the result above, how many bits is MATLAB using to store floating point numbers?

Therefor how many bits is **your calculator** using to store floating point numbers?

GROUPWORK

What would the output of the following MATLAB code be?
(Example found on page 211 of Recktenwald)

```
x = tan(pi/6);
y = sin(pi/6)/cos(pi/6);
if x==y
   fprintf('x and y are equal\n');
else
   fprintf('x and y are not equal: x - y = %e\n\n',x-y);
end
```

Cool Graphics

This is a figure containing 4 subplots which show different surface plot types of $z = 2 - x^2 - y^2$ on the domain $-5 \leq x \leq 5, -5 \leq y \leq 5$ on this page.
The commands are:

```
>> x=linspace(-5,5,20);
[X,Y] = meshgrid(x,x);
>> Z = 2 - X.^2 + Y.^2;
>> subplot(2,2,1); mesh(x,x,Z); title('mesh plot');
>> subplot(2,2,2); surf(x,x,Z); title('surf plot');
>> subplot(2,2,3); surfc(x,x,Z); title('surfc plot');
>> subplot(2,2,4); surfl(x,x,Z); title('surfl plot');
>>
```