Matlab/Programing Tutorial - Part 2

Basics of Programing - General Strategy: ~ 5 steps

1. Clearly define what is to be programmed
   - * a. Background theory, concepts, describing equations
   - b. Define what program is supposed to do
   - c. Who will use the program?
     - i. Once for you ==> quickly hack out code, few comments. - If very short program.
     - ii. Repeatedly for you ==> structured, comments, crude I/O. - Don't need user-friendly.
     - iii. Repeatedly by others ==> GUI, applications notes.
2. Have a calculated example or case for debugging, verification.

3. Define data structure and variables
   a. List of variables and what they are
   b. Data structure - variables, arrays, data types, structures
   c. Identify knowns and unknowns.
   d. Identify whether real, integer, complex, binary, string, etc.
   e. Identify precision of data - single, double, etc.

   - 4-byte, 8-byte.
4. Go through the calculation or algorithm. Interview yourself. What sequence of calcs do you go through? Make some notes as if you were outlining a design guide. This is more or less the sequence of calculations you'll need to code.
   a. Do hand calcs, and/or
   b. Make a flow chart, and/or
   c. Write some psuedocode.
   d. Define structure of code - main program, subroutines, functions, objects, I/O, data files to read or write, etc.

- Hard-code. eg. $a = 7.5;$
- Input: - Keybd.
  - file.

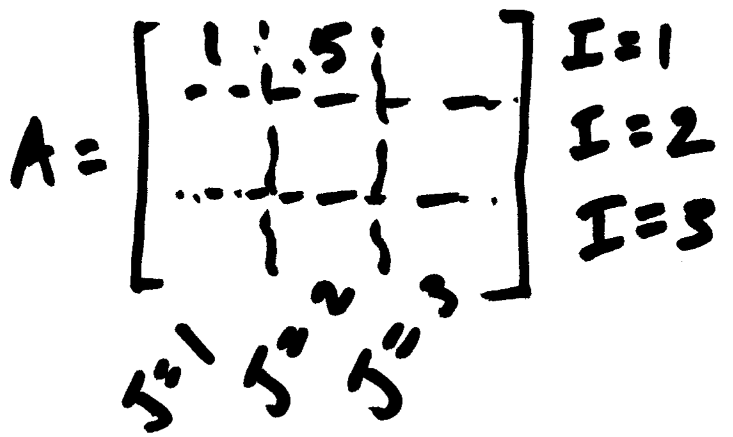
5. Now you can finally start "programing"
   a. Structure of source code file:
      i. Comment block at beginning, purpose, author, revisions, rev. nos/dates.
      ii. Comments referring to appropriate references, design guides, etc. Helpful - itemize req'd functions needed.
      iii. Define variables, data structure, I/O. (Comments)
      iv. Blocks of code, comments with each one.
      v. End of program. simple complex.
   b. Debugging - can vary from crude to elegant.
      i. Simple crude can be the best for simple programs:
         (1) Leave ; off end of line - will display result.
         (2) Run program, hold cursor over variable name whose value you wish to see.
         → (3) Single-stepping - insert pause. (also called break)
      ii. Use higher-level debugging tools.

Programing Basics, Cont'd

- for
- if
- while

For loops. Pasting example from Matlab help utility, for_ex.m

```
N=3;
FOR I = 1:N
    FOR J = 1:N
    A(I,J) = 1/(I+J-1);
    END
END
```



Main points:

- Can be "nested" – Always cycles thru inner loop first.
- Index variables I and J, typically integers (array positions).
- Start:stop range is 1:N, i.e. steps through 1,2,3,4, ... N.
- Default increment is +1, can define otherwise if desired, eg. <u>FOR J = 10:2:-1</u> will step thru 10, 9, 8, 7, ... 2.
- Can also use real number as range and increment.

IF constructs. Pasting an example from Help: "IF-THEN-ELSE"

```
IF expression1
      statement(s) to execute if expression1 is true.
ELSEIF expression2
      statement(s) to execute if expression2 is true.
ELSEIF expression3
      statement(s) to execute if expression3 is true.  } opt
.
.
.
ELSE
      statements to execute if none of above are true.                                       } opt
      (Used to handle exceptions, or sometimes "trap" logic errors, incompleteness)
END
```

Important! – Always executes <u>ONLY</u> the block of statements associated with first true condition found, then skips to END. I.e. if there are multiple true conditions, it still only executes first true one found.

- Elseif and Else are optional, only use if needed.
- If statements can be nested.
- Logic conditions are: ==, <, >, <=, >=, or $\underbrace{\sim=}$

$\neq$

Example (see file for_if_ex.m):

```
if I == J
      A(I,J) = 2;
elseif abs(I-J) == 1
      A(I,J) = -1;
else
      A(I,J) = 0;
end
```

WHILE - Repeat statements until a condition is true. Excellent way to implement a simple iteration. The general form of a WHILE statement is:

```
 WHILE expression
 statements
END
```

Example:

```
while norm(E+F-E,1) > 0
      E = E + F;
      F = A*F/N;
      N = N + 1;
end
```

Note: If-then-Break can be used to jump out of *for* or *while*.
Example of while usage (while_ex.m):

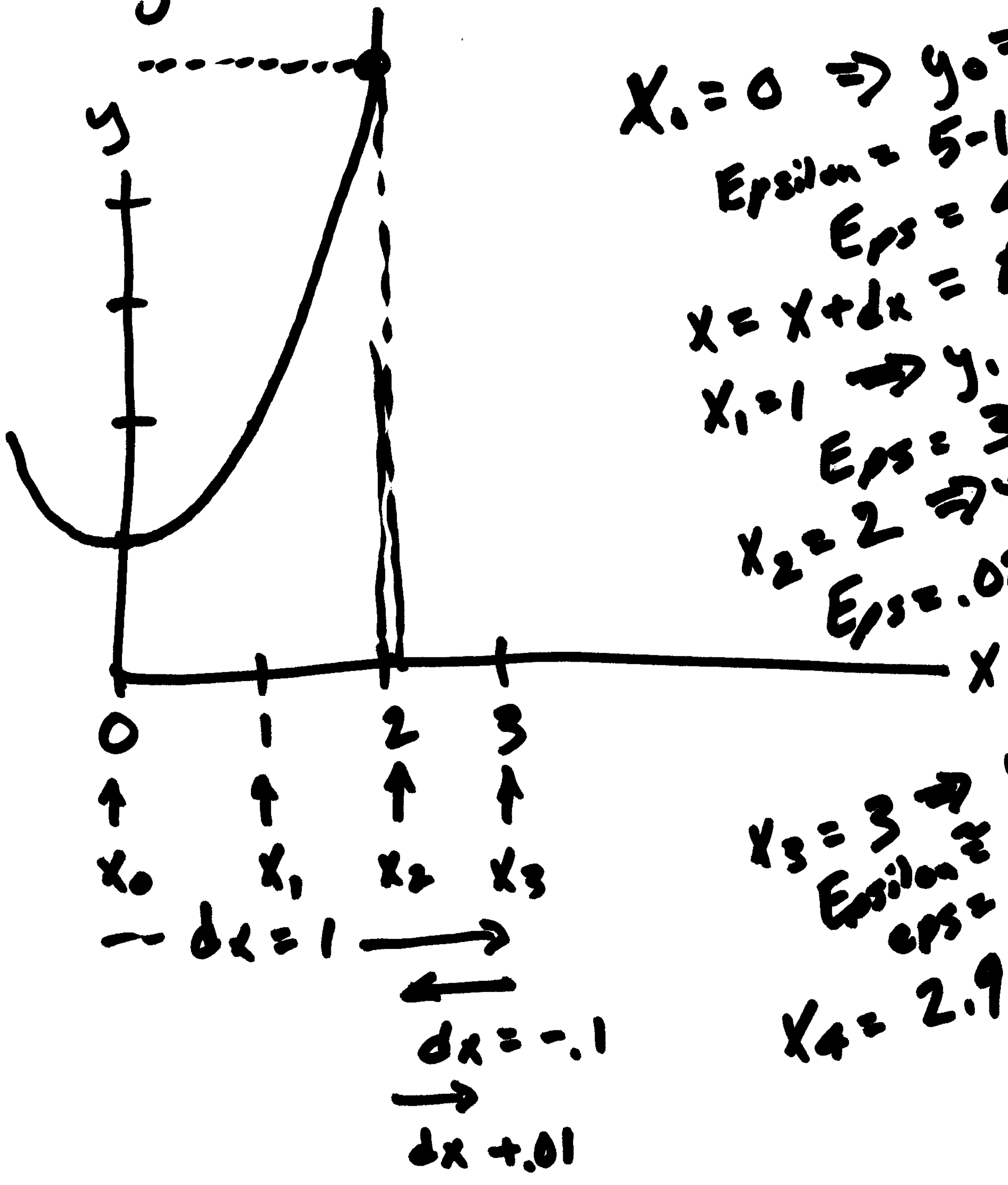
```
% example of while usage.  Search/iterate for value of x where y = 5.
% y = x^2 + 0.974382                y = X^2+1

clear;         %always clear workspace/memory before (re)running program.
x = 0;         %initialize x (first guess)
ysoln = 5.0;   % desired value of y.
epsilon = 20;  % epsilon is error, initially make big pos number.
eps = 20;      % abs value of error (conv. test).
dx = 1.0;      % increment to change x at each iteration.

while eps > 0.0000001      1x10^-7
   y = x^2 + 0.974382;
   eps_prev = epsilon; %save previous value of epsilon
   epsilon = 5-y;
   eps = abs(epsilon);
   if sign(epsilon) ~= sign(eps_prev)
      dx = dx/(-10);
   end
   x = x + dx;
end
x
y
```

$y = 5$
$x = \pm 2$

$y = x^2 + 1$



$X_0 = 0 \Rightarrow y_0 = 1$
Epsilon = 5-1 = 4
Eps = 4
$X = X + dx = 1$
$X_1 = 1 \Rightarrow y_1 = 2$
Eps = 3
$X_2 = 2 \Rightarrow y_2 = 5$
Eps = .026

$X_3 = 3 \Rightarrow y = 10$
Epsilon = -5
eps = 5
$X_4 = 2.9$