

MATLAB Marina – Debugging

Learning Objectives

1. Be able to identify the type of programming error from MATLAB messages.

Terms

debug, syntax error, run time error, logic error

MATLAB Functions, Keywords, and Operators

None

Debugging

When a program or function doesn't operate correctly, what do you do? Debugging a program involves systematically verifying correct/incorrect operation and correcting incorrect portions of the program. This typically involves running the program for various test cases to identify potential errors and correcting each error as they are found. Always correct errors that appear earlier in the program first as correcting these may eliminate errors that occur later.

Programs and functions can be debugged from the MATLAB editor or the command window. Test programs/functions for straightforward cases first then move to more involved cases. If the program has error handling, test the cases that might cause errors after all the regular cases are verified.

Programming Errors

There are three types of errors that can occur in programs: syntax errors, run-time errors, and logic errors.

Syntax errors are errors in the programming languages grammar (syntax) rules. Syntax errors are detected at compile time. MATLAB syntax errors are detected when the program is run and each line is compiled before being executed. Examples of syntax errors in MATLAB are misspelling a keyword such as for and not terminating an if statement with a corresponding end statement. MATLAB will display an error message including the line number where the error occurred in the Command Window for syntax errors. Figure 1a shows a program with a syntax error and Figure 1b shows the error message that MATLAB displays when the program is run.

Syntax errors can be corrected by finding the line number the error occurred at and correcting the syntax. Sometimes the cause of syntax errors is earlier than the indicated line. For example if a variable is used in a formula that was not previously defined, MATLAB will indicate the error occurs at the line number of the formula and the correction is assigning a value to that variable earlier in the program.

```
angle = pi/7;  
result = cos(angle);  
disp(result);
```

Figure 1a. MATLAB Program with Syntax Error

```
Error using pie (line 30)
Not enough input arguments.

Error in Program1 (line 1)
angle = pie/7;
```

Figure 1b. MATLAB Error Message for Syntax Error in Program of Figure 1a

The syntax error on line 5 in the program of Figure 1a is due to a misspelling of the built in constant. The error message “Not enough input arguments” is misleading as it is an error message that occurs when a function is not invoked correctly. MATLAB has a built in pie chart function named `pie`, and this is the error that MATLAB identified rather than a misspelling of `pi`. The line number indicated by the error message is good place to look for the syntax error but the error message may not be appropriate for what the programmer intended.

Run-time errors are errors that occur while the program is executing and occur due to a program action that is not allowed. Runtime errors are detected only when the particular line where the error occurs is executed and the program ceases operation (crashes). Examples of MATLAB run-time errors are formulas using an undefined variable and invoking a function with the incorrect name or arguments. MATLAB will terminate the program execution and display an error message including the line number where the error occurred in the Command Window for run-time errors. Figure 2a shows a program with a run-time error and Figure 2b shows the error message that MATLAB displays when the program is run.

```
num = 5;
result = num/den;
disp(result);
```

Figure 2a. MATLAB Program with Run-time Error

Run-time errors can be corrected by finding the line number the error occurred at and correcting the condition that caused the action that is not allowed. This often requires the use of a debugger so one can see the specific action and values that caused the error. The correction may require modifying the program earlier than where the error occurred. The run-time error on line 2 of the program of Figure 2a is due to the variable `den` not being defined before it is used. In this example, the MATLAB error message is meaningful and indicates the true error in the program.

```
Undefined function or variable 'den'.

Error in Program2 (line 2)
result = num/den;
```

Figure 2b. MATLAB Error Message for Run-time Error in Program of Figure 2a

Logic errors are errors in the program that cause the program to operate incorrectly. The program executes but produces an incorrect or unintended result. The program does not operate the way it is supposed to. Logic errors are detected by the programmer; no error messages are generated by MATLAB. Examples of logic errors are using an incorrect formula for a calculation, an incorrect logic expression in a conditional structure, an infinite while loop due to an incorrect sentinel value, and an incorrect algorithm. Figure 3a shows a program with a logic error and Figure 3b shows the resulting incorrect program output.

```
angle = 45;  
result = cos(angle);  
disp(result);
```

Figure 3a. MATLAB Program with Logic Error

```
0.5253
```

Figure 3b. Output of Program with Logic Error of Figure 3a

Logic errors are typically the most difficult errors to detect and correct. Since no error messages are generated, one does not have the line number where the error occurred. One must first identify the source of the error and then correct the problem. The logic error in the program of Figure 3a is due to the variable `angle` being assigned an angle in degrees rather than radians (or using the `cos` instead of `cosd` function). The only way to recognize this is to notice that the result 0.5253 is not the cosine of 45 degrees which should be 0.7071.

Techniques for identifying the source of logic errors include:

- Tracing the program by hand (or using a calculator) line by line to determine the result of each line of the program and whether or not it is correct.
- Execute each line one by one in the Command Window to determine the result of each line of the program and whether or not it is correct.
- Adding extra display statements to print the values of important variables before and after operations and in critical portions of control statements (if, for, while). Use the printed values to determine if the values are what they should be after each operation. Remember to remove the extra display statements (or comment them out) from the final version of the program.
- Using an interactive debugger and set breakpoints to monitor the values of variables in the workspace while the program runs. Most modern IDEs including MATLAB have built in interactive debuggers.

Last modified January 24, 2022

 [MATLAB Marina](#) is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.