

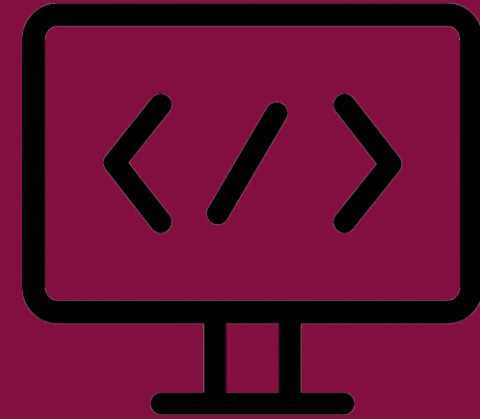
SEEE1022 INTRODUCTION TO SCIENTIFIC PROGRAMMING

CH8 Debugging

Dr. Mohd Saiful Azimi Mahmud (azimi@utm.my)
P19a-04-03-30, School of Electrical Engineering, UTM



UTM
UNIVERSITI TEKNOLOGI MALAYSIA



www.utm.my

innovative • entrepreneurial • global



univteknologimalaysia



utm_my



utmofficial

- Understand syntax error and run-time error.
- Know how to perform debugging.

- A software bug (error) is a problem that exists in the code you have written. The process of fixing the bugs is called debugging.
- Below are the types of errors:
 1. Syntax Error – typing error
 2. Run-time Error – Occurs when the program is running. Possible caused are:
 - i. Incompatible array size.
 - ii. Invalid operation – undefined variable, invalid array indexing, invalid input/output arguments.
 - iii. Logic Error.
 - iv. Rounding Error.
- MATLAB includes a number of tools to help you debug your code.

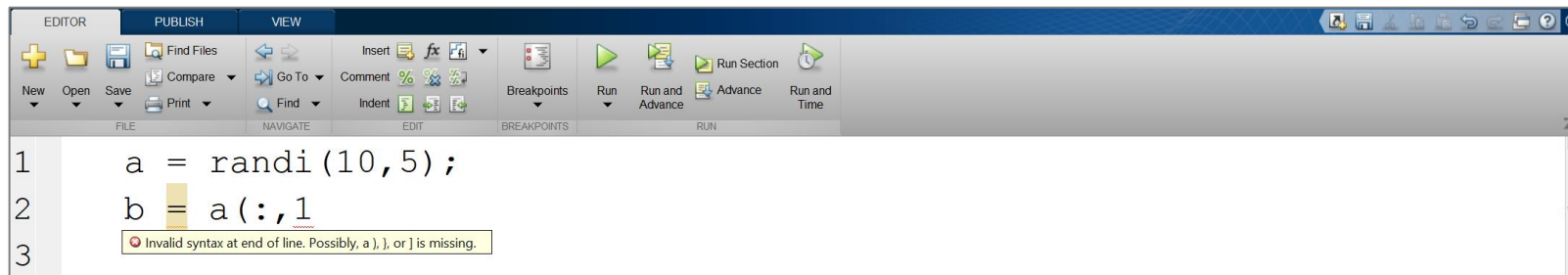
SYNTAX ERROR

INTRODUCTION

- Syntax error is a typing error according to the given syntax.
- Easily detected by MATLAB and directly shown in the editor while you typing the program with the followings:
 1. Error bar
 2. Error message
- The program will not be executed until the syntax error is cleared.
- If you run a program consists of syntax error, an error message will be returned at the command window prompt without executing the program.

ERROR BAR AND MESSAGE

- Whenever you use an M-file, notice that along the right-hand side of the figure window a vertical bar appears.
- That marks locations where there are syntax errors or where MATLAB® has issued warnings.



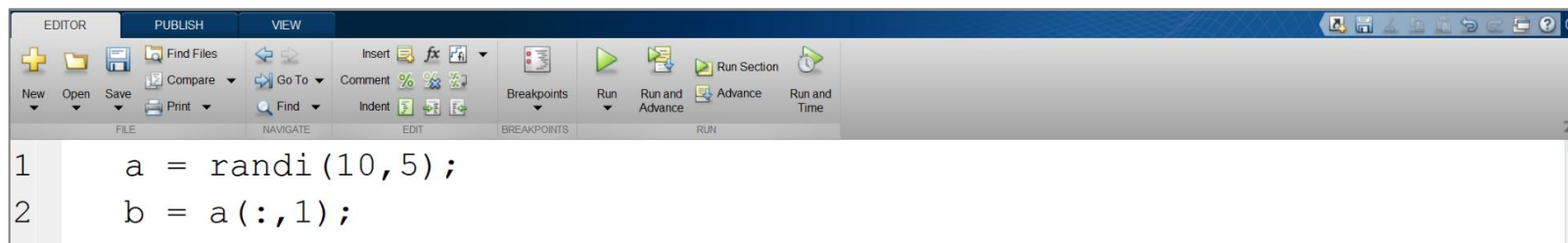
```

1 a = randi(10,5);
2 b = a(:,1
3

```

Invalid syntax at end of line. Possibly, a), or] is missing.

- The color turn green when the syntax error has been resolved.



```

1 a = randi(10,5);
2 b = a(:,1);

```

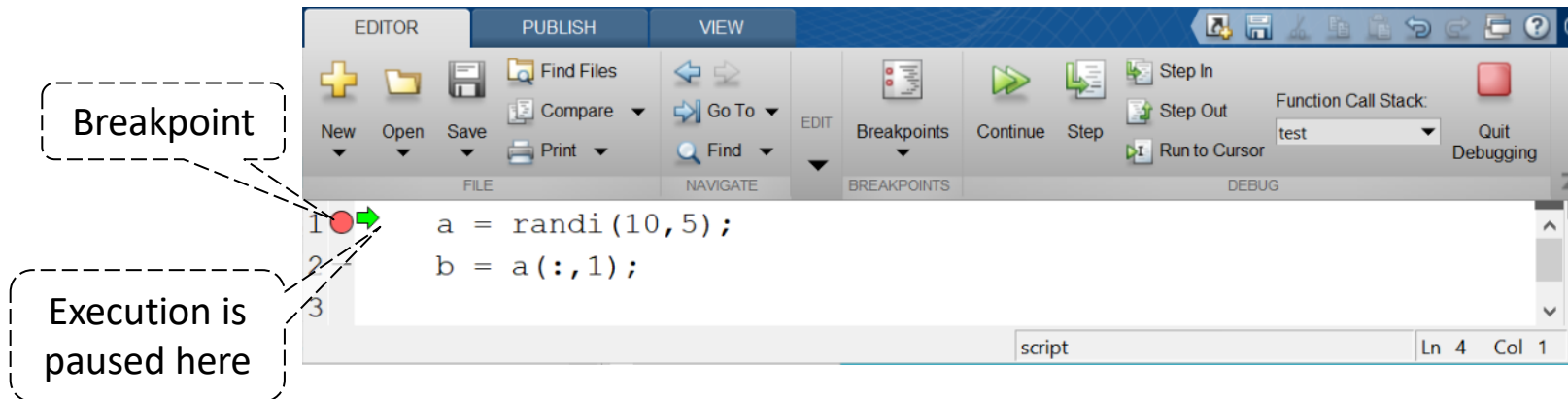
RUN-TIME ERROR

INTRODUCTION

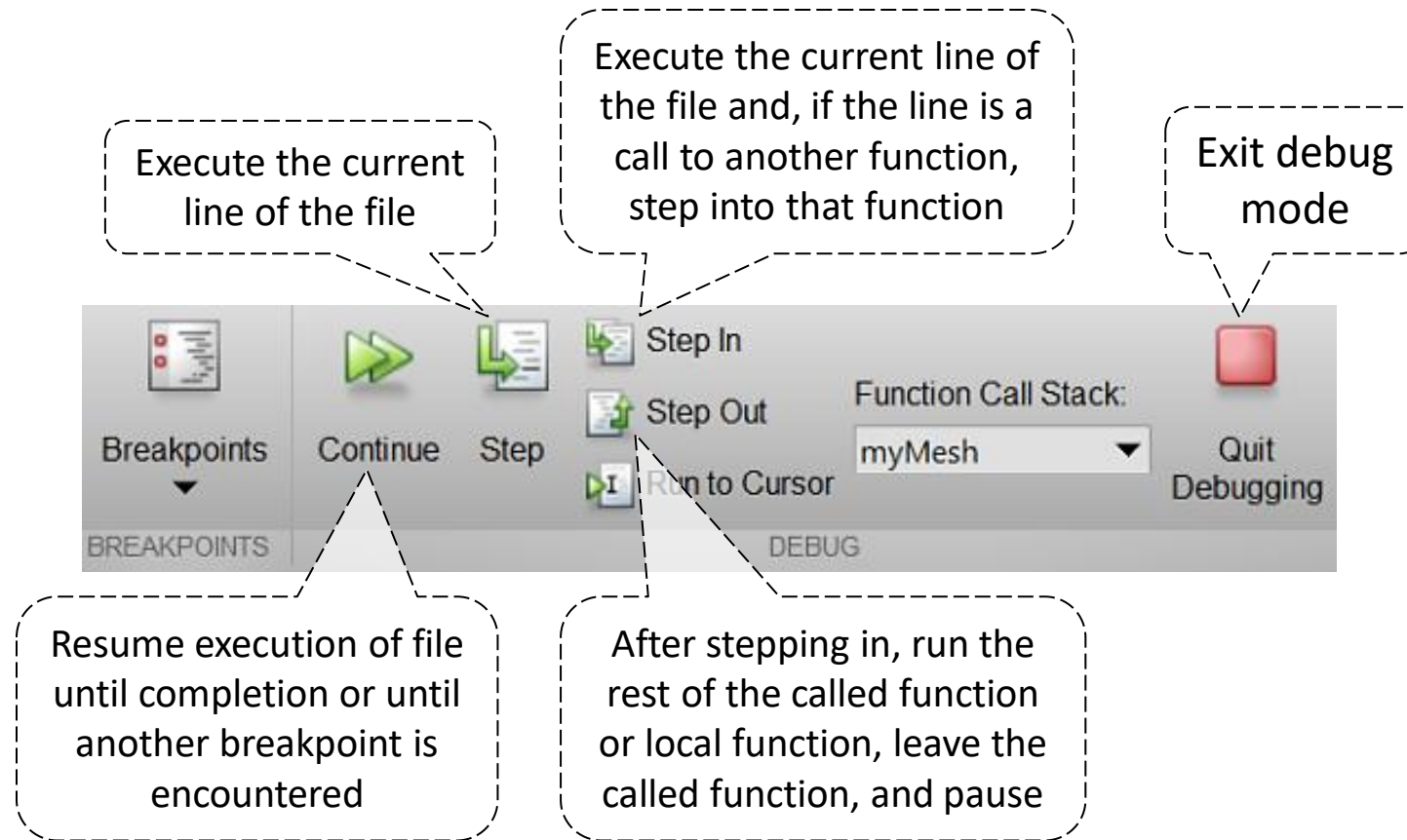
- Run-time error is harder to fix since the error will be detected by MATLAB only when the program is run or it will not be detected at all by MATLAB.
- Error that can be detected by MATLAB:
 - Incompatible array size
 - Invalid operation
- Error not detected by MATLAB
 - Logic error
 - Rounding error
- To debug this type of errors, MATLAB has a graphical debugger toolbar to work with breakpoints.

BREAKPOINT

- When trying to find and fix run-time errors in a program code, it is often useful to run sections of the program, then pause, evaluate what has happened, and continue.
- Breakpoints is used to pause the execution while you evaluate results.
- Breakpoint is set by clicking cursor at the right side of the code-line number.
- Breakpoints can't be enabled until all of the syntax errors have been resolved.



BREAKPOINT



UNDEFINED VARIABLE

EXAMPLE 1

```
data = [3 5 7 5];  
data2 = dataa([3 2 2 3])
```

Undefined function or variable 'dataa'.

Error in **chp8ex8_1** (line 2)
data2 = dataa([3 2 2 3])

This kind of typing error is not syntax error.

INCOMPATIBLE ARRAY SIZE

EXAMPLE 2

```
>> a = [2 4 5];  
>> b = [5 8];  
>> c = a + b;
```

Matrix dimensions must agree.

- Vector a and b must have similar length for plus operation.
- Refer to 'Compatible Array Sizes for Basic Operations' topic in MATALAB documentation for full list of compatible array size.

LOGIC ERROR

- Program will execute completely, but the answer that they return is incorrect.
- No error message is returned by MATLAB.
- Below are the possible reason of the logic error:
 - i. Divide by zero.
 - ii. Array operation mistake.
 - iii. Array indexing mistake.
 - iv. Looping indexing mistake.
 - v. Wrong decision statement condition.
 - vi. Data type mistake.
 - vii. Wrong operation or equation.

LOGIC ERROR

EXAMPLE 3

Code the equation below for $s = 0.1:0.1:0.2$ and $a = 0.1:0.1:0.2$.

$$H = \frac{(s - a)(s - 3)}{(s - 0.2)}$$

```
for s = 0.1:0.1:0.2
    for a = 0.1:0.1:0.2
        H = (s-a)*(s-3)/(s-0.2);
        fprintf('s = %.1f,    a = %.1f,    H = %.2f\n',s,a,H)
    end
end
```

```
s = 0.1,    a = 0.1,    H = 0.00
s = 0.1,    a = 0.2,    H = -2.90
s = 0.2,    a = 0.1,    H = -Inf
s = 0.2,    a = 0.2,    H = NaN
```

$$H = \frac{\cancel{(s - 0.2)}(s - 3)}{\cancel{(s - 0.2)}} = s - 3$$

The last result should return $H = -2.8$, but what is return is NaN.

LOGIC ERROR

EXAMPLE 3

Below is one way of how logic error in Example 3 can be resolved

```
for s = 0.1:0.1:0.2
    for a = 0.1:0.1:0.2
        if s==0.2 && s==a
            H = s-3;
        else
            H = (s-a)*(s-3)/(s-0.2);
        end
        fprintf('s = %.1f,    a = %.1f,    H = %.2f\n',s,a,H)
    end
end
```

```
s = 0.1,    a = 0.1,    H = 0.00
s = 0.1,    a = 0.2,    H = -2.90
s = 0.2,    a = 0.1,    H = -Inf
s = 0.2,    a = 0.2,    H = -2.80
```

LOGIC ERROR

EXAMPLE 4

Write MATLAB code for $Y = X(1 + A)$ where $X = [3 \ 3]$ and $A = \begin{bmatrix} 2 & 5 \\ 4 & 8 \end{bmatrix}$

```
>> X = [3 3];  
>> A = [2 5; 4 8];  
  
>> Y = X*(1+A)  
Y =  
    24    45
```

What do you think? Is there any error with the output? Lets rearrange the equation as $Y=X+XA$ and rewrite the code as below. The new code return different value compared to the above code. Which code do you think has the logic error?

```
>> Y = X + X*A  
Y =  
    21    42
```


ROUNDING ERROR

- Rounding error is an error which results from the finite precision available on the computer, i.e., eight bytes per variable, instead of an infinite number.

EXAMPLE 5

```
x = 0.1; a = 0;
while x <= 0.2
    x = x + 0.01;
    if x == 0.15
        disp('x is now 0.15')
    else
        a = 1;
    end
end
if a==1
    disp('x equals to 0.15 was not found')
end
```

```
x equals to 0.15 was not found
```

After fifth repetition of $x=x+0.01$, MATLAB should display 'x is now 0.15', but this was not happening .

ROUNDING ERROR

- Lets investigate what was actually happened.

```
a = 0.11:0.01:0.2;
x = 0.1;
n = 0;
disp('Plus 0.1      Error')
while x <= 0.2
    x = x + 0.01;
    n = n + 1;
    fprintf('%6.2f      %g\n', x, x-a(n))
end
```

Plus 0.1	Error
0.11	0
0.12	0
0.13	0
0.14	0
0.15	2.77556e-17
0.16	2.77556e-17
0.17	2.77556e-17
0.18	2.77556e-17
0.19	5.55112e-17
0.20	5.55112e-17

We can see a very small different when the operation $x=x+0.01$ reach 0.15. This is due to the rounding error.

FIXING ROUNDING ERROR

EXAMPLE 6

```
b = 1:100;  
x = 0.1;  
for x = 0.1:0.01:0.2  
    c = x + b(x*100);  
end
```

Subscript indices must either be real positive integers or logicals.

Error in **chp8ex8_6** (line 4)
 c = x + b(x*100);

This is also a situation where rounding error normally occur.

- `round()` function can be used to resolve the rounding error.

```
b = 1:100;  
x = 0.1;  
for x = 0.1:0.01:0.2  
    c = x + b(round(x*100));  
end
```

STUDY CASE : LOAD FACTOR

Example 7 to Example 10 will be based on the a program of load factor computation below:

In a power system grid, load factor is use to evaluate the effectiveness of power plants in generating electrical. Thus, an effective power distribution method to the end user can be designed. Below is an example on how a daily load factor is calculated. The no colour region of the table is recorded load usage for 24 hours period. The greyed region is how the load factor calculation is done.

Start Hour	End Hour	Load, MW	Interval, hr	Interval total energy
00:00	06:00	5	6	5x6 = 30
06:00	10:00	8	4	8x4 = 32
10:00	14:00	7	4	7x4 = 28
14:00	20:00	6	6	6x6 = 36
20:00	24:00	6	4	6x4 = 24
Total			24	150
Average Load = Total energy/Total hour				150/24 = 6.25
Load Factor = (Average Load/Peak Load)*100				(6.25/8)*100 = 78.13

LOAD FACTOR : THE PROGRAM

Having the recorded load usage saved in spreadsheet file loadfactor.xlsx, below is the program code computing the load factor. Run the code, identify the errors and fix them.

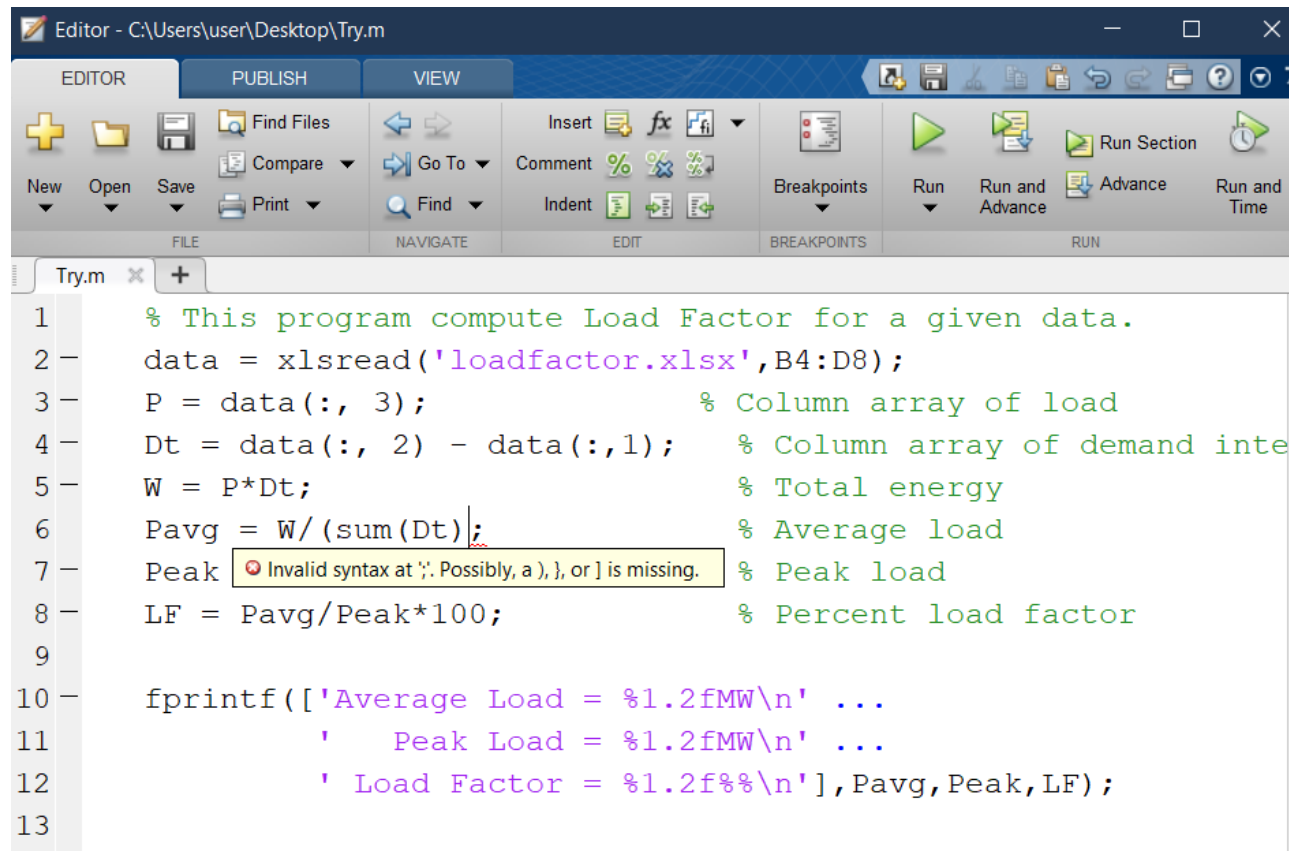
```
% This program compute Load Factor for a given data.
data = xlsread('loadfactor.xlsx',B4:D8);
P = data(:, 3);           % Column array of load
Dt = data(:, 2) - data(:,1); % Column array of demand
interval
W = P*Dt;                % Total energy
Pavg = W/(sum(Dt));      % Average load
Peak = max(P);          % Peak load
LF = Pavg/Peak*100;     % Percent load factor

fprintf(['Average Load = %1.2fMW\n' ...
        '   Peak Load = %1.2fMW\n' ...
        ' Load Factor = %1.2f%%\n'], Pavg, Peak, LF);
```

LOAD FACTOR : SYNTAX ERROR

EXAMPLE 7

From the editor, it shows that there is one syntax error at line 7. It can be resolved by removing the bracket before the sum function.



```

1  % This program compute Load Factor for a given data.
2  data = xlsread('loadfactor.xlsx',B4:D8);
3  P = data(:, 3);           % Column array of load
4  Dt = data(:, 2) - data(:,1); % Column array of demand inte
5  W = P*Dt;                % Total energy
6  Pavg = W/(sum(Dt));      % Average load
7  Peak = Pmax;             % Peak load
8  LF = Pavg/Peak*100;     % Percent load factor
9
10 fprintf(['Average Load = %1.2fMW\n' ...
11         '   Peak Load = %1.2fMW\n' ...
12         '   Load Factor = %1.2f%%\n'], Pavg, Peak, LF);
13
  
```

LOAD FACTOR : UNDEFINED VARIABLE

EXAMPLE 8

With no syntax error left, the code is now can be run. When running the code, MATLAB return the following error:

```
Undefined function or variable 'B4'.  
  
Error in Try (line 2)  
data = xlsread('loadfactor.xlsx',B4:D14);
```

By checking with the MATLAB documentation, the second input argument should be a string. This is where the code is wrong where a variable from expression B4:D14 was given instead of string. The error is resolved as below:

```
data = xlsread('loadfactor.xlsx','B4:D14');
```

LOAD FACTOR : INVALID OPERATION

EXAMPLE 9

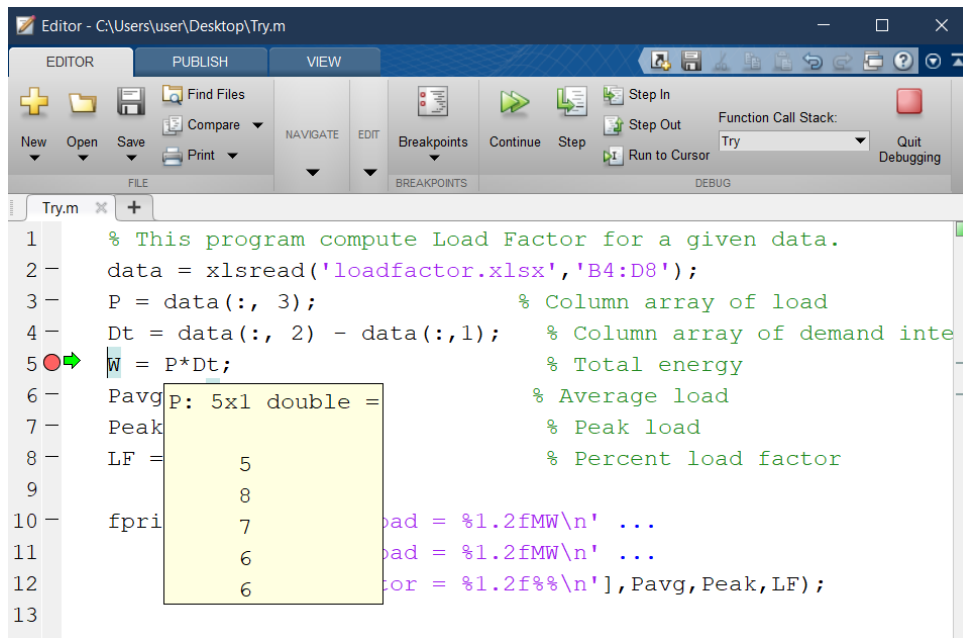
With previous error resolved, run again the program. This time, MATLAB return a new error message as below:

```
Error using *  
Inner matrix dimensions must agree.  
  
Error in Try (line 5)  
W = P*Dt;           % Total energy
```

Since the error message tell us that there is incompatible matrix size, then we need to check both size of variable P and Dt. We can do this by setting a breakpoint at line 5 and check the size while the program is paused by the breakpoint.

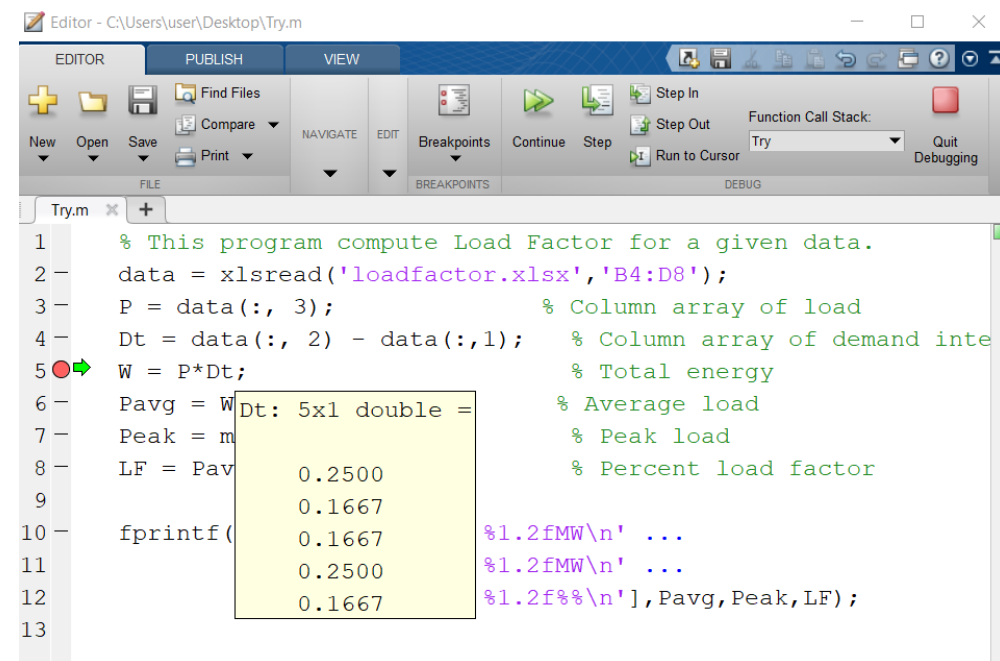
EXAMINE VALUES WHILE DEBUGGING

- With the breakpoint set at line 5, run the program. The program will be paused at line 5. Now, hover the cursor to variable P and Dt to check their values.
- From below figures, we now know that both are column vectors of the same size, which is size incompatible for matrix multiplication process.



```

1 % This program compute Load Factor for a given data.
2 data = xlsread('loadfactor.xlsx','B4:D8');
3 P = data(:, 3); % Column array of load
4 Dt = data(:, 2) - data(:,1); % Column array of demand inte
5 W = P*Dt; % Total energy
6 Pavg = W; % Average load
7 Peak = m; % Peak load
8 LF = 5; % Percent load factor
9
10 fprintf('Load = %1.2fMW\n' ...
11 'oad = %1.2fMW\n' ...
12 'or = %1.2f%%\n'], Pavg, Peak, LF);
13
  
```



```

1 % This program compute Load Factor for a given data.
2 data = xlsread('loadfactor.xlsx','B4:D8');
3 P = data(:, 3); % Column array of load
4 Dt = data(:, 2) - data(:,1); % Column array of demand inte
5 W = P*Dt; % Total energy
6 Pavg = W; % Average load
7 Peak = m; % Peak load
8 LF = Pavg; % Percent load factor
9
10 fprintf('Load = %1.2fMW\n' ...
11 'oad = %1.2fMW\n' ...
12 'or = %1.2f%%\n'], Pavg, Peak, LF);
13
  
```

- One way to get rid of the error is to use array operation instead of the matrix operation.

LOAD FACTOR : LOGIC ERROR

EXAMPLE 10

When array operation (with period symbol ‘.’) is used to replace the matrix operation, below is the result when running the program:

```
Average Load = 1.25MW
  Peak Load = 1.33MW
  Load Factor = 1.17%
Average Load = 1.50MW
  Peak Load = 1.00MW
  Load Factor = 8.00%
Average Load = 15.63MW
  Peak Load = 16.67MW
  Load Factor = 14.58%
Average Load = 18.75MW
  Peak Load = 12.50MW
  Load Factor = >>
```

This time, no error message was returned by MATLAB. However the results are wrong since the expected output supposed to be one value for each Average Load, Peak Load and Load Factor. This is what we called logic error.

After some investigation, the multiplication on line 6 should use matrix operation to get the single output. Thus, to resolved the previous error on the incompatible array size, variable P need to be first transpose before the multiplication.

LOAD FACTOR : RESOLVED CODE

EXAMPLE 10

Below is the final code free of error.

```
% This program compute Load Factor for a given data.
data = xlsread('loadfactor.xlsx','B4:D8');
P = data(:, 3);           % Column array of load
Dt = data(:, 2) - data(:,1); % Column array of demand interval
W = P'*Dt;               % Total energy
Pavg = W/sum(Dt);        % Average load
Peak = max(P);           % Peak load
LF = Pavg/Peak*100;      % Percent load factor

fprintf(['Average Load = %1.2fMW\n' ...
        '   Peak Load = %1.2fMW\n' ...
        ' Load Factor = %1.2f%%\n'], Pavg, Peak, LF);
```

```
Average Load = 6.25MW
   Peak Load = 8.00MW
Load Factor = 78.13%
```



univteknologimalaysia



utm_my



utmofficial

Thank You

www.utm.my

innovative • entrepreneurial • global