

MATLAB 程序调试方法和过程

如何发现 MATLAB 程序中的错误，最简单的莫过于 MATLAB 直接告诉你哪行出错了，但是很多时候情况并不这么简单。比如第 35 行出错了，但是出错的原因是因为上面几行的一些中间结果出错导致；或者程序根本没有报错，但是最后的结果不对。这种时候就需要调试和优化。

MATLAB 程序的调试和优化

在 MATLAB 的程序调试过程中，不仅要求程序能够满足设计者的设计需求，而且还要求程序调试能够优化程序的性能，这样使得程序调试有时比程序设计更为复杂。MATLAB 提供了强大的程序调试功能，合理的运用 MATLAB 提供的程序调试工具尤其重要。

MATLAB 程序调试方法和过程（1）

MATLAB 是一种解释和执行同时进行的语言，这使得程序的调试变得相对便利，尤其是 MATLAB 具有良好的所见即所得特性。在 MATLAB 程序调试过程中，可运用的除了一系列调试函数外，MATLAB 还提供了专门的调试器，即 M 文件编译器，通过该 M 文件编译器和调试函数的共同使用，用户能够完成大部分的程序调试工作。

1. 调试的基本任务

程序调试（Debug）的基本任务就是要找到并去除程序中的错误。程序的错误大致可以分为如下三类。

语法错误：由于程序员疏忽、输入不正确等原因而造成的代码违背程序语言规则的错误。

运行错误：由于对所求解问题的理解差异，导致程序流程出错或对程序本身的特性认识有误而造成的程序执行结果错误的情况。

异常：程序执行过程中由于不满足条件而造成的程序执行错误。

语法错误是初学者最常犯的错误，例如，变量或函数名拼写错误、缺少引号或括号等。这类错误对于熟练掌握 MATLAB 的用户来说很容易避免，并且当 MATLAB 运行发现这些错误时会立即标识出这些错误，并向用户说明错误的类型以及在 M 文件中的位置，下面用一个例子来说明，在 debug.m 文件中输入 如下内容：

```
1 1 A=[1 2 3,4 5 6,7 8 9];
```

```
%定义矩阵 A
```

```
2 2 B=[1 2 3 4,5 6 7 8,9 10 11 12,13 14 15 16];
3 3 C=A*B
```

%定义矩阵 B
%C 为矩阵 A 和 B 相乘

运行时 MATLAB 命令窗口会给出如下错误提示:

1. ??? Error using ==> mtimes
2. Inner matrix dimensions must agree.

在上述矩阵四则运算的例子中，矩阵 A 和矩阵 B 的维数不满足运算前置条件，即两个矩阵的维数不同不能进行运算。

运行错误也能够被 MATLAB 发现，但是用户却不知道错误到底发生在何处，也就不能通过查询函数工作区域的方法来查询错误来源，更多时候是 MATLAB 无法发现运行错误，但是运行结果在验证时出错。这类错误的处理方法多是依靠编程经验解决，下面就求解方程组的例子来进行说明，在命令窗口中输入以下指令：

1. >> A=[1 2 3;4 5 6;7 8 9];
 2. >>B=[9 8 7;6 5 4;3 2 1];
 3. >>x = B/A
- %x 为矩阵 B 除以 A

运行结果为：

1. x =
2. -2.6667 0 1.6667
3. -2.1667 0 1.1667
4. -1.6667 0 0.6667

该结果在不同计算机的不同版本的 MATLAB 下可能不完全相同。为了验证这个结果，在命令窗口中输入如下命令：

1. >> A*x-B; %矩阵 A,x 和 B 进行计算
2. >> norm(A*x-B) %返回表达式计算结果的最大奇异值

运算结果为：

1. ans =

2. 74.4236

显然 x 不是 $A*x=B$ 的解。

说明这就是一个简单的运行错误，MATLAB 同样有运行结果，但是进行验证时结果却不正确。原因是在求解 $A*x=B$ 方程的解时，应该不能用 B 右除 A ，而应该是左除，例如，在 MATLAB 命令窗口输入如下命令：

1. `>> x=A\B`

运行结果为：

```
1. x =  
2.   -27  -26  -17  
3.    42   41   24  
4.   -16  -16   -8
```

验证结果为：

```
1. >> A*x-B  
2. ans =  
3.     0     0     0  
4.     0     0     0  
5.     0     0     0  
6. >> norm(A*x-B)  
7. ans =  
8.     0
```

运行错误通常很难发现，用户在分析问题时要做到非常细心，并且有时需要做必要的验证。

异常的错误往往出现在规模较大的 MATLAB 程序中，并且涉及多个函数的调研以及数据的调用，异常的种类也很多，例如，被调用的文件不存在、数据传输路径错误、异常的数据输入等。

MATLAB 程序调试方法和过程（2）

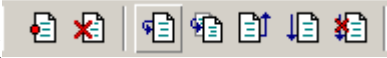
2. 调试工具

MATLAB 提供了大量的调试函数供用户使用,这些函数可以通过 help 指令获得,在 MATLAB 命令执行窗口输入如下指令:

1. >> help debug

用户便可获得这些函数,这些函数都有一个特点,就是以“db”开头,具体功能和作用如下:

1.	dbstop	- Set breakpoint	% 设置断点
2.	dbclear	- Remove breakpoint	% 清除断点
3.	dbcont	- Resume execution	% 重新执行
4.	dbdown	- Change local workspace context	% 下移本地工作空间内容
5.	dbmex	- Enable MEX-file debugging	% 使 MEX 文件调试有效
6.	dbstack	- List who called whom	% 列出函数调用关系
7.	dbstatus	- List all breakpoints	% 列出所有断点
8.	dbstep	- Execute one or more lines	% 单步或多步执行
9.	dbtype	- List M-file with line numbers	% 列出 M 文件
10.	dbup	- Change local workspace context	% 上移本地工作空间内容
11.	dbquit	- Quit debug mode	% 退出调试模式

在 MATLAB 中,这些调试函数都有相应的图形化调试工具,使得程序的调试更加方便、快捷。这些图形化调试工具在, MATLABM 文件编译器的“debug”和“Breakpoints”菜单中,和调试工具栏  上以方便调试使用。

3. 调试方法

对于简单的 MATLAB 程序中出现的语法错误,可以采用直接调试法,即直接运行该 M 文件,MATLAB 将直接找出语法错误的类型和出现的地方,根据 MATLAB 的反馈信息对语法错误进行修改。

当 M 文件很大或 M 文件中含有复杂的嵌套时,则需要使用 MATLAB 调试器来对程序进行调试,即使用 MATLAB 提供的大量调试函数以及与之相对应的图形化工具。

下面通过一个判断 2000 年至 2010 年间的闰年年份的示例来介绍 MATLAB 调试器的使用方法。

(1) 创建一个 leapyear.m 的 M 函数文件,并输入如下函数代码程序。

```

1. %程序为判断 2000 年至 2010 年 10 年间的闰年年份
2. %本程序没有输入/输出变量
3. %函数的使用格式为 leapyear, 输出结果为 2000 年至 2010 年 10 年间的闰年年份
4. function leapyear          %定义函数 leapyear
5. for year=2000:2010        %定义循环区间
6.     sign=1;
7.     a = rem(year,100);      %求 year 除以 100 后的剩余数
8.     b = rem(year,4);        %求 year 除以 4 后的剩余数
9.     c = rem(year,400);      %求 year 除以 400 后的剩余数
10.    if a=0                  %以下根据 a、b、c 是否为 0 对标志变量 sign 进行处理
11.        signsign=sign-1;
12.    end
13.    if b=0
14.        signsign=sign+1;
15.    end
16.    if c=0
17.        signsign=sign+1;
18.    end
19.    if sign=1
20.        fprintf('%4d \n',year)
21.    end
22. end

```

(2) 运行以上 M 程序，此时 **MATLAB 命令窗口会给出如下错误提示：**

1. ??? Error: File: leapyear.m **Line: 10 Column: 6**
2. The expression to the left of the equals sign is not a valid target for an assignment.

由错误提示可知，在程序的**第 10 行**存在语法错误，检测可知 if 选择判断语句中，用户将“==”写成了“=”。因此将“=”改成“==”，同时也更改第 13、16、19 行中的“=”为“==”。

MATLAB 程序调试方法和过程 (3)

(3) 程序修改并保存完成后，可直接运行修正后的程序，程序运行结果为：

1. 2001
2. 2002
3. 2003
4. 2005
5. 2006
6. 2007
7. 2009
8. 2010

显然，2001 年至 2010 年间不可能每年都是闰年，由此判断程序存在运行错误。

(4) 分析原因。可能由于在处理年号是否是 100 的倍数时，变量 sign 存在逻辑错误。

(5) 断点设置。断点为 MATLAB 程序运行时人为设置的中断点，程序运行至断点时便自动停止运行，等待用户的下一步操作。设置断点只需要用鼠标单击程序左侧标尺栏上的“-”使得“-”变成红色的圆点（当存在语法错误时圆点颜色为灰色），如图 3.2 所示(也可用“debug”和“Breakpoints”菜单，和调试工具栏来设置断点)。应该在可能存在逻辑错误或需要显示相关代码执行数据附近设置断点，例如，本例中的 12、15 和 18 行。如果用户需要去除断点，可以再次单击红色圆点去除，也可以单击工具栏中的工具去除所有断点。

(6) 运行程序。按“F5”键或单击工具栏中的 按钮执行程序，这时其他调试按钮将被激活。程序运行至第一个断点暂停，在断点右侧则出现向右指向的绿色箭头，如图 3.3 所示。

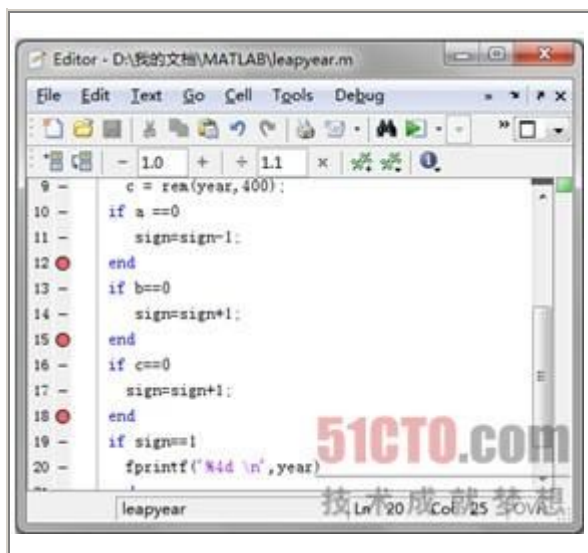


图 3.2 断点标记

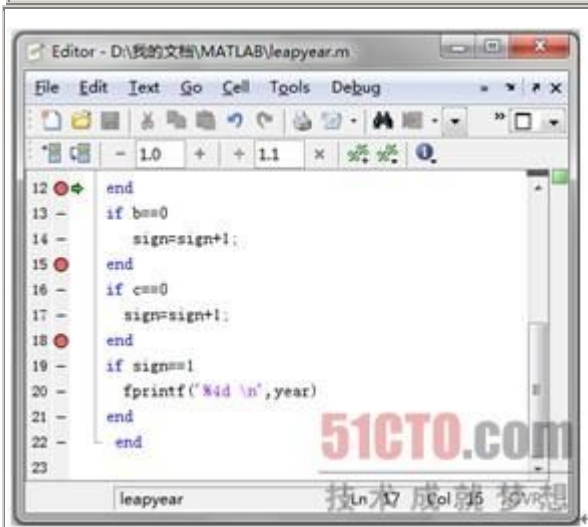


图 3.3 程序运行至断点处暂停

程序调试运行时，在 MATLAB 的命令窗口中将显示如下内容：

1. K>>

此时可以输入一些调试指令，更加方便对程序调试的相关中间变量进行查看。

(7) 单步调试。可以通过按“F10”键或单击工具栏中相应的单步执行图形按钮，此时程序将一步一步按照用户需求向下执行，如图 3.4 所示。

(8) 查看中间变量。可以将鼠标停留在某个变量上，MATLAB 将会自动显示该变量的当前值，也可以在 MATLAB 的 workspace 中直接查看所有中间变量的当前值，如图 3.5 所示。



图 3.4 程序单步执行



图 3.5 中间变量查看

MATLAB 程序调试方法和过程 (4)

(9) 修正代码。通过查看中间变量可知，在任何情况下 sign 的值都是 1，此时调整修改代码程序如下所示。

1. %程序为判断 2000 年至 2010 年 10 年间的闰年年份
2. %本程序没有输入/输出变量

```
3. %函数的使用格式为 leapyear, 输出结果为 2000 年至 2010 年 10 年间的闰年年份
4. function leapyear
5. for year=2000:2010
6.     sign=0;
7.     a = rem(year,400);
8.     b = rem(year,4);
9.     c = rem(year,100);
10.    if a ==0
11.        signsign=sign+1;
12.    end
13.    if b==0
14.        signsign=sign+1;
15.    end
16.    if c==0
17.        signsign=sign-1;
18.    end
19.    if sign==1
20.        fprintf('%4d \n',year)
21.    end
22. end
```

按“F5”键再次执行程序，得到的运行结果如下：

```
1. 2000
2. 2004
3. 2008
```

分析发现，结果正确，此时程序调试结束。