



信陽師範学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

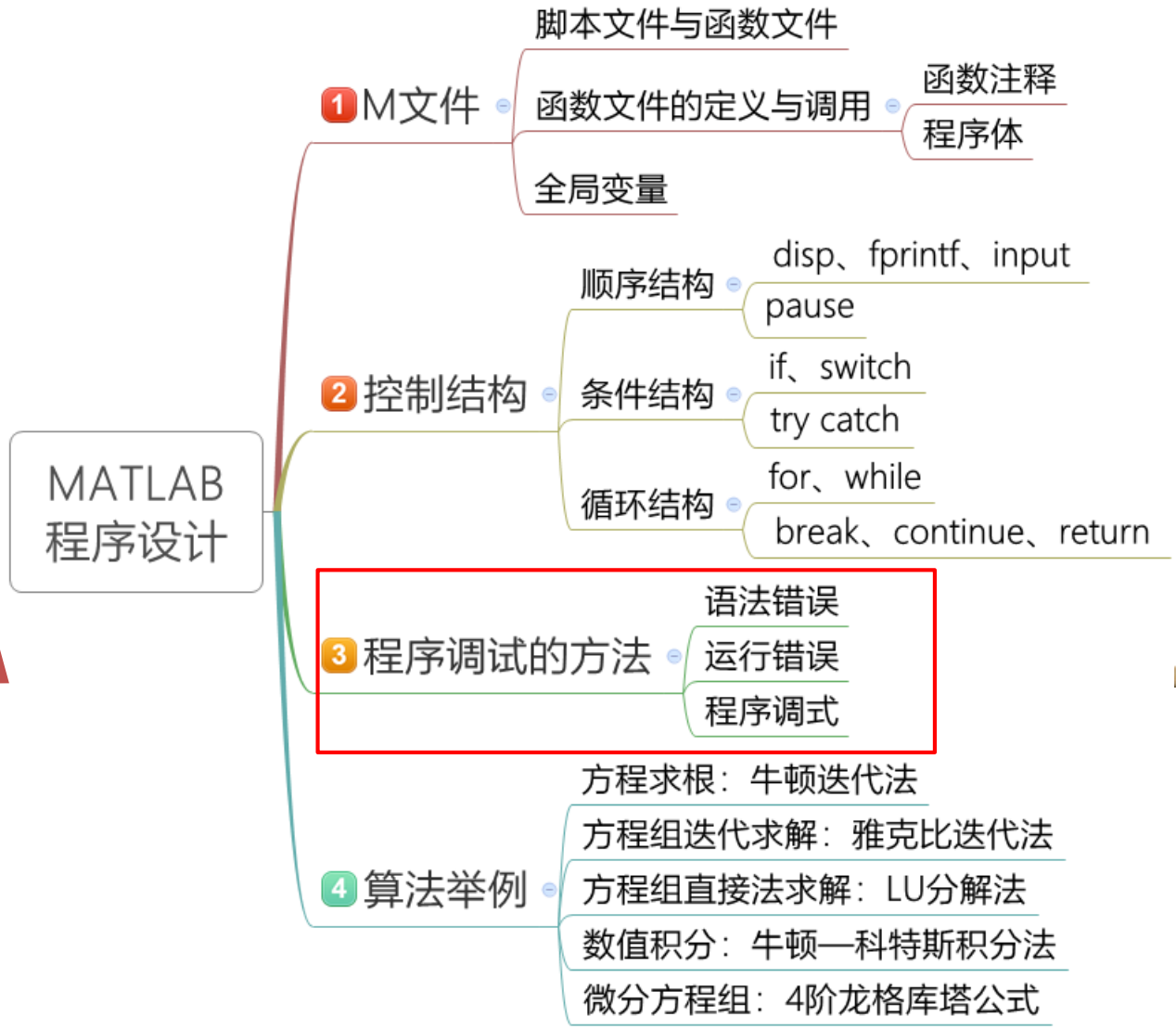
# 第9章 MATLAB程序设计

**讲授人**：牛言涛

**日期**：2020年4月3日

CONTENTS

# 目录


MATLAB
程序设计
1 M文件
脚本文件与函数文件
函数文件的定义与调用
函数注释
程序体
全局变量
2 控制结构
顺序结构
disp、fprintf、input
pause
条件结构
if、switch
try catch
循环结构
for、while
break、continue、return
3 程序调试的方法
语法错误
运行错误
程序调式
4 算法举例
方程求根：牛顿迭代法
方程组迭代求解：雅克比迭代法
方程组直接法求解：LU分解法
数值积分：牛顿—科特斯积分法
微分方程组：4阶龙格库塔公式


01
IDEA
02
LEARNING
03
CREATIVE
INSPIRATION

# 9.3 程序调试的方法



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。

**程序调试（Debug）的基本任务就是要找到并去除程序中的错误。**

- 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。这类错误也称为程序逻辑错误。
- 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。

# 1. 语法错误

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。如在debug_ex1.m文件中输入：

debug_ex1.m

```
1 —    A = [1 2 3,4 5 6,7 8 9];     %定义矩阵A
2 —    B = [1 2 3 4,5 6 7 8,9 10 11 12,13 14 15 16];  %定义矩阵B
3 —    C = A*B;    %C为矩阵A和B相乘
4 —    disp(C)
```

命令行窗口

```
>> debug_ex1
错误使用 *
内部矩阵维度必须一致。
出错 debug_ex1 (line 3)
C = A*B;    %C?        A    B
```

在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。

debug_ex1.m

```
1 —    A = [1 2 3,4 5 6,7 8 9];     %定义矩阵A
2 —    B = [1 2 3 4,5 6 7 8,9 10 11 12,13 14 15 16];  %定义矩阵B
3 —    [r1,c1] = size(A);
4 —    [r2,c2] = size(B);
5 —    if c1 ~= r2
6 —        error('矩阵维度不一致，不能进行矩阵乘法运算！')
7 —    end
8 —    C = A*B;    %C为矩阵A和B相乘
9 —    disp(C)
```

# 2. 运行错误

信阳师范学院
数学与统计学院

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。如

```
debug_ex2.m
1 —    A=[6 2 3;5 4 6;7 6 4];
2 —    B=[9 8 7;6 7 4;3 2 3];
3 —    x = B/A;   %x为矩阵B除以A
4 —    disp(x)
```

```
命令行窗口
>> debug_ex2
   -0.1429    0.5714    1.0000
   -0.6000    0.1000    1.3000
    0.1429    0.4286         0
```

```
>> errC = A*x-B   %矩阵A,x和B进行计算
errC =
  -10.6286   -3.0857    1.6000
   -8.2571   -1.1714    6.2000
   -7.0286    4.3143   11.8000
>> errN = norm(A*x-B) %返回表达式计算结果的最大奇异值
errN =
   18.9571
```

程序运算没有出现任何语法错误，但为了验证结果，在命令窗口中输入命令验证：

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。

# 2. 运行错误

```
A=[6 2 3;5 4 6;7 6 4];
B=[9 8 7;6 7 4;3 2 3];
x = A\B;  %x为矩阵B除以A
errMax = norm(A*x-B);
if errMax >= 1e-12 %返回表达式计算结果的最大奇异值
    error('方程组求解错误，或精度不够！')
end
disp('方程组的解：')
disp(x)
disp(['计算结果的最大奇异值为：',num2str(errMax)])
```

```
>> debug_ex2
方程组的解：
    1.7143    1.2857    1.4286
   -2.1857   -2.2143   -1.4714
    1.0286    1.5714    0.4571
计算结果的最大奇异值为：2.1433e-15
```

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。

# 3. 程序调试

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

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 debug指令获得。

```
dbstop    - Set breakpoint                          %设置断点
dbclear   - Remove breakpoint                       %清除断点
dbcont    - Resume execution                        %重新执行
dbdown    - Change local workspace context         %下移本地工作空间内容
dbmex     - Enable MEX-file debugging               %使MEX文件调试有效
dbstack   - List who called whom                    %列出函数调用关系
dbstatus  - List all breakpoints                    %列出所有断点
dbstep    - Execute one or more lines               %单步或多步执行
dbtype    - List M-file with line numbers           %列出M文件
dbup      - Change local workspace context         %上移本地工作空间内容
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MATLAB编译器的"debug"和"Breakpoints"菜单中，以方便调试使用。

# 4. 案例——公历闰年计算方法

1、普通年能被4整除且不能被100整除的为闰年。（如2004年就是闰年,1900年不是闰年）

2、世纪年能被400整除的是闰年。(如2000年是闰年，1900年不是闰年)

leapyear_error.m

```
function leapyear_error(yspan)
    fprintf('指定范围【%d,%d】内闰年为：\n',yspan(1),yspan(2));
    for year = yspan(1):yspan(2)   %定义循环区间
        sign = 1;
        a = rem(year,100);      %求year除以100后的剩余数
        b = rem(year,4);        %求year除以4后的剩余数
        c = rem(year,400);      %求year除以400后的剩余数
        if a == 0       %以下根据a、b、c是否为0对标志变量sign进行处理
            sign = sign - 1;
        end
        if b == 0
            sign = sign + 1;
        end
        if c == 0
            signs = sign + 1;
        end
        if sign == 1
            fprintf('%4d \n',year);
        end
    end
end
```

首先处理M文件中那些标记有红色波浪线的代码，这里发现变量符号的书写错误。这种错误是一种语法错误，但没有被程序运行检测到。

```
>> leapyear_error([2000,2010])
指定范围【2000,2010】内闰年为：
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
```

# 4. 案例——公历闰年计算方法

1、普通年能被4整除且不能被100整除的为闰年。（如2004年就是闰年,1900年不是闰年）

2、世纪年能被400整除的是闰年。(如2000年是闰年，1900年不是闰年)

leapyear_error.m

```
function leapyear_error(yspan)
   fprintf('指定范围【%d,%d】内闰年为：\n',yspan(1),yspan(2));
   for year = yspan(1):yspan(2)   %定义循环区间
      sign = 1;
      a = rem(year,100);      %求year除以100后的剩余数
      b = rem(year,4);        %求year除以4后的剩余数
      c = rem(year,400);      %求year除以400后的剩余数
      if a == 0      %以下根据a、b、c是否为0对标志变量sign进行处理
         sign = sign - 1;
      end
      if b == 0
         sign = sign + 1;
      end
      if c == 0
         sign = sign + 1;
      end
      if sign == 1
         fprintf('%4d \n',year);
      end
   end
end
```

1、分析原因：程序的输出取决于sign是否为1，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。

```
>> leapyear_error([2000,2010])
指定范围【2000,2010】内闰年为：
2001
2002
2003
2005
2006
2007
2009
2010
```

显然程序存在逻辑错误，程序的输出结果都不是闰年。而闰年没有得到输出

# 4. 案例——公历闰年计算方法

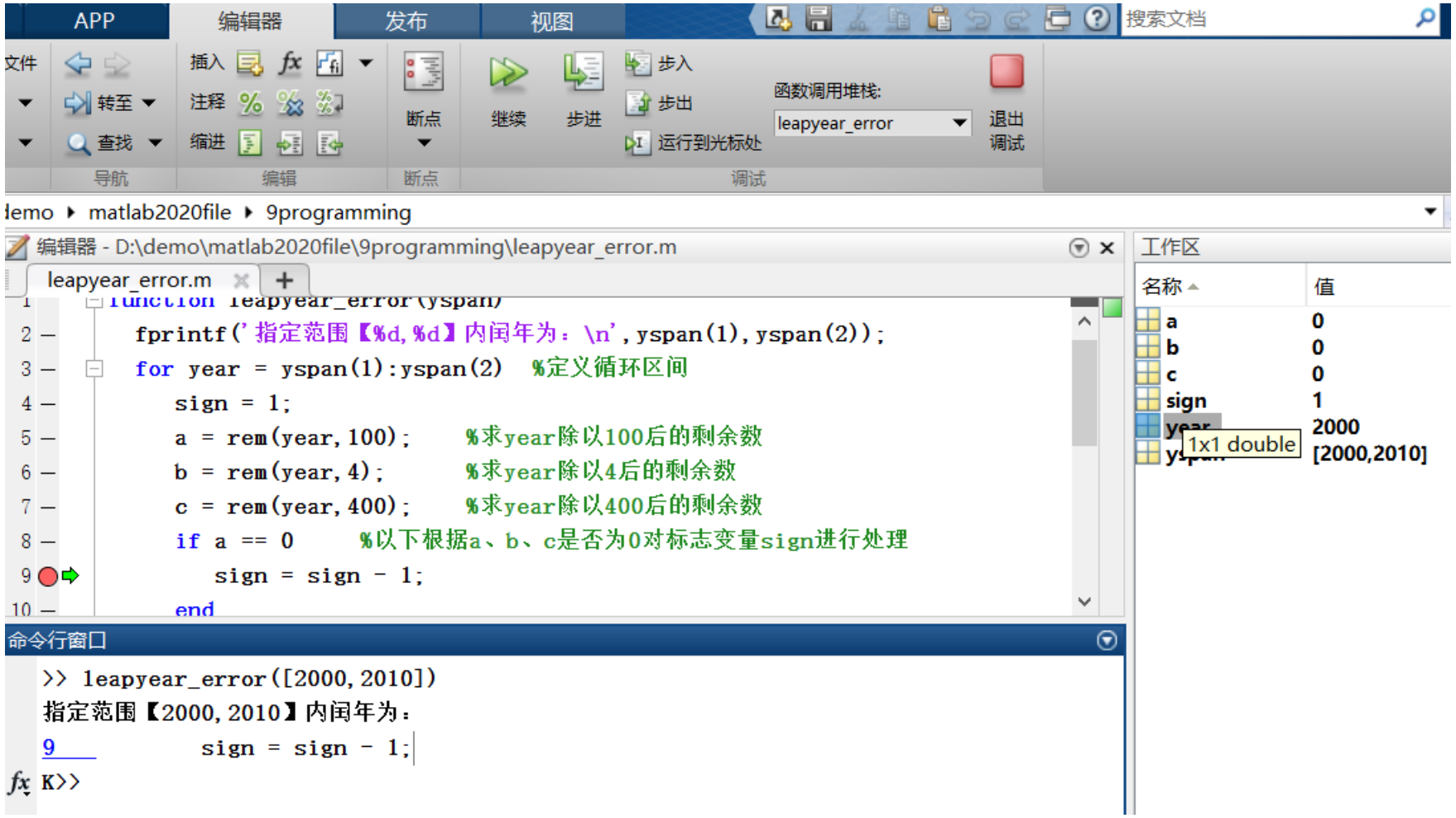
2、断点设置：断点为程序执行时人为设置的中断点，程序运行至断点时便自动停止运行，等待用户的下一步操作。

- 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）。
- 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。
- 如果需要去除断点，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。

```
8 —      if a == 0      %以下根据a、b、c是否为0对标志变量sign进行处理
9 ●         sign = sign - 1;
10 —     end
11 —     if b == 0
12 ●        sign = sign + 1;
13 —     end
14 —     if c == 0
15 ●        sign = sign + 1;
16 —     end
17 —     if sign == 1
18 ●        fprintf('%4d \n',year);
19 —     end
```

# 4. 案例——公历闰年计算方法

3、运行程序：按“F5”键，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。



# 4. 案例——公历闰年计算方法

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

**4、程序调试运行时，在MATLAB的命令窗口中将显示如下内容：**

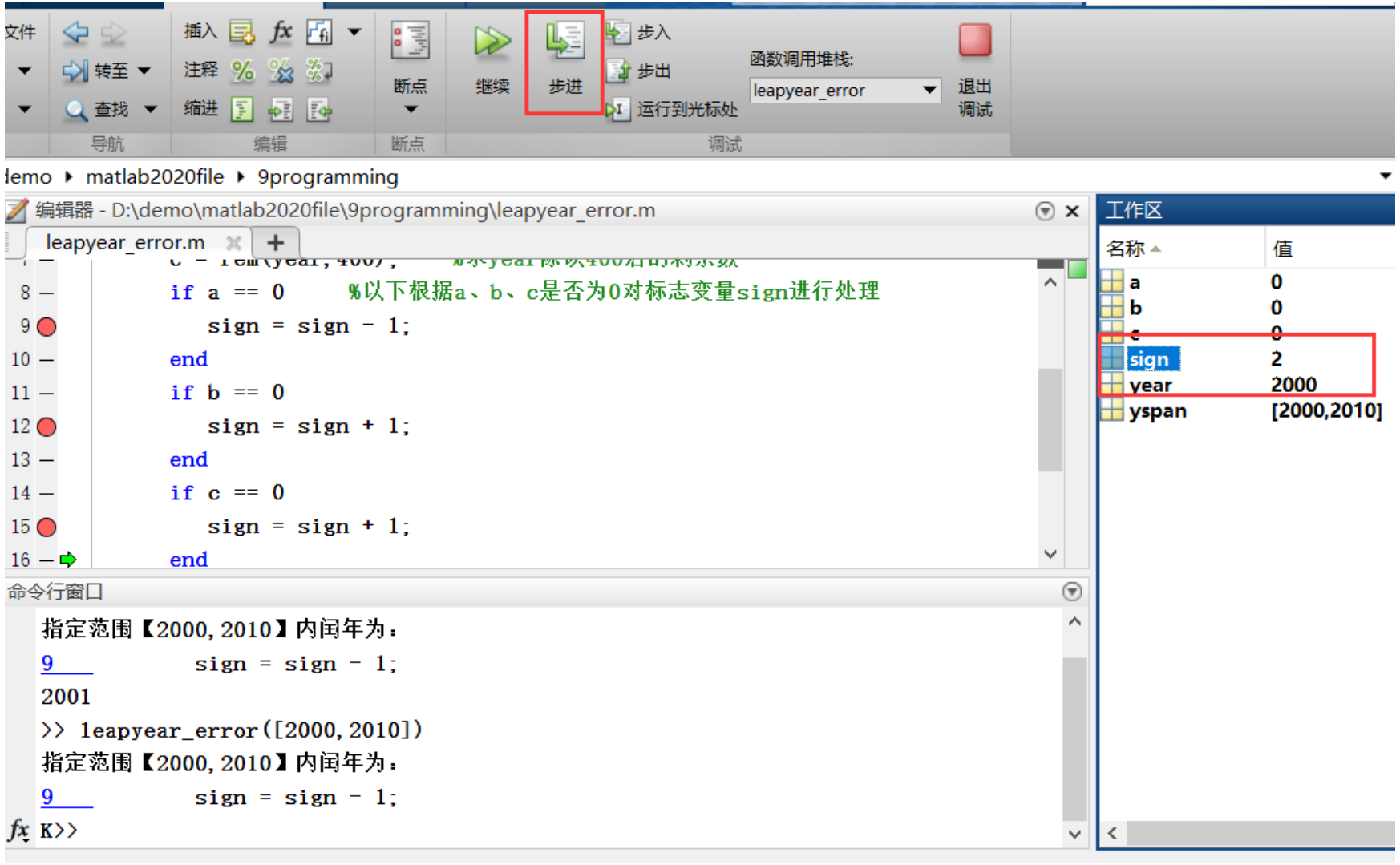
- K>>
- **此时可以输入一些调试指令，更加方便对程序调试的相关中间变量进行查看。**

| 说明 | 工具栏按钮 | 备用函数 |
| --- | --- | --- |
| 继续执行文件，直到光标所在行。也可从上下文菜单中获得。 | 运行到光标处 | 无 |
| 执行当前文件行。 | 步长 | dbstep |
| 执行当前文件行，如果该行调用另一个函数，则步入该函数。 | 步入 | dbstep in |
| 继续执行文件，直到完成或遇到另一断点为止。 | 继续 | dbcont |
| 步入后，运行被调用函数或局部函数的其余部分，离开被调用函数并暂停。 | 步出 | dbstep out |
| 暂停调试模式。 | 暂停 | 无 |
| 退出调试模式。 | 退出调试 | dbquit |

# 4. 案例——公历闰年计算方法

5、单步调试：可以通过按“F10”键或单击工具栏中“步进”按钮，此时程序将一步一步按照用户需求向下执行。

6、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。



# 4. 案例——公历闰年计算方法

信阳师范学院
数学与统计学院

- 发现当year = 2000的时候，sign = 2，此时第15行代码尝试修改为sign = sign − 1。
- 继续运行，当year = 2001的时候，sign = 1，各if语句并未起到作用，怀疑是否sign初始化为1的时候错误，尝试修改为sign = 0，即不是闰年，各if不起作用，不输出。
- 重新运行，发现当year = 2000时，最终sign = -1，

  此时怀疑存在逻辑上的错误。
- 既然2000能被100和4整除，也能被400整数， sign两处都减一操作，有误，修改代码第9行为sign = sign + 1.

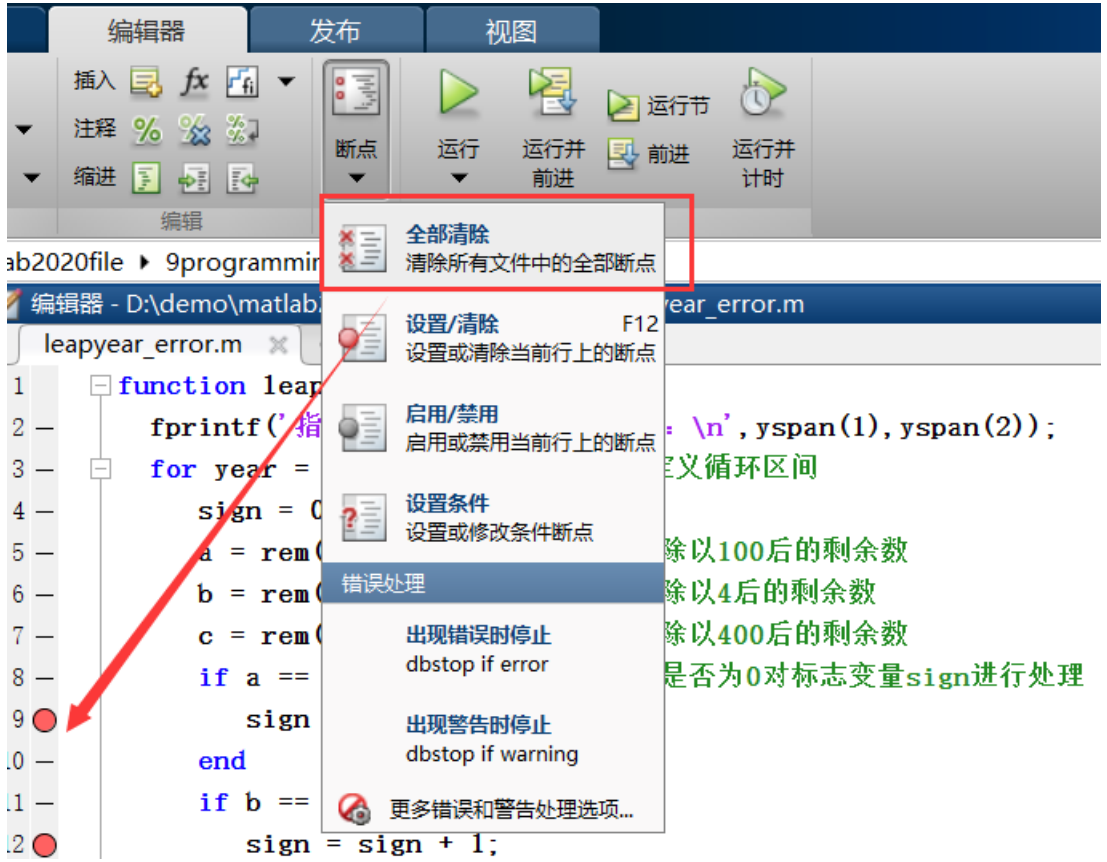
  2004能被4整数，不能被100和400整除，故b处if语句起作用，加一操作，并进行输出。
- 继续执行，发现程序运行正确，不存在逻辑错误。

leapyear_error.m

```
1  function leapyear_error(yspan)
2      fprintf('指定范围【%d,%d】内闰年为：\n',yspan(1),yspan(2));
3      for year = yspan(1):yspan(2)   %定义循环区间
4          sign = 0;
5          a = rem(year,100);     %求year除以100后的剩余数
6          b = rem(year,4);       %求year除以4后的剩余数
7          c = rem(year,400);     %求year除以400后的剩余数
8          if a == 0      %以下根据a、b、c是否为0对标志变量sign进行处理
9              sign = sign + 1;
10         end
11         if b == 0
12             sign = sign + 1;
13         end
14         if c == 0
15             sign = sign - 1;
16         end
17         if sign == 1
18             fprintf('%4d \n',year);
19         end
20     end
21 end
```

# 4. 案例——公历闰年计算方法

通过菜单清除全部断点，重新运行程序，得到正确结果。




```
>> leapyear_error([2000,2020])
指定范围【2000,2020】内闰年为：
2000
2004
2008
2012
2016
2020
```


01
IDEA
02
LEARNING
03
CREATIVE
INSPIRATION
EDUCATION
BUSINESS
PHOTO BOOK

# 感谢聆听