



信阳师范学院  
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

# 目录

## CONTENTS

### 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



# 1. 控制输入输出与程序的暂停

命 令	说 明
disp(A) , disp('text')	显示数组A的内容，而不打印数组的名称。显示单引号内部的文本串text。
format	控制屏幕输出的显示格式
fprintf	执行格式化的写入到屏幕或者写入到一个文件
x = input('text')	显示单引号中的文本，等待用户的键盘输入，并且将输入值存储在x中
x = input('text','s')	显示单引号中的文本，等待用户的键盘输入，并且将输入作为字符串存储在x中
[indx,tf] = listdlg('ListString',list,Name,Value)	创建一个模态对话框，允许用户从指定的列表中选择一个或多个项目。 list 值是要显示在对话框中的项目列表。menu不再被推荐使用。

- 暂停程序的执行可以使用pause函数，其调用格式为： **pause(延迟秒数)**
- 如果省略延迟时间，直接使用pause，则将暂停程序，直到用户按任一键后程序继续执行。
- 如果意外创建了一个无限循环（即永远不会自行结束的循环），按下 Ctrl+C 停止执行循环。

# 1. 控制输入输出与程序的暂停

- `fprintf(fileID,formatSpec,A1,...,An)` 按列顺序将 `formatSpec` 应用于数组 `A1,...,An` 的所有元素，并将数据写入到一个文本文件。`fprintf` 使用在对 `fopen` 的调用中指定的编码方案。
- `fprintf(formatSpec,A1,...,An)` 设置数据的格式并在屏幕上显示结果。
  - `formatSpec` 使用格式化操作符指定。`formatSpec` 还可以包括普通文本和特殊字符。
  - `formatSpec` 可以用单引号引起来的字符向量，从 R2016b 开始，也可以是字符串标量。
  - 格式化操作符以百分号 `%` 开头，以转换字符结尾。转换字符是必需的。您也可以在 `%` 和转换字符之间指定标识符、标志、字段宽度、精度和子类型操作符。（操作符之间的空格无效）



# 1. 控制输入和输出

## 把计算结果写入到文本文件

bengS %调用脚本文件

%以写方式打开文件bangS.text, 文件不存在, 则创建

```
fileID = fopen('bungeeS.txt','w');
```

%对字符串tf和Vval指定宽度, 并写入文件

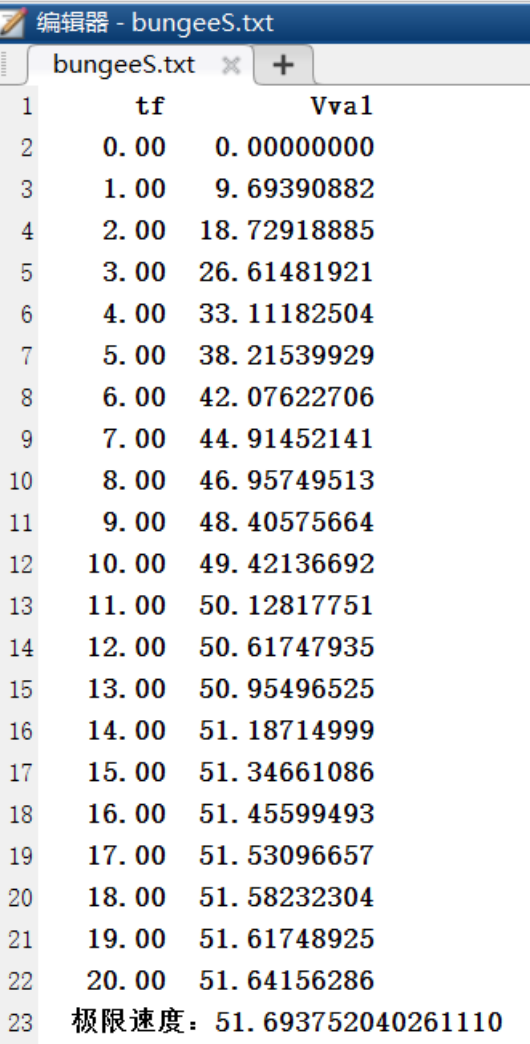
```
fprintf(fileID,'%6s %12s\n','tf','Vval');
```

%给定宽度和精度要求

```
fprintf(fileID,'%6.2f %14.8f\n',[tf;vval]);
```

```
fprintf(fileID,'极限速度: %.8f',limS);
```

```
fclose(fileID); %关闭文件
```



1	tf	Vval
2	0.00	0.00000000
3	1.00	9.69390882
4	2.00	18.72918885
5	3.00	26.61481921
6	4.00	33.11182504
7	5.00	38.21539929
8	6.00	42.07622706
9	7.00	44.91452141
10	8.00	46.95749513
11	9.00	48.40575664
12	10.00	49.42136692
13	11.00	50.12817751
14	12.00	50.61747935
15	13.00	50.95496525
16	14.00	51.18714999
17	15.00	51.34661086
18	16.00	51.45599493
19	17.00	51.53096657
20	18.00	51.58232304
21	19.00	51.61748925
22	20.00	51.64156286
23	极限速度: 51.693752040261110	

# fprintf的转换字符

值类型	转换	详细信息
有符号整数	%d 或 %i	以 10 为基数
无符号整数	%u	以 10 为基数
	%o	以 8 为基数（八进制）
	%x	以 16 为基数（十六进制），小写字母 a–f
	%X	与 %x 相同，大写字母 A–F
浮点数	%f	定点记数法（使用精度操作符指定小数点后的位数。）
	%e	指数记数法，例如 3.141593e+00（使用精度操作符指定小数点后的位数）。
	%E	与 %e 相同，但为大写，例如 3.141593E+00。
	%g	更紧凑的 %e 或 %f，不带尾随零（使用精度操作符指定有效数字位数。）
	%G	更紧凑的 %E 或 %f，不带尾随零（使用精度操作符指定有效数字位数。）
字符或字符串	%c	单个字符
	%s	字符向量或字符串数组。输出文本的类型与 formatSpec 的类型相同。

- 可选标识符、标志、字段宽度、精度和子类型操作符进一步定义了输出文本的格式。
- 标识符：处理函数输入参数的顺序。使用语法 `n$`，其中 `n` 代表函数调用中其他输入参数的位置。
  - 示例：(`'%3$s %2$s %1$s %2$s'`, `'A'`, `'B'`, `'C'`) 将输入参数 `'A'`、`'B'`、`'C'` 输出为：C B A B。
  - 注意：如果输入参数为数组，则不能使用标识符指定该输入参数中的特定数组元素。
- 标志

'-'	左对齐。示例： <code>%-5.2f</code> 、 <code>%-10s</code>
'+'	始终为任何数值输出符号字符（+ 或 -）。右对齐文本。示例： <code>%+5.2f</code> 、 <code>%+10s</code>
' '	在值之前插入空格。示例： <code>% 5.2f</code>
'0'	在值之前补零以填充字段宽度。示例： <code>%05.2f</code>
'#'	修改选定的数值转换：对于 <code>%o</code> 、 <code>%x</code> 或 <code>%X</code> ，将输出 <code>0</code> 、 <code>0x</code> 或 <code>0X</code> 前缀。对于 <code>%f</code> 、 <code>%e</code> 或 <code>%E</code> ，即使精度为零也将输出小数点。对于 <code>%g</code> 或 <code>%G</code> ，不删除尾随零或小数点。示例： <code>%#5.0f</code>

# fprintf的可选操作符

- 字段宽度

要输出的最低字符数。字段宽度操作符可以是数字，也可以是指向输入参数的星号 (\*)。

当您将 \* 指定为字段宽度操作符时，其他输入参数必须指定打印宽度和要打印的值。宽度和值可以是参数对组，也可以是数值数组中的对组。使用 \* 作为字段宽度操作符时，您可以打印具有不同宽度的不同值。

**示例：**输入参数 ('%12d',intmax) 等效于 ('%\*d',12,intmax)。

**示例：**输入参数 ('%\*d',[2 10 5 100]) 返回 '10 100'，其中两个空格分配给 10，五个空格分配给 100。您也可以将字段宽度和值指定为多个参数，如 ('%\*d',2,10,5,100) 中所示。

除非标志另行指定，否则该函数使用空格填充值之前的字段宽度。

[注：其他内容通过help进行学习。](#)

- 精度

对于 %f、%e 或 %E	小数点右侧的位数 <b>示例：</b> '%.4f' 将 pi 输出为 '3.1416'
对于 %g 或 %G	有效位数 <b>示例：</b> '%.4g' 将 pi 输出为 '3.142'

精度操作符可以是数字，也可以是指向参数的星号 (\*)。

当您将 \* 指定为字段精度操作符时，其他输入参数必须指定打印精度和要打印的值。精度和值可以是参数对组，也可以是数值数组中的对组。使用 \* 作为精度操作符时，您可以打印具有不同精度的不同值。

将 \*.\* 指定为字段宽度和精度操作符时，必须以三元组形式指定字段宽度、精度和值。

**示例：**输入参数 ('%.4f',pi) 等效于 ('%.\*f',4,pi)。

**示例：**输入参数 ('%6.4f',pi) 等效于 ('%.\*f',6,4,pi)。

**示例：**输入参数 ('%\*.\*f',6,4,pi,9,6,exp(1)) 返回 '3.1416 2.718282'，其中以 9 和 6 分别作为 exp(1) 输出的字段宽度和精度。



# 1. 控制输入和输出

```
编辑器 - D:\demo\matlab2020file\9programming\solve_eqex.m
solve_eqex.m x +
1  % 以下为求解一元二次方程的脚本文件，简单演示控制输入和输出
2  disp(' 以下为求解一元二次方程a*x^2 +b*x+c=0的根')
3
4  %% 控制函数的输入，交互输入
5  a = input(' 请输入变量a的值: ');
6  b = input(' 请输入变量b的值: ');
7  c = input(' 请输入变量c的值: ');
8
9  %% 根的求解公式
10 delta = b*b-4*a*c;
11 x = [(-b+sqrt(delta))/(2*a); (-b-sqrt(delta))/(2*a)];
12
13 %% 控制函数的输出
14 if delta < 0
15     disp(' 存在两个虚根: ');
16     % 格式化输出
17     fprintf(' x1 = %.8g + %.8gi \n x2 = %.8g - %.8gi.\n',...
18         x(1), imag(x(1)), x(2), abs(imag(x(2))));
19 elseif delta == 0
20     disp(' 存在两个相同的实根: ');
21     fprintf(' x = %.8f \n.', x(1)); % 格式化输出
22 else
23     disp(' 存在两个不同的实根: ')
24     % 格式化输出
25     fprintf(' x1 = %.8f \n x2 = %.8f.\n', x(1), x(2));
26 end
```

例1: 求一元二次方程  $ax^2 + bx + c = 0$  的根。

```
>> solve_eqex
```

以下为求解一元二次方程 $a*x^2 + b*x + c = 0$ 的根

请输入变量a的值: 3

请输入变量b的值: 3

请输入变量c的值: 3

存在两个虚根:

x1 = -0.5 + 0.8660254i

x2 = -0.5 - 0.8660254i.

## 2. 条件结构——if语句

条件语句可用于在运行时选择要执行的代码块。当希望针对一组已知值测试相等性时，请使用 switch 语句。对于 if 和 switch，MATLAB执行与第一个 true 条件相对应的代码，然后退出。

### 单分支if语句

```
if 条件
    语句组
end
```

条件表达式可以包含  
**关系运算符**（如 < 或 ==）和**逻辑运算符**（如 &&、|| 或 ~）。

### 双分支if语句

```
if 条件
    语句组1
else
    语句组2
end
```

条件表达式使用**逻辑运算符 and 和 or** 创建复合表达式。

### 多分支if语句

```
if 条件1
    语句组1
elseif 条件2
    语句组2
.....
else
    语句组n
end
```

## 2. 条件结构——if语句

```
ifdemo.m  +
1 — limit = 0.75;
2 — A = rand(10,1);
3 — %使用any比较数组，只要有一个元素满足，即为真，all是全部元素满足
4 — if any(A > limit)
5 —     disp('There is at least one value above the limit.')
6 — else
7 —     disp('All values are below the limit.')
8 — end
9
10 — %使用 isequal 而不是 == 运算符比较数组来测试相等性
11 — %因为当数组的大小不同时 == 会导致错误。
12 — A = magic(5);
13 — B = rand(3,4,5);
14 — if isequal(size(A),size(B))
15 —     C = [A; B];
16 — else
17 —     disp('A and B are not the same size.')
18 —     C = [];
19 — end
20
21 — %使用 strcmp 比较字符向量。
22 — %当字符向量的大小不同时，使用 == 测试相等性会产生错误。
23 — reply = input('Would you like to see an echo? (y/n): ','s');
24 — if strcmp(reply,'y')
25 —     disp('I love you...')
26 — else
27 —     disp('bay bay...')
28 — end
```

```
30 — %表达式可以包含关系运算符（例如 < 或 ==）和逻辑运算符（例如 &&、|| 或 ~）
31 — if exist('myfunction.m','file') && (myfunction(42) >= pi)
32 —     disp('Expressions are true')
33 — else
34 —     disp('File is not exist or function call error ...')
35 — end
```

```
>> ifdemo
```

There is at least one value above the limit.

A and B are not the same size.

Would you like to see an echo? (y/n): y

I love you...

File is not exist or function call error ...

## 2. 条件结构——switch语句

switch语句  
根据表达  
式的取值  
不同，分  
别执行不  
同的语句，  
其语句格  
式为：

```
switch 表达式
    case 表达式1
        语句组1
    case 表达式2
        语句组2
    .....
    case 表达式m
        语句组m
    otherwise
        语句组n
end
```

一般而言，如果具有多个可能的离散已知值，读取 switch 语句比读取 if 语句更容易。但是，无法测试 switch 和 case 值之间的不相等性。

```
[dayNum, dayString] = weekday(date, 'long', 'en_US');
```

```
switch dayString
    case 'Monday'
        disp('Start of the work week')
    case 'Tuesday'
        disp('Day 2')
    case 'Wednesday'
        disp('Day 3')
    case 'Thursday'
        disp('Day 4')
    case 'Friday'
        disp('Last day of the work week')
    otherwise
        disp('Weekend!')
end
```

## 2. 条件结构——switch语句

例2：学生的成绩管理，演示switch结构的应用。

```
scoregrade.m
1 function scoregrade(score, name)
2 % 划分区域：满分(100)，优秀(90-99)，良好(80-89)，
3 % 中等(70-79)，及格(60-69)，重修(<60)。
4 excellent = num2cell(89 + [1:10]);
5 good = num2cell(79 + [1:10]);
6 secondary = num2cell(69 + [1:10]);
7 pass = num2cell(59 + [1:10]);
8 %数值转化为元胞数组
9 Mark = num2cell(score);
10 Rank = cell(length(Mark), 1);
11 %创建构架数组s，它有三个域。
12 S = struct('Name', name, 'Marks', Mark, 'Rank', Rank);
13 %根据学生的分数，求出相应的等级。
14 for i = 1:length(Mark)
15     switch S(i).Marks
16     case 100 %得分为100时，列为'满分'等级
17         S(i).Rank = '满分';
18     case excellent %得分在90和99之间，列为'优秀'等级
19         S(i).Rank = '优秀';
```

```
22 case good %得分在80和89之间，列为'良好'等级
23     S(i).Rank = '良好';
24 case secondary %得分在70和79之间，列为'中等'等级
25     S(i).Rank = '中等';
26 case pass %得分在60和69之间，列为'中等'等级
27     S(i).Rank = '及格';
28 otherwise %得分低于60，列为'重修'等级
29     S(i).Rank = '重修';
30 end
31 end
32 fprintf('%10s%10s%10s\n', '学生姓名', '得分', '等级');
33 for i=1:length(Mark)
34     fprintf('%9s %13.1f %10s\n', S(i).Name, S(i).Marks, S(i).Rank);
35 end;
36 end
```

命令行窗口

```
>> [score, name] = xlsread('scoregrade.xlsx', 1, 'B2:C20');
```

```
>> scoregrade(score, name)
```

学生姓名	得分	等级
林青霞	72.0	中等
张曼玉	83.0	良好
关之琳	56.0	重修
赵雅芝	94.0	优秀
张柏芝	100.0	满分
赵丽颖	88.0	良好
刘德华	95.0	优秀
田润安	60.0	及格

## 2. 条件结构——try catch语句

### 语法

```
try
    statements
catch exception
    statements
end
```

### 说明

try statements, catch statements end 执行 try 块中的语句并在 catch 块中捕获产生的错误。此方法允许您改写一组程序语句的默认错误行为。如果 try 块中的任何语句生成错误，程序控制将立即转至包含错误处理语句的 catch 块。

exception 是 MException 对象，您可以用它来标识错误。catch 块将当前异常对象分配给 exception 中的变量。

try 和 catch 块都可包含嵌套的 try/catch 语句。

### 提示：

- 不能在一个 try 块中使用多个 catch 块，但可以嵌套完整的 try/catch 块。
- 与一些其他语言不同，MATLAB 不允许在 try/catch 语句中使用 finally 块。

1. 用户输入了非法数据，或计算中数据的溢出。
  2. 要打开的文件不存在或调用的函数不存在。
  3. 网络通信时连接中断，安全性问题。
  4. 内存溢出，或因程序占用资源过大而异常终止。
  5. I/O（输入\输出）异常。
- 使用 try 和 catch 关键字可以捕获异常。

## 2. 条件结构——try catch语句

```
A = [1 2 1; 3 5 8; 2 4 2];
```

```
b = [1;2;3];
```

```
try
```

```
    C = [A;b]; %有错误（列的维度不一致）,跳转到catch语句行  
    并执行
```

```
    disp(C); %发现错误后，该行代码不执行
```

```
catch ErrorInfo %捕获到的错误是一个MException对象
```

```
    disp(ErrorInfo);
```

```
    disp(ErrorInfo.identifier);
```

```
    disp(ErrorInfo.message);
```

```
    disp(ErrorInfo.stack);
```

```
    disp(ErrorInfo.cause);
```

```
    %发生错误时的其他动作
```

```
end
```

```
trycatch_demo.m x +
1  function x = trycatch_demo(A, b)
2      % 该代码功能只是演示，不存在实际功能意义
3
4      try
5          x = A/b;
6          %y = sin(x).*e(x);
7          %B = magic(10);
8          %C = A*B;
9          %D = [A;B];
10     catch ME
11         switch ME.identifier
12             case 'MATLAB:dimagree'
13                 warning('系数矩阵和右端向量维度不一致!');
14                 x = 0;
15             case 'MATLAB:innerdim'
16                 error('内部矩阵维度必须一致!');
17             case 'MATLAB:UndefinedFunction'
18                 error('试图调用未定义或不存在的函数!');
19             case 'MATLAB:catenate:dimensionMismatch'
20                 error('串联的矩阵的维度不一致。');
21             otherwise
22                 throw(ME)
23         end
24     end
25 end
```

### 3. 循环结构——for语句

执行过程是依次将矩阵的各列元素赋给循环变量，然后执行循环体语句，直至各列元素处理完毕。

- for语句的格式为：

for 循环变量 = 表达式1:表达式2:表达式3

循环体语句

end

- for语句更一般的格式为：

for 循环变量=矩阵表达式

循环体语句

end

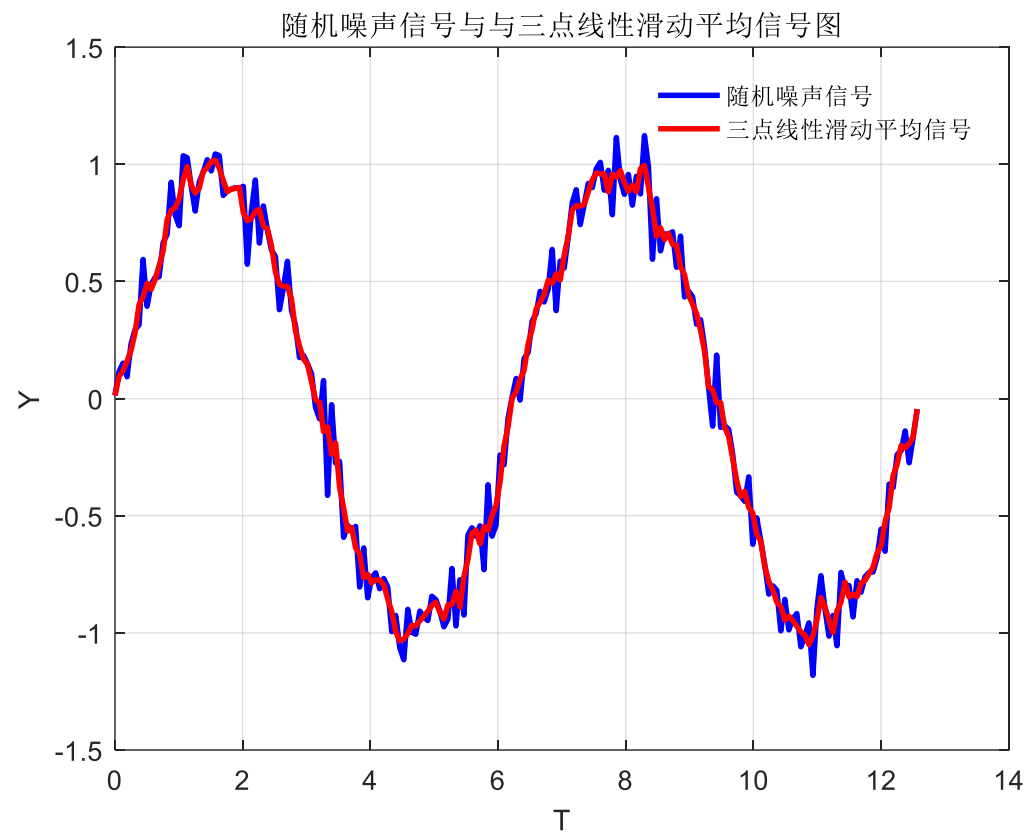
例4：有一周期为 $4\pi$ 的正弦波上叠加了方差为0.1的正态分布的随机噪声的信号，用循环结构编制一个三点线性滑动平均的程序。（用 $0.1*\text{randn}(1,n)$ 产生方差为0.1的正态分布的随机噪声，三点线性滑动平均就是依次取每三个相邻数的平均值作为新的数据。）



### 3. 循环结构——for语句

```
random_signal.m
1 function [y,ym] = random_signal(span,delta)
2 % 正弦波上叠加了方差为0.1的正态分布的随机噪声的信号
3 % 用循环结构编制一个三点线性滑动平均的程序
4 % 此处省略函数调用格式的注释...
5
6 t = span(1):pi/50:span(2); %在给定区间等分数值
7 n = length(t);
8 y = sin(t) + delta*randn(1,n); %叠加随机噪声信号
9
10 %% 实现三点线性滑动平均
11 ym(1) = y(1); %第一个值
12 %从第二个值到倒数第二个值实现三点线性滑动平均
13 for i = 2:n-1
14     ym(i) = sum(y(i-1:i+1))/3;
15 end
16 ym(n) = y(n); %最后一个值
17
18 %% 可视化
19 plot(t,y,'b',t,ym,'r','LineWidth',2)
20 legend('随机噪声信号','三点线性滑动平均信号')
21 legend('boxoff')
22 title('随机噪声信号与与三点线性滑动平均信号图')
23 xlabel('T'); ylabel('Y'); grid on
24 end
```

>> [y,ym] = random\_signal([0,4\*pi],0.1);



### 3. 循环结构——while语句

**while expression, statements, end** 计算一个表达式，并在该表达式为 true 时在一个循环中重复执行一组语句。表达式的结果非空并且仅包含非零元素（逻辑值或实数值）时，该表达式为 true。否则，表达式为 false。

```

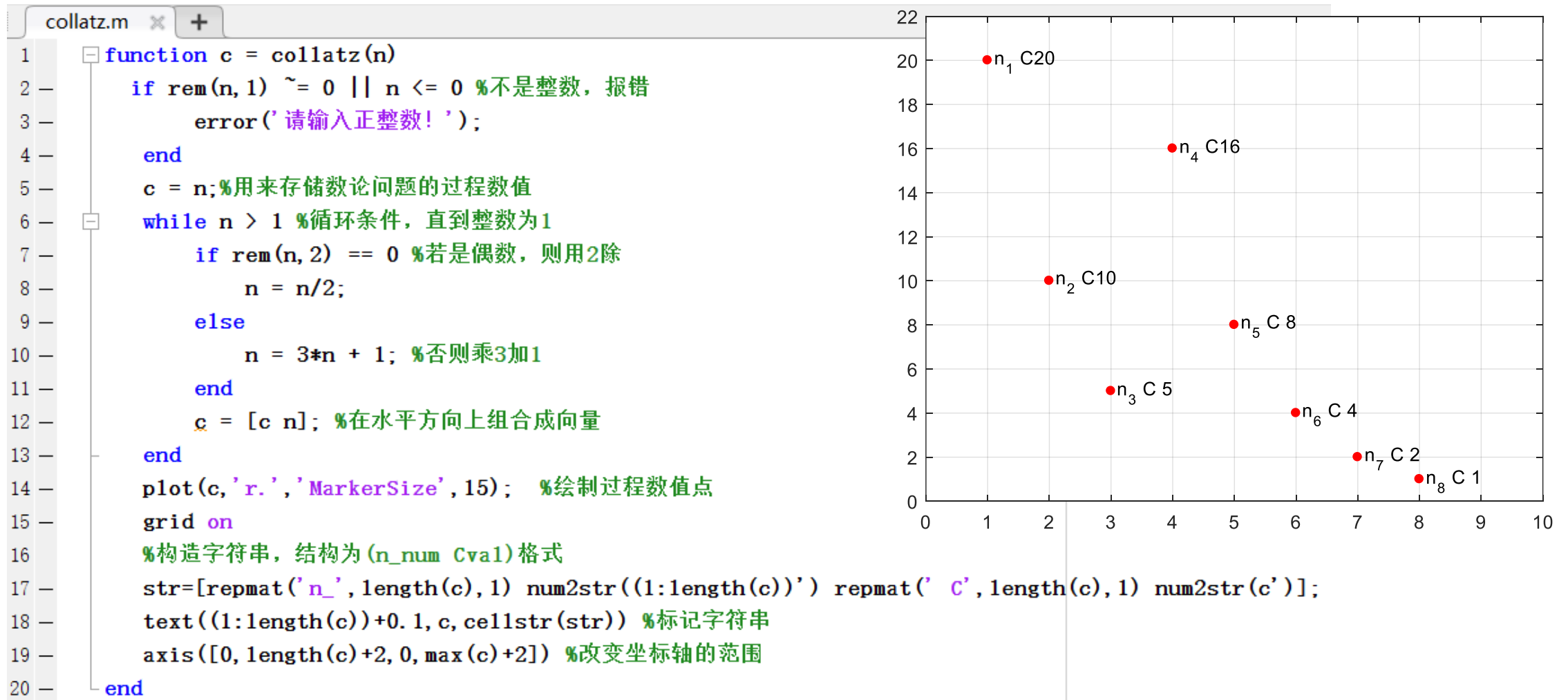
1  fid = fopen('magic.m','r'); %打开文件
2  count = 0; %记录文件行数
3  while ~feof(fid) %检测文件末尾，若为1则到文档末尾
4      line = fgetl(fid); %读取文件中的行，并删除换行符
5      %line是否为空？是否为注释？是否为字符数组？
6      if isempty(line) || strncmp(line,'% ',1) || ~ischar(line)
7          continue %满足条件继续，后续语句不执行
8      end
9      count = count + 1; %行数加一
10 end
11 %返回指定文件的路径名称、文件名和扩展名
12 [filepath,name,ext] = fileparts('magic.m');
13 disp([name,ext,'文件共',num2str(count),'代码。'])

```

- 如果条件表达式的计算结果是一个矩阵，则仅当该矩阵中的所有元素都为 true（非零）时，MATLAB 才会计算这些语句。要在任何元素为 true 时执行语句，请在 any 函数中对表达式换行。
- 要以编程方式退出循环，请使用 break 语句。要跳过循环中的其余指令，并开始下一次迭代，请使用 continue 语句。
- 嵌套许多 while 语句时，每个 while 语句都需要一个 end 关键字。

### 3. 循环结构——while语句

例5：编制一个解数论问题的函数文件：取任意整数，若是偶数，则用2除，否则乘3加1，重复此过程，直到整数变为1。



### 3. 循环结构——break、continue和return语句

与循环结构相关的语句还有break语句和continue语句。它们一般与if语句配合使用。

- break语句用于终止循环的执行。当在循环体内执行到该语句时，程序将跳出循环，继续执行循环语句的下一语句。
- continue语句控制跳过循环体中的某些语句。当在循环体内执行到该语句时，程序将跳过循环体中所有剩下的语句，继续下一次循环。
- return就是直接退出程序或函数返回了
- 大概的关系如下：return > break > continue

### 3. 循环结构——break、continue和return语句



```
firstrem.m  +
1  function k = firstrem(span,n)
2  % 求解指定范围内第二个能被n整除的整数
3
4  %% 循环判断输入参数n是否为整数，否则，重新输入！
5  while true
6      if rem(n,1) ~= 0 %判断n是否为整数
7          n = input(' 输入的n必须是正整数！请重新输入n = ');
8      else
9          break %跳出循环
10     end
11 end
12
13 %% 判断区间的起点是否是整数，以便后续用步长1枚举整数
14 if mod(span(1),1) ~= 0
15     disp(' 区间起点不是整数!')
16     k = 0; %防止函数调用中存在输出参数
17     return %后续代码不执行
18 end
19
20 %% 计算第二个被n整除的整数
21 start = span(1) + n;
22 for k = start:span(2) %枚举测试
23     if rem(k,n) ~= 0
24         continue %不被整数，则继续
25     end
26     break %跳出循环
27 end
28 end
```

例6：求指定范围内第二个能被n整除的整数。

```
>> k = firstrem([400,1000],321.5)
```

输入的n必须是正整数！请重新输入n = 321

```
k =
```

```
963
```

```
>> k = firstrem([400.5,1000],321)
```

区间起点不是整数！

```
k =
```

```
0
```

```
>> k = firstrem([400,1000],321)
```

```
k =
```

```
963
```



---

# 感谢聆听

---