



- Matlab与Python的对比
- Matlab的版本选择与安装

- 认识Matlab基本界面
- Matlab基本语法
- Matlab基础绘图

- 矩阵运算
- 递推计算
- 数值计算

- 线性回归
- 非线性规划模型与启发式算法求解

- 线性回归
- 非线性规划

## 6 总结与答疑

# Matlab简介

Matlab（矩阵实验室）是由美国MathWorks公司开发的一种高级技术计算语言和交互式环境，广泛应用于科学计算、数据分析、算法开发和可视化等领域。

# 选Matlab还是Python

## Matlab

- 内置数学计算函数和工具箱、交互式工作区、工程领域标准。不需要会环境配置，安装好后开箱即用
- 文档齐全，内置帮助文档（按F12即可查看函数详细用法），相比去问AI会更加精确且全面
- 付费软件，昂贵的授权费，每个工具箱还需要单独付费<sup>a</sup>，很多学校没有资金订阅

<sup>a</sup>其实我们学校之前是有订阅过的，但是嘛.....

## Python

- 有numpy、sklearn、matplotlib等科学计算库、jupyter notebook交互式工作区，但是需要掌握环境管理/虚拟环境（anaconda、uv、venv等）
- 教程齐全。作为一门热门语言，网络上教程很多，问LLM也能得到不错的答案
- 免费软件，免费、开源、社区活跃，每个人都能自由的使用（freeware）

# 为什么建议数模初学者用Matlab

- Matlab 是工程领域的计算器+画图板+方程求解器——5分钟从数据到图表，无需折腾环境配置。安装完立即算矩阵、画图、解方程，让大家专注算法而非工具；
- 而 Python 需要先学虚拟环境管理，对完全零基础的同学是额外负担。
- 装 Python 的编程环境常见的环境问题（依赖冲突、版本不兼容）往往劝退初学者。

# Matlab版本选择

Matlab相邻版本之间差别并不大。且近年更新的高级功能做数模一般用不上，所以只要不用过于老的版本（推荐使用  $\geq 2016b$ ），语法都是通用的。

**ab**尾缀的含义：

- **a**: 上半年发布的版本，通常在3月份发布
- **b**: 下半年发布的版本，通常在9月份发布

通常来说，一个a版本发布后，经过几个月的使用和反馈，MathWorks会修复一些Bug，并在同年秋季的b版本中使其更加稳定。所以推荐使用**b**尾缀的版本。

接下来的教程是基于Matlab R2024b的，如果你的版本不是2024b也没有关系，操作完全相同。

# Matlab安装演示

自行下载Matlab，网盘链接在钉钉群中  
如果下载速度过慢，可以带u盘下课找我拷贝  
安装方法：参考公众号安装教程

双击Matlab图标，等待一会儿，就会出现白色的Matlab软件窗口。

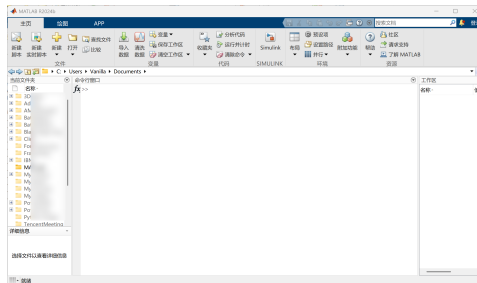


图: Matlab R2024b 界面示例



## 认识Matlab基本界面

## Matlab基本界面介绍

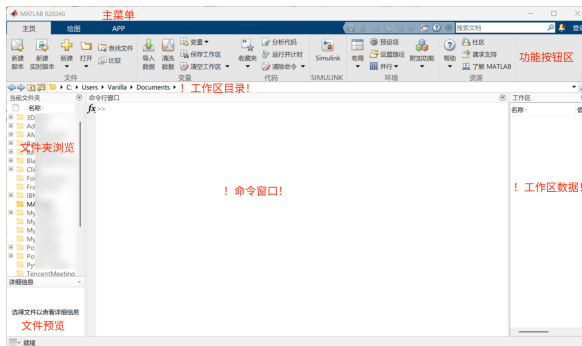


图: Matlab R2024b 界面示例

# Matlab工作区

工作区，顾名思义就是进行工作的地方。建议先更换工作区到准备好的地方，做好文件管理，不要到处都散乱着代码文件。

通常，初次启动Matlab时，默认路径可能是C:\Users\用户名\Documents\MATLAB，建议更改为自己准备好的工作目录。

个人习惯在比赛目录下新建一个code或者workspace文件夹专门用来存放代码

# 更换工作区目录

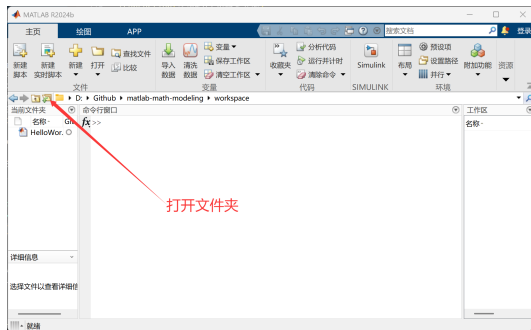


图: 更换当前工作目录

# 命令行窗口

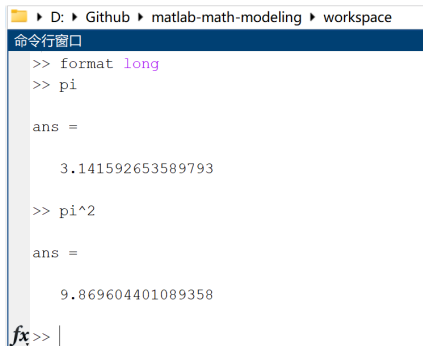
命令行窗口的左上角会有三个大于号：>>，表示等待用户输入命令。

命令行窗口是 **Matlab** 最常用的窗口，用于输入命令、执行命令、查看结果。

可以直接在命令行窗口中写一些简单的代码，也可以把它当作一个**全能计算器**。

```
>> format long % set long format
>> pi
ans =
    3.141592653589793
>> pi^2
ans =
    9.869604401089358
>>
```

# 命令行窗口



```

D:\Github\matlab-math-modeling\workspace
命令行窗口
>> format long
>> pi

ans =

    3.141592653589793

>> pi^2

ans =

    9.869604401089358

fx>> |
```

图: 命令行窗口

# 文件命名规范

MATLAB 文件名必须以字母开头，最多包含 63 个字母数字字符或下划线。

也就是说：

- 不能以数字开头
- 不能包含中文
- 不能用减号 -，应使用下划线 \_

养成良好的文件命名习惯，可以帮助我们在数模比赛中快速的找到文件。在数模比赛中，由于会有很多道子问题，所以个人推荐这样命名，既规整又有含义：

- `problem1_GA.m` — 问题一，使用遗传算法
- `problem2_draw.m` — 问题二，画图相关代码

新建 HelloWorld.m 文件，写入如下内容，然后用编辑器的运行按钮或命令行运行。

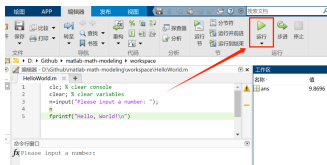
```
clc; % clear console
clear; % clear variables
n=input("Please input a number: ");
n
fprintf("Hello, World!\n")
```

### 常见运行方式:

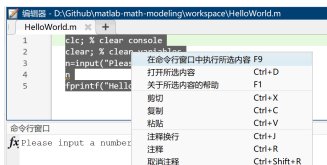
- 1 上方的“运行”按钮
- 2 选中代码，按 F9（在命令行中执行所选）
- 3 在行号左侧点击出现的运行蓝条

## 认识Matlab基本界面

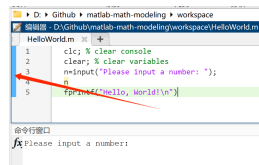
## 运行代码示例



(a) 运行按钮



(b) 选中代码按F9/右键运行



(c) 左侧点击运行（运行段）

运行后命令行会显示“Please input a number: ”，  
输入一个数字并回车后，会输出你输入的数字并打印Hello, World!。



# 从矩阵开始

Matlab 的名字是“矩阵实验室”，内部数据以矩阵为主，直接面向矩阵进行运算。

文件: Section2\_1\_matrix.m

```
clc;clear; % 清空控制台和变量
```

```
A=[1,2,3;4,5,6;7,8,9] % 创建矩阵A，是一个3行3列的矩阵
```

```
% [] 表示构建一个矩阵
```

```
% 逗号或空格用来分隔同一行不同列的元素
```

```
% 分号用来分隔不同行（同一列）的元素，或者说表示换行
```

```
A(1:2,:) % 拆：取A的前两行
```

```
% 矩阵后面连着一个括号，代表取矩阵的某个部分
```

```
% 第一个参数 1:2 表示“从1到2”，即第1行和第2行
```

```
% 第二个参数 : 表示“所有”
```

```
% 所以这里取的是A的前两行，所有列
```

```
B=[A;10,11,12] % 添：在A的下面添加一行新数据，构成新矩阵B
```

```
% 把新行 [10 11 12] 用分号和A连接，表示把新行添加到A的下面
```

# 向量是特殊的矩阵

向量是特殊的矩阵，可以看作是只有一行或者一列的矩阵。

同理，标量可以看作是1行1列的矩阵。这也是为什么说Matlab中万物皆矩阵

文件: Section2\_2\_vector.m

```
clc;clear
```

```
x = [1 2 3 4 5]           % 一行五列→ 行向量  
y = [1; 2; 3; 4; 5]       % 五行一列→ 列向量  
a = [1,2,3,4,5,6,7,8,9]   % 手写会很累  
b = 1:1:10                % 从1开始，步长1，到10结束→ [1 2 3 4 5 6 7 8 9 10]  
d = 0:2:20                 % 从0开始，步长2，到20结束→ [0 2 4 6 8 10 12 14 16 18 20]  
e = 10:-1:1               % 从10开始，步长-1，到1结束→ [10 9 8 7 6 5 4 3 2 1]
```

步长可正可负，会构造一个等差数列。可以用这种方法快速的构造等间隔的数轴坐标、时间序列等。

```
clc;clear
```

```
a=[1,2,3,4,5,6,7,8,9] % 创建一个向量
```

```
aa=a' % 加'是转置, 行→列
```

**b=[1,2,3;4,5,6;7,8,9] % 创建一个矩阵**

```
c=1:1:10 % 创建一个从1到10的向量
```

```
e=eye(4) % 生成4维 (4*4) 的单位矩阵I (对角线为1)
```

```
z=zeros(1,4) % 生成1行4列的全零矩阵
```

```
o=ones(4,1) % 生成4行1列的全1矩阵
```

% 随机矩阵

```
r=rand(4) % 生成4*4的0-1范围内的随机矩阵
```

```
rn=randn(4) % 生成4*4的均值为0，方差为1的正态分布随机矩阵
```

```
ri=randi([1,10],2,4) % 生成2*4的随机整数矩阵 (1到10之间)
```

```
% 对角矩阵
```

```
d=diag([1,2,3]) % 对角线上是1,2,3的3×3矩阵
```

```
d=diag(b)
```

# 常用的矩阵创建方法

文件: Section2\_3\_create.m

% 三角矩阵

**U=triu(b)** % 上三角矩阵 (下三角变为0)

**L=tril(b)** % 下三角矩阵 (上三角变为0)

% 生成相同维度的矩阵

**size(b)** % 会输出b的尺寸 [3 3]

**f=zeros(size(b))** % 生成一个和b矩阵尺寸一样的矩阵

% **f=zeros(size(b,1),size(b,2))** 和 **f=zeros(height(b),width(b))** 同理

% 重复矩阵

**R= repmat([1 2],2,3)** % 把 [1 2] 重复成2行3列块

% 常用序列

**x=linspace(0,10,100)** % 0到10之间等距100个点, 无需手动设置步长

**y=logspace(1,3,5)** %  $10^1$ 到 $10^3$ 之间对数间隔5个点

# 变量名和ans

在刚刚的例子中，我们创建了很多的变量，比如R，x，y等等。

在Matlab中，变量名可以用字母和数字，但是不能以数字开头。

如果某一行只是输出结果，但没有给任何变量赋值，那么结果会自动存储在一个名为ans的变量中。

文件: Section2\_4\_variable.m

```
clc;clear;
```

```
3+5           % 没有变量名→ 自动存到  ans
```

```
ans           % 输出ans查看结果→ 8
```

```
score=95      % 有名字→ 存到  score
```

```
score         % 查看→ 95
```

```
ans           % 此时ans未被修改，还是8
```

# 分号与常用输出方式

在之前的示例中，可以发现：

- 如果行末不存在分号，那么Matlab会自动输出该行的结果；
- 如果行末有分号，则不会输出结果。

文件：Section2\_5\_output.m

```
clc;clear;
```

```
% 使用分号选择性的输出运算结果
```

```
A=[1,2];B=[3,4]; % 不展示A和B
```

```
C=A+B           % 展示C（会输出矩阵名）
```

# 分号与常用输出方式

使用“无分号”来输出变量，输出的格式较为混乱且不可控（好处是啥都能展示）。在实际的运算中，我们通常会希望仅部分结果，或者按照某种格式输出结果。

下面介绍几种更优雅的输出方式：

- disp() 函数
- disp() 与num2str() 函数结合
- fprintf() 函数

# 分号与常用输出方式

## 1. disp() 函数

**disp**可以说是 MATLAB中的最常用的输出方式。简单直接，适合快速查看变量内容。相比于“无分号”直接输出，**disp**不会输出变量名，只会输出变量的值，更加简洁。

文件: Section2\_5\_output.m

```
% 1. disp
disp(A);           % 打印矩阵（不会输出矩阵名）
disp(pi);          % 打印标量（不会输出变量名）
disp("Ciallo~");   % 打印字符串
% 简单直接，适合快速查看
% 不能混合输出文字和数字
```



# 分号与常用输出方式

## 2. disp() 与 num2str() 函数结合

disp() 函数也可以和 num2str() 函数结合使用，将数字转为字符串再输出。

文件: Section2\_5\_output.m

```
% 2. num2str
```

```
x = 3.14;
```

```
disp(['The value of pi is: ' num2str(x)]) % 注意各个字符串之间需要空格/逗号
```

```
% 本质上是字符数组的拼接
```

将会输出: The value of pi is: 3.14

# 分号与常用输出方式

## 3. fprintf() 函数

fprintf函数是Matlab中功能最强大的输出函数，类似于C语言中的printf函数。

其整体的结构是：fprintf('格式字符串'，变量1，变量2，...)。

常用的格式符有：%d、%f、%s、%c、%e、%g、%%等。

比如：%.2f表示输出浮点数并保留两位小数。%d表示输出整数。

文件：Section2\_5\_output.m

```
% 3. fprintf
```

```
x = 3.1415926535;
```

```
fprintf('The value of pi rounded to two decimal places is: %.2f\n', x)
```

```
fprintf('Today''s temperature is: %d\n', 25) % 需要加\n, fprintf不会自动换行
```

# 矩阵的基本运算

MATLAB的核心优势就是矩阵运算。掌握这些运算符，你就能像处理数字一样轻松而批量快速的处理矩阵。

文件: Section2\_6\_operation.m

```
A = [1 2; 3 4];  
A_T = A';      % 加一个单引号，
```

# 矩阵的基本运算

矩阵加减法：对应元素相加减，要求矩阵维度相同。

文件：Section2\_6\_operation.m

```
A = [1 2; 3 4];
B = [5 6; 7 8];
C = A + B      % [6 8; 10 12]
D = A - B      % [-4 -4; -4 -4]
```

矩阵乘法和矩阵点乘：矩乘要求矩阵A的列数等于矩阵B的行数；点乘要求矩阵维度相同，对应元素相乘。

文件：Section2\_6\_operation.m

```
A = [1 2; 3 4];
B = [5 6; 7 8];
% 矩阵乘法（线代）
C = A * B      % 2x2 * 2x2 = 2x2
% 点乘（对应元素相乘）
D = A .* B      % [1*5 2*6; 3*7 4*8] = [5 12; 21 32]
```

# 矩阵的基本运算

## 矩阵除法和点除

文件: Sectione2\_6\_operation.m

```
A = [2 4; 6 8];
B = [1 2; 3 4];
C = A / B      % 矩阵右除 (等价于 A * inv(B))
D = A ./ B     % 点除 [2/1 4/2; 6/3 8/4] = [2 2; 2 2]
% 向左倾斜的是左除, A\B等价于inv(A) * B。
```

## 矩阵幂和点幂

文件: Sectione2\_6\_operation.m

```
A = [1 2; 3 4];
B = A^2        % A * A (矩阵乘法)
C = A.^2       % [1^2 2^2; 3^2 4^2] = [1 4; 9 16]
```

总结: 矩阵运算符前加点 (.\*, ./, .^ ) 表示对应元素操作。否则则为矩阵操作。如果出现运算的报错, 优先检查是不是忘记加点了

# 矩阵的基本运算

## 矩阵元素的访问与修改

文件: Section2\_6\_operation.m

% matrix(row,col) : 访问row行, col列的元素, 修改可直接加上赋值号。

% matrix(num) : 按照顺序访问, 特别注意, matlab一列一列的访问, 与其它语言不同。

% matrix(row,:) : 访问第row行的所有元素。

% matrix(row,:)= [1,2,.....] : 修改第row行, 直接赋值覆盖原值。

% matrix(row,:)= [] : 删除第row行。

% matrix(:,col) : 表示第col列的相关操作, 和行一致。

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

```
[matrix(1) matrix(2) matrix(3)] % 发现单参数访问顺序是 1 5 2 6 3 7 5 8
```

```
matrix(2,4) % 第2行第4列
```

```
matrix(1,:) % 第一行, 所有列, 即第一行一整行
```

```
matrix(:,1) % 所有行, 第一列, 即第一列一整列
```

```
matrix(1,1)=100 % 修改第一行第一列为100
```

```
temp=matrix; temp(1,:)=[] % 删除第一行。同时矩阵会从2*4变成1*4
```

```
temp=matrix; temp(:,2)=[] % 删除第二列。同时矩阵会从2*4变成2*3
```

# Matlab中的循环与条件语句

作为一门编程语言，为什么这么晚才讲循环与条件？因为Matlab的核心思想就是“向量化>循环”，能用矩阵运算的就不要用循环。

下面简单介绍循环和条件语句的写法。

文件：Section2\_7\_loop\_condition.m

```
score = 85;
if score >= 90
    disp("优秀")
elseif score >= 60
    disp("及格")
else
    disp("挂科")
end
```

if后直接跟着条件，以end结束。elseif和else可选。

用==表示等于而不是赋值=。此外，有&表示逻辑与，|表示逻辑或，~表示逻辑非。

# Matlab中的循环与条件语句

文件: Section2\_7\_loop\_condition.m

```
sum_val = 0;  
for i = 1:100  
    sum_val = sum_val + i; % 计算1+2+...+100  
end  
disp(sum_val) % 结果: 5050
```

% 错误示范: 在循环中动态扩展数组

```
A = []; % 应该写成 A=zeros(10000,1);  
for i = 1:10000  
    A(i) = i;  
end
```

% 会给警告: 变量的大小似乎在(脚本内的)每个循环迭代都会更改。请考虑进行预分配以提升速度。

特别地, 在for中尽量避免动态地扩展数组。这是因为循环中每次迭代都要重新分配内存, 这会耗费大量时间。



# Matlab中的循环与条件语句

推荐用向量化代替循环。

可以直接用sum函数计算1到100的和，而不需要循环。

对于简单的条件语句，也可以用逻辑索引来代替if。

文件: Section2\_7\_loop\_condition.m

% 判断成绩等级

scores = [85, 92, 45, 78]; % 4个学生的成绩

pass = scores >= 60; % [1, 1, 0, 1] 逻辑向量

disp(scores(pass)) % 直接输出及格分数: [85,92,78]

% 计算1+2+...+100

sum\_val = sum(1:100); % 1行代码 vs 5行循环

Matlab 支持很多形式：匿名函数 ( $f=@(x)x.^2$ )、文件内多函数、嵌套函数、类方法..... 但入门阶段只需要掌握最常用的一种：单文件函数。



```
function s = mySum(a, b) % 直接以function开头，且函数名与文件名完全一致
% 传入了两个参数a和b
s = a + b; % s是函数的返回值，在遇到function的end时会将s作为函数的值返回
end
```

上面定义了一个简单的函数mySum，功能是计算两个数的和。

```
% 需要保证与函数文件mySum.m在同一个目录下
result = mySum(10, 20);
```

函数的三个要点:

- **function** 输出 = 函数名(输入)
- 文件名必须和函数名相同
- 用**end**结束

# Matlab中的函数定义与调用

函数内部可以写多行代码，也可以返回多个变量。

文件: stat2.m

```
function [m, s] = stat2(x)
m = mean(x); % 计算均值
s = std(x); % 计算标准差
end
```

调用:

文件: Section2\_9\_function.m

```
function [m, s] = stat2(x)
[data_mean, data_std] = stat2([1,2,3,4,5])
% 函数也支持向量化运算，可以实现输入向量或矩阵得到批量运算结果。
[datas_mean, datas_std] = stat2([1,2,3;4,5,6;7,8,9])
end
```

函数适合：重复计算、逻辑封装、让脚本更清晰。

# 符号运算

符号运算：让 Matlab 像在纸上做代数、微积分推导，而不是只是计算小数结果。

文件：Section2\_10\_symb.m

```
clc;clear
```

% 1. 定义一个符号变量和函数  $f(x)$

```
syms x % 告诉 Matlab: x 是“符号”
```

```
f = x^3 - 2*x + 1; % 像在高数里写函数
```

% 2. 求导  $f'(x)$

```
df = diff(f) % 结果:  $3 \times x^2 - 2$ 
```

% 3. 求不定积分  $\int f(x)dx$

```
F = int(f) % 结果:  $x^4/4 - x^2 + x$ 
```

% 4. 求定积分  $\int_0^1 f(x)dx$

```
I = int(f, x, 0, 1) % 得到一个精确结果 (分数)
```

% 5. 在  $x=2$  处代入

```
val = subs(f, x, 2) % 把 x 换成 2 代入
```

这块和大家的高等数学 / 数学分析联系很紧：以后遇到复杂积分、推公式，可以先让 Matlab 帮你验算。

# 绘图的作用

绘图是数模与科研中必不可少的一环，可以帮助我们：

- 快速观察数据趋势（如上升、震荡、周期性）；
- 验证模型是否合理；
- 制作论文与报告中的结果图像。

Matlab 中最简单的绘图方式：只给  $y$ ，默认横轴为 1,2,3,...

文件：Section2\_11\_plot.m

```
clc;clear;close all; % close用来关闭之前生成的绘图窗口  
y = [2, 4, 3, 5, 6, 4];  
plot(y); % 本质上是默认 x = 1:length(y)
```

只需要简单的一个`plot(y)`，就能画出数据的折线图。

# plot(x, y): 绘制自变量与因变量

更常用的绘图方式：指定  $x$  和  $y$ 。

文件: Section2\_11\_plot.m

```
clc;clear;close all;  
x = 0:0.1:2*pi;      % 自变量  
y = sin(x);          % 因变量  
plot(x, y);           % 绘制 y = sin(x)
```

要点:

- $x$  与  $y$  必须长度一致;
- $x$  可以是任意数值 (时间、空间、数据点);
- 适合用来绘制函数曲线或实验数据。

一张好的图要能说明问题，展示数据的含义。因此在图中，做好各处标注很重要。

文件: Section2\_11\_plot.m

```
clc;clear;close all;
x = 0:0.1:2*pi;
y = sin(x).^2;
plot(x, y)
title("y = sin(x)^2")           % 标题
xlabel("x (rad)")               % x轴标签
ylabel("y")                     % y轴标签
legend("sin(x)^2")              % 图例（说明线条含义）
```

特别提醒一下，画图模块的字符串大多都是支持LaTeX的，比如可以用 $\sin(x)^2$ 来表示数学公式。但是有时候也会触发奇怪的错误，导致标签显示不正常。



# 图像窗口控制：figure

Matlab 默认在同一个窗口中绘图，如果想创建新的图像窗口，需要使用 **figure**。

文件：Section2\_11\_plot.m

```
clc;clear;close all;
```

```
x = 0:0.1:2*pi;
```

```
figure;           % 新建一个绘图窗口
```

```
plot(x, sin(x))
```

```
title("这是第一张图")
```

```
figure;           % 再打开一个新窗口
```

```
plot(x, cos(x))
```

```
title("这是第二张图")
```

# 在同一张图中绘制多条曲线

默认情况下，新的plot会覆盖旧图。可以使用 `hold on` 可以在同一张图中添加多条线。

文件: Section2\_11\_plot.m

```
clc;clear;close all;
```

```
x = 0:0.1:2*pi;
```

```
hold on           % 开始叠加
```

```
plot(x, sin(x))
```

```
plot(x, cos(x))
```

```
hold off          % 结束叠加
```

```
legend("sin(x)", "cos(x)")
```

在数模中常用于：

- 比较真实数据 vs 模型预测数据
- 比较不同参数的模型结果
- 展示多个方案的曲线

# 线型、颜色与标记控制

美感也是论文追求的目标之一。Matlab 支持丰富的绘图样式，可用于论文美化。

文件: Section2\_11\_plot.m

```
clc;clear;close all;
```

```
x = 0:0.1:2*pi;
```

```
plot(x, sin(x), 'r--', 'LineWidth', 1.5) % 红色、虚线、加粗
```

```
hold on
```

```
plot(x, cos(x), 'b-o', 'MarkerSize', 4) % 蓝色、圆点标记
```

```
hold off
```

```
legend("sin(x)", "cos(x)")
```

plot函数有很多的样式控制方法，具体可以右键plot，打开关于plot函数的帮助，查看官方文档。常见样式参数：

- 颜色: 'r'红, 'b'蓝, 'g'绿, 'k'黑...
- 线型: '-'实线, '--'虚线, ':'点线
- 标记: 'o'圆点, '\*'星, 's'方块

# 问题1

1. 请用分别一行代码，生成以下矩阵：

- 3行4列的随机矩阵
- 3行10列的矩阵，元素为1到30的连续整数
- 下三角部分为[0,1]上是均匀随机数，其余为全1的5行5列矩阵

（提示：请使用help自学reshape函数）

2. 已知 $x=[1 \ 2 \ 3]$ ； $y=[4 \ 5 \ 6]$ ；，请用一行代码计算两个向量的点积（提示：元素相乘再求和）

3. 已知 $A=[3, -1, 4, -2, 5]$ ；，请用一行代码把A中负数变成0，正数不变（提示：用逻辑索引）

# 问题1答案

文件: Section3\_1\_ans.m

% 3行4列的随机矩阵

A = rand(3, 4)

% 3行10列的矩阵，元素为1到30的连续整数

B = reshape(1:30, 3, 10)

% 下三角部分为[0,1]上是均匀随机数，其余为全1的5行5列矩阵

C = ones(5) + tril(rand(5)-1)

% 计算两个向量的点积

x = [1 2 3]; y = [4 5 6];

sum(x .\* y)

% 负数变0，正数不变

A = [3, -1, 4, -2, 5];

A(A<0) = 0

## 问题2

计算斐波那契数列第 $N$ 项的最低4位数，即：

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$$

求  $F_N \bmod p$

其中， $N = 40$ ， $p = 10^4$

**Tips:** 可以尝试用“递推+for循环”和“递归+函数”两种写法实现

如果 $N = 10^8$ 呢？如果 $N = 10^{18}$ 呢？同时看看，你的程序需要多长时间计算出结果？

推荐阅读：oi-wiki: 斐波那契数列，zhihu: Pisano 周期

## 问题2答案

文件: Section3\_2\_fib.m

```
clc;clear;
```

```
% 计算斐波那契数列第N项模p的值
```

```
N=40;
```

```
p=10000;
```

```
% 递推关系:  $F(i) = F(i-1) + F(i-2)$ 
```

```
tic
```

```
A1 = 1; % F(1), 即前一项 -> 在迭代过程中代表F(i-1)
```

```
A2 = 1; % F(2), 即当前项 -> 在迭代过程中代表F(i)
```

```
for i=3:N % 然后这里只需要从第3项开始迭代, 每次迭代生成第i项
```

```
    t=A2; % 临时存储一下当前项A2, 其在下一轮会变成前一项A1
```

```
    A2=mod(A1+A2,p); % 递推出新的一项, 并取模防止溢出
```

```
    A1=t; % 将原来的当前项作为下一轮的前一项
```

```
end
```

```
result=A2; % F(N)
```

```
disp(result);
```

```
toc
```

# 问题3

下面给出一个函数：

$$f(x) = x^2 e^{-x}$$

请完成以下任务（每题都要求用一行代码完成）：

- (1) 求  $f(x)$  的导数  $f'(x)$
- (2) 计算定积分  $\int_0^3 f(x) dx$
- (3) 在  $x = 2$  处，求  $f(2)$  的数值
- (4) 解方程  $f(x) = 0.1$ ，求正根（提示：使用 `vpasolve` 或 `fzero`）
- (5) 画图验证(4)的正确性



# 问题3答案

文件: Section3\_3\_symb.m

```
clc;clear;
```

```
% 定义函数
```

```
syms x
```

```
f = x.^2 * exp(-x)
```

```
%求  $f(x)$  的导数  $f'(x)$ 
```

```
df = diff(f)
```

```
% 计算定积分  $\int_0^3 f(x) dx$ 
```

```
int(f, x, 0, 3)
```

```
% 在  $x=2$  处, 求  $f(2)$  的数值
```

```
subs(f, x, 2)
```

# 问题3答案

文件: Section3\_3\_symb.m

% 解方程  $f(x) = 0.1$ , 求正根

**format** long

r1 = **vpasolve**(f == 0.1, x, 1)      % 从 1 附近找一个解

r2 = **fzero**(@g, 1)      % 注意: g 里已经减掉 0.1

% 画图验证

**figure**

xx = 0:0.01:5;

yy = xx.^2 .\* **exp**(-xx);

**hold** on

**plot**(xx, yy)      % 曲线  $y = f(x)$

**plot**(xx, 0.1\*ones(size(xx)))      % 水平线  $y = 0.1$

**plot**(double(r1), double(subs(f,x,r1)), 'ro')      % r1 对应的交点

**hold** off

**function** y = g(x) % 函数也可以定义在文件的末尾

y = x.^2 .\* **exp**(-x) - 0.1;

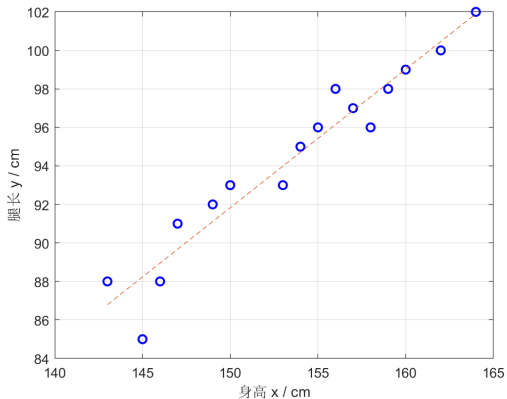
**end**

身高	143	145	146	147	149	150	153	154	155	156	157	158	159	160	162	164
腿长	88	85	88	91	92	93	93	95	96	98	97	96	98	99	100	102

能不能让电脑帮我们找一条最接近所有点的直线呢？

```
clc; clear;
x = [143 145 146 147 149 150 153 154 155 156 157 158 159 160 162 164];
y = [ 88  85  88  91  92  93  93  95  96  98  97  96  98  99 100 102];
figure;
plot(x, y, 'bo', 'MarkerSize',6, 'LineWidth',1.5);
grid on; % 打开背景的网格，看的更清楚
```

# 最接近所有数据点的直线



# 一元线性回归模型

观察散点图后，我们希望用一条“直线”去近似这种关系：

$$y = \beta_0 + \beta_1 x + \varepsilon$$

其中：

- $\beta_0$ ：截距（当身高为0时腿长的理论值，在本例中主要是数学意义）；
- $\beta_1$ ：斜率，表示身高每增加1cm，腿长平均增加多少；
- $\varepsilon$ ：误差项，表示“个体差异”“测量误差”等噪声。

一元线性回归的目标：找到一组 $\hat{\beta}_0, \hat{\beta}_1$ ，使得这条直线“尽可能贴近所有点”。

# 最小二乘

如何衡量“一条直线是否接近所有点”？

对于第  $i$  个样本，有真实值  $y_i$ ，预测值  $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ ，它们之间的“差”称为残差：

$$e_i = y_i - \hat{y}_i$$

最小二乘法（Least Squares）的思想：

$$\text{选 } \hat{\beta}_0, \hat{\beta}_1 \text{ 使得 } \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \text{ 最小.}$$

不做详细推导，只需要知道：最小二乘法给出了一条“总体误差平方和最小”的直线，这就是 Matlab 中 `regress` 命令所做的事。

# regress命令

我们可以把一元线性回归写成矩阵形式：

$$Y = X\beta, \quad X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

在 Matlab 中，只需几行代码就可以完成回归分析：

文件：Section4\_2\_regress.m

```
[b, bint, r, rint, stats] = regress(y, X);
```

其中 **b** 给出  $\hat{\beta}_0, \hat{\beta}_1$  的估计；**stats** 中包含拟合优度  $R^2$ 、F 值以及 p 值。

# regress命令

在 Matlab 中，只需几行代码就可以完成回归分析：

文件：Section4\_2\_regress.m

% 显示结果

```
fprintf("截距beta0 = %.2f\n", b(1));  
fprintf("自变量beta1 = %.2f\n", b(2));  
fprintf("残差向量 r\n"); disp(r);  
fprintf("残差区间 rint\n"); disp(rint); % 若区间跨过0，则该点可能为异常点。  
% 回归总体统计量 stats  
fprintf("R^2 (拟合优度) = %.4f\n", stats(1));  
fprintf("F 统计量 = %.4f\n", stats(2));  
fprintf("p 值 = %.4e\n", stats(3));
```



# 回归结果的解释

运行上一页的代码，得到的结果为：

$$\hat{\beta}_0 \approx -16.07, \quad \hat{\beta}_1 \approx 0.72, \quad R^2 \approx 0.9282.$$

可以这样理解：

- 斜率  $\hat{\beta}_1 \approx 0.72$ ：身高每增加 1cm，腿长平均增加约 0.7cm；
- 截距  $\hat{\beta}_0 \approx -16$ ：在本例中主要是数学量，不必过度解读其现实意义；
- $R^2 \approx 0.9282$ ：说明直线解释了约 92.82% 的  $y$  变化，拟合效果非常好。

此外，`bint` 给出回归系数的置信区间。`stats(3)` 中的  $p$  值若小于 0.05，说明“身高对腿长的线性影响”在统计上是显著的。

得到回归系数后，就可以计算每个样本点的预测值  $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ ，并画出拟合直线：

文件: Section4\_3\_plot.m

```
clc; clear;
```

```
x = [143 145 146 147 149 150 153 154 155 156 157 158 159 160 162 164]'; % 注意需要先转置，变成列向量
```

```
y = [ 88  85  88  91  92  93  93  95  96  98  97  96  98  99 100 102]';
```

```
X = [ones(length(x),1), x];
```

```
b = regress(y, X);
```

```
y_hat = X * b; % 预测值
```

```
figure;
```

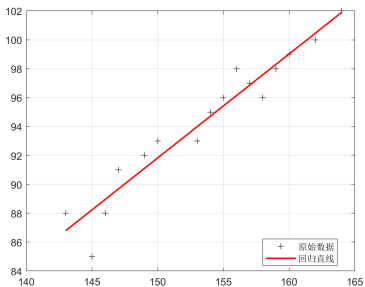
```
plot(x, y, 'k+', 'MarkerSize',6); hold on; % 原始数据
```

```
plot(x, y_hat, 'r-', 'LineWidth',1.5); % 回归直线
```

```
legend("原始数据","回归直线","Location","best");
```

```
grid on;
```

# 预测与拟合直线的绘制



在建模与论文写作中，这种“散点 + 拟合直线”的图是非常常见的展示方式。

# 多元线性回归

刚才我们只考虑了一个自变量（身高）。如果有多个影响因素，例如：

- $x_1$ : 身高,  $x_2$ : 体重,  $x_3$ : 年龄

那么就得到多元线性回归模型：

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \varepsilon.$$

在 Matlab 中，只需要把  $X$  矩阵从两列扩展为多列，其余用法完全相同——调用 `regress` 命令即可得到结果。

文件: Section4\_4\_multi.m

```
clc; clear;
```

```
% 数据在这里忽略
```

```
X = [ones(length(x1),1), x1, x2, x3];
```

```
[b, ~, ~, ~, stats] = regress(y, X)
```

# 线性回归小结

本节学习了：

- 一元线性回归模型的基本形式与含义
- 最小二乘法思想
- 如何使用 `regress` 命令完成回归分析
- 解释回归结果并画出拟合直线
- 拓展多元线性回归

推荐自学：

- 线性回归求解本质：用矩阵除法也可以解决线性回归问题
- 评估回归效果： $R^2$  的含义、F检验、t检验、p值、置信区间、残差分析
- 预测方式：点预测，区间预测
- `stepwise` 逐步回归，`polyfit` `polyval` 多项式回归，`rstools` 多元二项式回归，`nlinfit` 非线性回归

## 非线性规划模型与启发式算法求解

文件：

## 非线性规划模型与启发式算法求解

文件：

文件：



文件：

# 总结

- 1 Matlab介绍
  - Matlab与Python的对比
  - Matlab的版本选择与安装
- 2 Matlab的基础使用
  - 认识Matlab基本界面
  - Matlab基本语法
  - Matlab基础绘图
- 3 Matlab简单应用
  - 矩阵运算
  - 递推计算
  - 数值计算
- 4 简单数学模型
  - 线性回归
  - 非线性规划模型与启发式算法求解
- 5 课后习题
  - 线性回归
  - 非线性规划
- 6 总结与答疑

# 总结

今天晚上和周六周日，我会在数模5群里面看大家的问题

大家可以把不懂的地方发到群里，我和数模协会的同学会尽量帮大家解答