

Matlab与简单数学模型

Vanilla_Yukirin

Vanilla-Yukirin/matlab-math-modeling

2025 年 11 月 13 日

目录

- 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

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^a，很多学校没有资金订阅

^a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

TODO: 安装Matlab过程截图

Matlab基本界面介绍

双击Matlab图标，等待一会儿，就会出现白色的Matlab软件窗口。

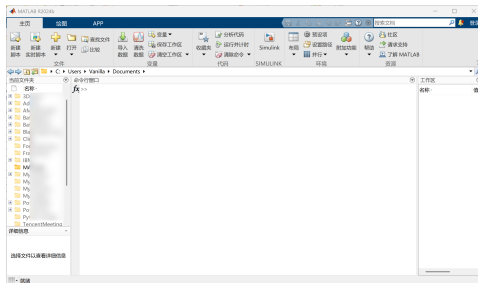


图: Matlab R2024b 界面示例

认识Matlab基本界面

Matlab基本界面介绍

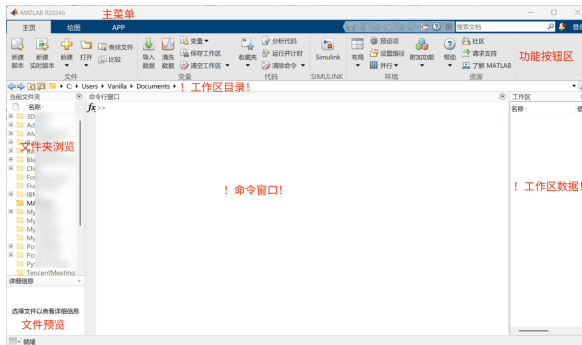


图: Matlab R2024b 界面示例

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

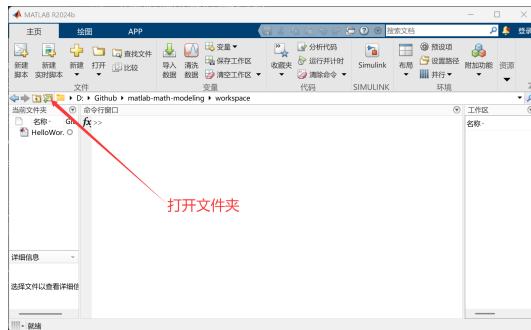


图: 更换当前工作目录

命令行窗口

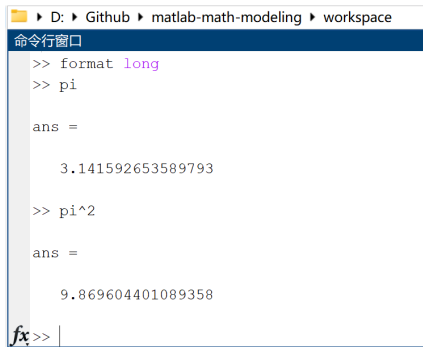
命令行窗口的左上角会有三个大于号：>>，表示等待用户输入命令。

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

运行代码

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。

文件: HelloWorld.m

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

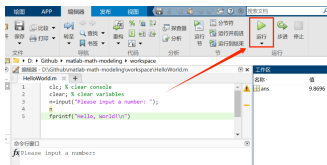
常见运行方式：

- 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

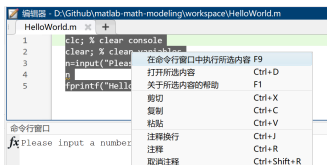


认识Matlab基本界面

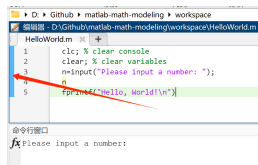
运行代码示例



(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。

常用的矩阵创建方法

文件: Section2_3_create.m

```
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) % 提取b的对角线元素
```

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: Section2_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: Section2_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

受于篇幅原因，只介绍最简单的函数定义方式，可以用于符号计算等场景。





绘图的作用

绘图是数模与科研中必不可少的一环。

绘图可以更方便的观察数据，也可以进行简单的数据处理。模型求解也通常需要用图像进行辅助展示。



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**项模**p**的值

$$F_n = F_{n-1} + F_{n-2} \mod p$$

求 $F_N \mod p$

其中, $N = 10^8$, $p = 10^9 + 7$

同时看看, 你的程序需要多长时间计算出结果?

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=100000000;
```

```
p=1000000007;
```

```
% 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
```



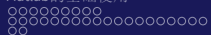
问题3

问题3答案









- 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

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

行内等宽： ...

这里演示行内等宽：`for i = 1:n`，适合短的、简单的代码或变量名。

使用 ...

演示 listings 行内: `disp('hello')`, 适合需要 listings 语法高亮的短代码片段。