

1.1 MATLAB 软件安装

1.1.1 MATLAB 概要

MATLAB 是美国 MathWorks 公司出品的一款商业数学软件,是一种数值计算环境和编程语言。MATLAB 是 matrix laboratory 两个词的组合,意为矩阵工厂(矩阵实验室)。

MATLAB 可以进行矩阵运算、绘制函数和数据、实现算法、创建用户界面、连接其他编程语言的程序等,主要应用于工程计算、信号处理与通讯、图像处理、信号检测、机器人、控制系统、金融建模设计与分析等领域。

1.1.2 MATLAB 的安装

MATLAB 的安装过程比较简单,下面以在 Windows 7 (64 位) 系统下安装 MATLAB 2013a 为例进行介绍,其过程如下。

1) 插入 MATLAB 的安装光盘,启动 setup 文件,显示图 1.1.1 所示的“Mathworks 安装程序”窗口。窗口中包括:标准安装,即使用 Internet 连接安装产品;在不连接 Internet 的情况下进行安装,即使用文件安装密钥安装产品。用户可根据自己的需要自由选择。单击“下一步”按钮继续进行安装。



图 1.1.1 “Mathworks 安装程序”对话框

2) 出现“许可协议”窗口,如图 1.1.2 所示,选择“是”单选按钮,接受软件协议,然后单击“下一步”按钮进行下一步的安装。

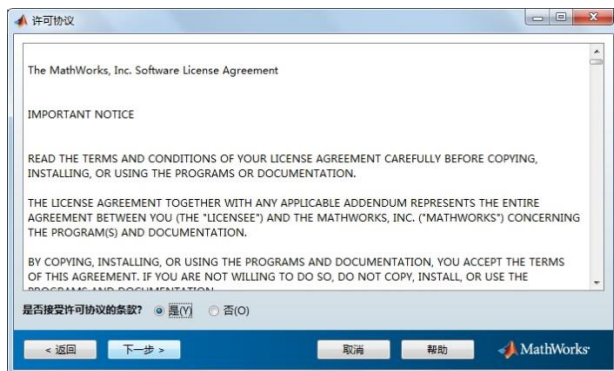


图 1.1.2 “许可协议”窗口

3) 在图 1.1.3 所示的“文件安装密钥”窗口中填写使用许可密码,单击“下一步”按钮进行下一步的安装。



图 1.1.3 “文件安装密钥”窗口

4) 在图 1.1.4 所示的“选择安装类型”窗口中,选择选择“典型”安装选项,系统将按照默认设置,自动安装用户所购买的组件。

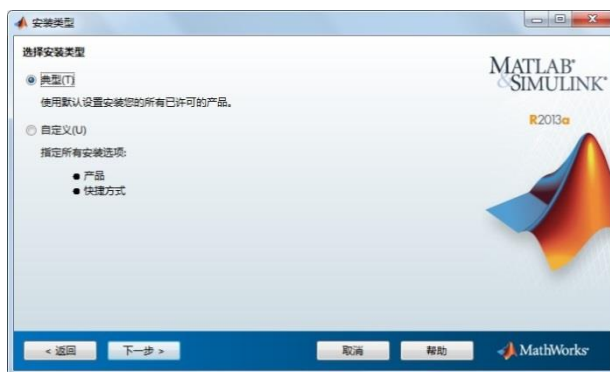


图 1.1.4 “选择安装类型”窗口

5) 弹出图 1.1.5 所示的“选择文件夹”窗口，用户可以单击“浏览”按钮选择安装目录，然后单击“下一步”按钮进行下一步的安装。



图 1.1.5 “选择文件夹”窗口

6) 弹出图 1.1.6 所示的“确认”窗口，单击“安装”按钮确认安装。

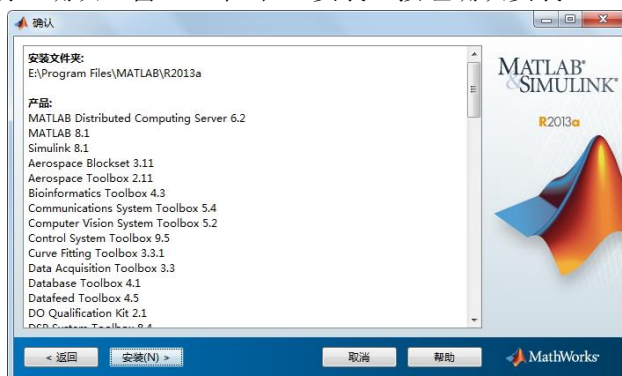


图 1.1.6 “确认”窗口

7) 经过几十分钟的安装过程之后，将弹出“安装完成”窗口，如图 1.1.7 所示，单击“下一步”按钮。

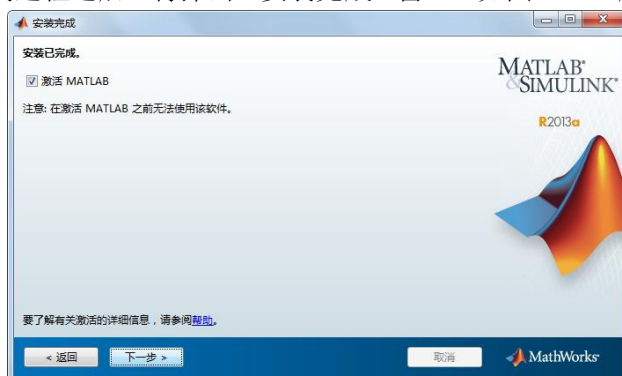


图 1.1.7 “安装完成”窗口

8) 弹出图 1.1.8 所示的“Mathworks 软件激活”窗口，其中有“使用 Internet 自动激活”和“不使用 Internet 手动激活”两个选项。用户可根据自己的软件购买类型进行选择。选择后，单击“下一步”继续。

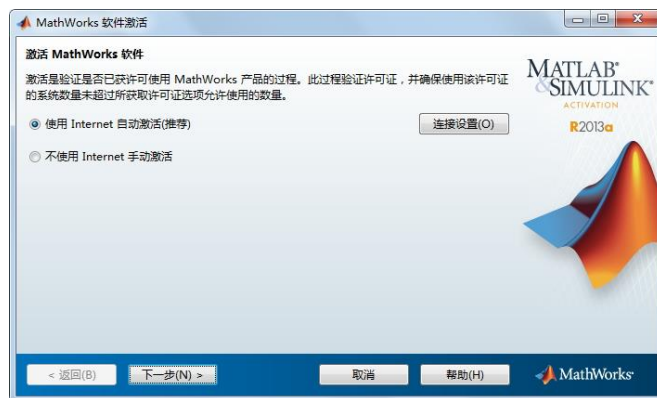


图 1.1.8 “Mathworks 软件激活”窗口

9) 安装完成后出现的界面如图 1.1.9 所示，单击“完成”按钮完成安装。

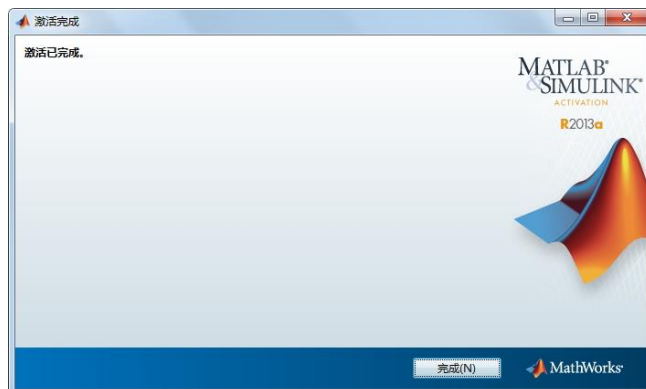


图 1.1.9 “激活完成”窗口

1.2 MATLAB 软件功能简介

1.2.1 MATLAB 的特点

(1) 可视化的开发环境。MATLAB 具有完备的图形处理功能，可以实现计算结果和编程的可视化，包括MATLAB主窗口、文件编辑器、帮助文档等。

(2) 强大的数学计算能力。MATLAB 具有高效的数值计算及符号计算功能，可进行包括基本函数运算、复杂算法、高级矩阵运算等计算功能，特别适合矩阵代数领域的应用，能使用户从繁杂的数学运算分析中解脱出来。

(3) 简单高效的编程语言。MATLAB 友好的用户界面及接近数学表达式的自然化语言，使学者易于学习和掌握。MATLAB 的运算符使得程序简短，且程序书写格式自由，丰富的库函数使得用户可以绕开子程序的编写而直接调用，自定义函数也大大提高了程序设计的自由度。

(4) 强大的图形功能。MATLAB 提供了丰富的绘图函数命令，并且具有较强的编辑图形界面的能力。

(5) 强大的工具箱。MATLAB 功能丰富的应用工具箱为用户提供了大量方便实用的处理工具。MATLAB 工具箱分为功能性工具箱和科学性工具箱两类。

功能性工具箱：主要用于扩充符号计算功能、图示建模仿真功能（如 Simulink）、文字处理功能以及与硬件实时交互功能。

(6) 方便的应用程序接口功能。MATLAB 提供了应用程序接口，可以使用 C、C++ 或 FORTRAN 等其它高级编程语言进行编程，实现与 MATLAB 程序的混合编程。

(7) 设计图形用户界面。MATLAB 可以使用交互式工具 GUIDE（图形用户界面开发环境）布置、设计及编辑用户界面。利用 GUIDE 可以在用户界面中加入列表框、下拉菜单、按钮、单选按钮、滑块、MATLAB 图形、ActiveX 控件和菜单等。此外，用户也可以用编写代码的方式创建GUI。

1.2.2 Simulink 的特点

Simulink 是 MATLAB 的一个工具箱，它主要用来实现对工程问题的模型化及动态仿真，其本身具有良好的图形交互界面，可以提供一个动态系统建模、仿真和综合分析的集成环境。通过采用 Simulink 模块

组合的方法，能够快速、准确地创建动态系统的计算机模型。**Simulink** 的详细特点如下。

(1) 动态系统的建模与仿真。**Simulink** 支持线性、非线性、连续、离散、多变量和混合式系统结构，所以几乎任何一种类型的真实动态系统 **Simulink** 都能胜任。

(2) 建模方式直观。**Simulink** 是一种图形化的仿真工具，利用其可视化的建模方式，可迅速地建立动态系统的框图模型。

(3) 模块可定制。**Simulink** 允许自定义模块的使用，可以对模块的图标、对话框等进行自定义编辑。**Simulink** 也允许将 C、C++、FORTRAN 代码直接移植到 **Simulink** 模型当中。

(4) 仿真模拟快速、精确。**Simulink** 先进的求解器提高了非线性系统仿真的精度，能确保连续系统或离散系统的仿真高速、精确地进行。

(5) 复杂系统的层次性。**Simulink** 利用子系统模块，使得庞杂的系统模型构建变得简单易行。整个系统可以按照自上而下或自下而上的方式进行分层构建，子系统的嵌套使用不受限制。

(6) 仿真分析的交互性。**Simulink** 提供示波器等观察器，用于对动画或图形的显示。仿真过程中，利用这些观察器可以监视仿真结果。这种交互式特性能让开发者快速进行算法评估以及参数优化。

1.2.3 Desktop 操作界面简介

进入MATLAB，首先看到的是 MATLAB 的 Desktop 操作界面，如图1.2.1所示。

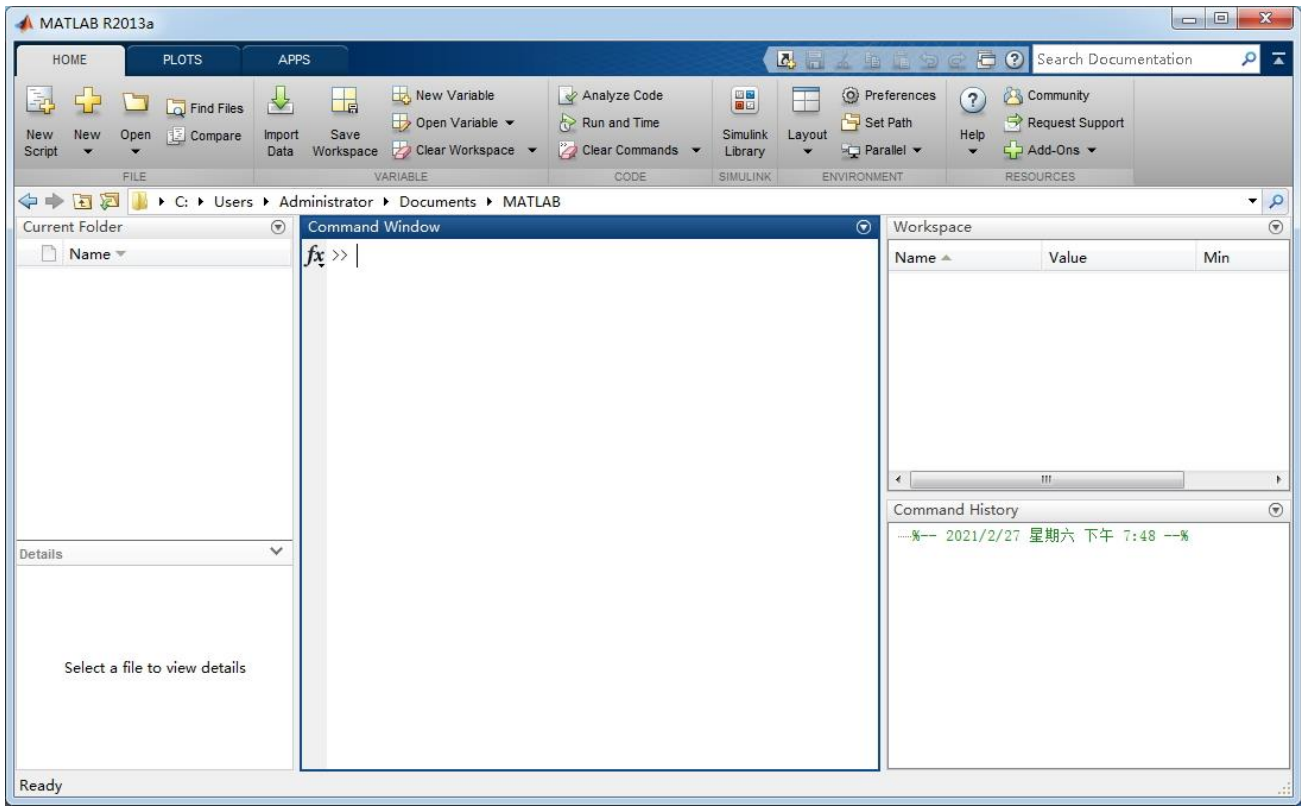


图 1.2.1 MATLAB 2013a 默认 Desktop 操作界面

如图 1.2.1 所示，MATLAB 2013a Desktop 操作界面包括多个窗口。除了包含菜单栏外，还包括 Command Window（命令）窗口、Command History（历史记录）窗口、Current Folder（当前目录）窗口、Workspace（工作区）窗口等。

1.2.4 Command Window 运行入门

MATLAB 有许多使用方法，但最基本且入门时首先要掌握的是 MATLAB Command Window（命令窗口）的使用方法。

命令窗口用于输入命令、运行 MATLAB 函数和脚本，并显示结果。默认情况下，命令窗口位于 MATLAB Desktop 操作界面的中部。正常模式下，命令窗口中会显示 >> 运算提示符。若显示 K>> 提示符，表示当前处于调试模式，键入 dbquit 则可返回正常模式。

为了方便操作，在 Command Window 中可以对输入的命令进行编辑。表 1.2.1 给出来了键盘常用快捷键的使用说明。表 1.2.2 列出了命令行中常用的操作命令。

表 1.2.1 命令窗常用快捷键

功 能 键	功 能 说 明	功 能 键	功 能 说 明
↑	调出前一个输入的命令	Backspace	清除光标所在位置前面的字符
↓	调出后一个输入的命令	F9	运行选中命令
←	光标左移一个字符	Ctrl+k	删除光标之后到行尾的所有字符
→	光标右移一个字符	Ctrl+c	中断正在执行的命令
Ctrl+←	光标左移一个单词	Ctrl+d	打开选中的变量或函数文件
Ctrl+→	光标右移一个单词	Ctrl+0	打开 Command Window
Home	光标移至行首	Ctrl+1	打开 Command History
End	光标移至行尾	Ctrl+2	打开 Current Folder
Esc	清除当前行	Ctrl+3	打开 Workspace
Del	清除光标所在位置后面的字符		

表 1.2.2 常用的操作命令

命 令	功 能 说 明	命 令	功 能 说 明
cd	设置当前工作目录	exit	关闭退出 MATLAB
clf	清除当前图形窗口内的图形	quit	关闭退出 MATLAB
clc	清除 Command Window 的显示内容	md	创建目录
clear	清除 MATLAB 工作区中保存的变量	more	使其后显示的内容分页进行
dir	列出指定目录下的文件和子目录清单	type	显示指定 M 文件的内容
whos	显示工作区中的所有变量信息	close	关闭当前图形窗口

1.2.5 Command History 窗口

MATLAB 的 Command Window 提供了非常友好的交互功能，用户可以在此环境中边思考边验证。完成设计之后，可以通过 MATLAB 的历史记录功能将已验证的命令再次提取出来。这种记录命令的能力就体现在 MATLAB 的 Command History（历史记录）窗口。

在 Command History 窗口中，记录了在 MATLAB 命令窗口中键入的所有命令，包括每次启动 MATLAB 的时间。这些命令不但可以记录在 Command History 窗口中，而且可以再次执行。双击 Command History 窗口中记录的某一条命令，就可以将其发送到命令窗口中再次执行。

为了方便以后使用，用户可能希望将命令窗口中使用的命令通过文件的方式保存起来。MATLAB 为此提供了 diary 命令，实现上述功能，具体步骤如下。

- 系统总是保存日志文件到当前工作路径下，所以需要将需要保存日志文件的目录（如 D:\comhis）设置为当前目录。设置目录可以通过命令“cd d:\comhis”命令实现（也可以在 Current Folder 窗口点击选择目标文件夹）。
- 在 MATLAB 中运行命令“diary filename”开始保存之后的所有命令，文件名为 filename（文件名可以任意取）。
- 运行关闭命令“diary off”后，内存中保存的操作内容就将全部记录在 D:\comhis 目录下的名为 filename 的目录文件中。

保存后的文件不能直接在 MATLAB 中运行，但可以通过 MATLAB 中的 M 文件编辑器或其它文本读写软件阅读和编辑。

1.2.6 Current Folder 窗口

MATLAB 加载任何文件、执行任何命令都是从当前目录下开始的，因此 MATLAB 提供了 Current Folder（当前目录）窗口，所有文件的保存和读取都是在这个默认路径下进行的。

可以通过点击 Current Folder 上方的路径选择按钮  完成当前目录的更改，也可以通过 diary 命令完成同样的功能。

1.2.7 Workspace 和 Variable Editor 窗口

➤ Workspace 窗口

MATLAB 要处理各种各样的数据，还需要有一个专门的内存空间来存放它们，这个地方就是 Workspace 窗口。数据存放在 Workspace 窗口中，可以随时被调用。在 Workspace 窗口中将显示目前内存中所有的变量名称、数据结构、字节数、类型、最大值、最小值、平均值及方差等统计信息，不同的变量类型分别对应不同的变量名图标。

➤ Variable Editor 窗口

双击 Workspace 窗口中的变量名，会弹出 Variable Editor 窗口。通过该窗口可以查看变量的内容，还可以对变量进行各种编辑操作。双击需要修改的数据单元，即可对相应的数据进行修改。在 Variable Editor 窗口中用户可以选择所需要的元素，然后通过 PLOTS 选项卡中的绘图工具进行快速绘图。在 VARIABLE 选项卡中，用户可以对变量进行插入、删除、转置、排序等操作。

1.2.8 帮助系统

➤ 帮助浏览器

MATLAB 拥有非常强大的帮助浏览器即 Help Browser，进入帮助浏览器的方法有以下几种。

1) 直接单击 MATLAB 桌面工具栏中的  按钮。

2) 使用快捷键 F1。

通过以上方法打开的是帮助浏览器窗口的导航页。它主要包括搜索栏以及各工具文档连接两部分。单击某一工具箱链接即可进入具体工具箱的详细帮助页面。在这里用户可以通过选择目录查找相关的内容，或者通过搜索关键词来查找所需内容。

➤ 在 Command Window 中查询帮助

熟练的用户可以使用更为快速的 Command Window 查询帮助。这些帮助主要有 help 命令和 lookfor 命令。

● help 命令

在 MATLAB 中，可以直接使用 help 获得指令的使用说明。如果准确的知道所要求助的主体词或指令名称，那么使用 help 是获得在线帮助最简单有效的方式。具体的，在命令窗口中输入“help name”即可，其中 name 根据需要换成要查找的主体词或指令名称。

● lookfor 命令

命令 lookfor 与 help 不同，lookfor 只列出 M 文件的 H1 行内容，且对关键字的搜索不是完全匹配式的。用 lookfor 搜索，一旦发现此行中含有所查询的字符串，则将所有满足条件的函数名及对应的 H1 行注释全部显示出来。

1.3 MATLAB 基本语法规则

1.3.1 数值和变量

➤ 数值

MATLAB 的数值采用习惯的十进制表示方法，可以带小数点或者负号以及科学计数法。MATLAB 中的数值精度是 $\text{eps} (=2.2204\text{e-}16)$ ，即大约保留有效数字 16 位。数值范围为 $10\text{e-}308 \sim 10\text{e}308$ ，即 $1 \times 10^{-309} \sim 1 \times 10^{309}$ 。

➤ 变量

变量是保存数据信息的一种最基本的数据类型。在 MATLAB 中，变量不用预先声明就可以进行赋值。变量的命名应遵循如下规则：

- 1) 变量名必须以字母开头；
- 2) 变量名可由字母、数字和下划线混合组成；
- 3) 变量名区分字母大小写；
- 4) 变量名尽量不要使用系统已有的函数名（如 sin、conv 等）；
- 5) MATLAB 保留了一些具有特定意义的默认变量，可以直接使用，且在变量命名时不要使用这些保留字符，表 1.3.1 为 MATLAB 的系统保留变量。

表 1.3.1 MATLAB 的系统保留变量

变量名	含 义
-----	-----

i 或 j	虚数单位（代表 $\sqrt{-1}$ ）
pi	圆周率（ π ）
ans	存放最近一次无赋值变量语句的计算结果
Inf 或 inf	无穷大（ ∞ ）
eps	数值精度 2.2204e-16，为机器的浮点运算误差限（若某变量的绝对值小于 eps，则视为 0）
NaN 或 nan	0/0，或 inf/inf，或超出储存大小的值
lasterr	存放最后一的错误信息
lastwarn	存放最后一的警告信息
nargin	函数输入的变量数目
nargout	函数输出的变量数目
realmax	最大正实数 1.7977e+308
realmin	最小正实数 2.2251e-308

➤ 复数

MATLAB 将复数作为一个整体处理，但是把实部和虚部分开处理。虚数单位用预定义变量 i 或者 j 表示，且 MATLAB 中复数的书写形式与其自然书写形式一致，如 5+6i 或 0.2-1.7j。常用的复数处理函数如表 1.3.2 所示。

表 1.3.2 MATLAB 复数处理函数

函 数	功 能	函 数	功 能
real(x)	取复数 x 的实部	abs(x)	取复数 x 的模
imag(x)	取复数 x 的虚部	angle(x)	取复数 x 的相位角

1.3.2 赋值语句

MATLAB 采用命令行形式的表达式语言，每一个命令行就是一条语句，其格式与书写的数学表达式十分相近，非常容易掌握。用户在命令窗口输入语句并按回车键确认后，该语句就由 MATLAB 系统解释运行，并给出运行结果。MATLAB 的赋值语句包括两种结构：直接赋值语句、函数调用语句。

➤ 直接赋值语句

基本赋值语句的基本结构如下：

赋值变量 = 赋值表达式

其中，等号右边的表达式可以由变量名、常数、函数和运算符构成。此时，直接赋值语句把右边表达式的值直接赋给左边的赋值变量。

需要注意的是：

- 1) 若赋值语句后面没有分号“;”，MATLAB 命令窗口将显示表达式的运算结果；若不想显示运算结果，则应该在赋值语句末尾加上分号；
- 2) 若等式右边的赋值表达式不是数值，而是字符串，则字符串两边应加单引号；
- 3) 若省略赋值语句左边的赋值变量和等号，则表达式运算结果将默认赋值给系统保留变量 ans。

➤ 函数调用语句

函数调用语句的基本结构如下：

[返回变量列表] = 函数名(输入变量列表)

其中，函数可以分为两大类：一类是 MATLAB 内核中已经存在的内置函数；另一类是读者根据需要自定义的函数。返回变量列表和输入变量列表均可以由若干变量名组成，若返回变量个数大于 1，则它们之间应该用逗号或空格分隔；若输入变量个数大于 1，则它们之间只能用逗号分隔。

关于函数定义、调用方法的详细介绍请参考后文相关章节。

1.3.3 矩阵及其元素表示

MATLAB 的中文意思是“矩阵实验室”，矩阵是 MATLAB 进行数据处理的基本变量单元。因此，掌握矩阵的表示方法是进行 MATLAB 编程和应用的基础。

➤ 矩阵的表示

矩阵可以看做是二维数组，这里先讨论一维数组的表示。

- 一维数组（行向量）的创建
 - 1) 直接列举数组元素：将数组中的所有元素一一列举，中间用空格或逗号隔开，并用方括号“[]”包括矩阵的所有元素，例如 [1 2 3] 和 [1,2,3]。
 - 2) 步长生长法：按照 a:inc:b 的格式创建一维数组，其中 a 表示数组的第一个元素，b 表示数组的最后一个元素，inc 表示增长的步长，例如 1:0.2:2 与 [1,1.2,1.4,1.6,1.8,2] 等价。当 inc 为 1 时，inc 可省略，即表示为 a:b。

- 二维数组（矩阵）的创建

按照如下格式创建二维数组

[一维数组(1); 一维数组(2);... 一维数组(m)]

表示创建 m 行的矩阵，每行之间用分号“;” 隔开（也可以这样理解，逗号“,” 分隔开的是两个列向量）。

为了便于用户使用，提高编程效率，除了最基本的直接输入法外，MATLAB 还提供给用户一些可以直接调用的内置基本矩阵函数，可以快速的创建特殊的矩阵。具体如表 1.3.3。

表 1.3.3 MATLAB 主要内置基本矩阵函数

函 数	功 能
ones(m,n)	产生 m 行 n 列的全 1 矩阵
zeros(m,n)	产生 m 行 n 列的全 0 矩阵
rand(m,n)	产生 m 行 n 列的在 [0,1] 区间均匀分布的随机数矩阵
randn(m,n)	产生 m 行 n 列的的正态分布的随机数矩阵
eye(n)	产生 n 行 n 列的单位矩阵

- 矩阵元素的引用

矩阵元素的行号和列号称为该元素的下标，是通过圆括号“()” 中的数字来标识的。矩阵元素可以通过其下标来引用，如 A(i,j) 表示矩阵 A 第 i 行第 j 列的元素。

这里还要介绍一下矩阵元素引用中冒号“:” 的用法，这里举例说明。A(2,:) 表示矩阵 A 第 2 行全部元素组成的行向量；A(:,2) 表示矩阵 A 第 2 列全部元素组成的列向量；A(1,1:2) 表示矩阵 A 第 1 行第 1~2 列的全部元素组成的行向量。这种操作在有些语言里叫做“矩阵切片” 操作。

1.3.4 数据的输出及文件的读写

- disp 与 fprintf 函数的使用
 - disp 函数的使用：

disp 函数可以直接将数组变量所对应的值输出到屏幕，调用格式为：

disp(X)

其中 x 为数组变量；x 也可以是固定的字符串，要输出的字符串要用单引号括起来。
 - fprintf 函数的使用：

MATLAB 中的 fprintf 函数可以将数据按指定格式写入到文本文件中或输出到屏幕。

其调用格式为：

fprintf(fid,format,variables)

其中 fid 为文件句柄，可用 fopen 函数获取，若缺省，则输出到屏幕。

format 为格式控制输入项，格式控制输入项是用单引号括起来的字符串，也称为转换控制字符串。其中包括格式字符和普通字符。

格式字符用来进行格式说明，作用是将输出的数据转换为指定的格式。格式字符通常以 % 字符开头。表 1.3.4 列出了常用格式字符。

普通字符是需要原样输出的字符，包括单引号内的逗号、空格和字符串。format 中还会用到转义字符，转义字符是一种特殊的字符，用于进行格式控制或输出特定的字符。在 MATLAB 中大部分转义字符以反斜杠“\” 开头，使用时也放在单引号内，常用的转义字符及其含义如表 1.3.5 所示。

表 1.3.4 fprintf 函数的格式字符

格 式 字 符	功 能 说 明
%d 或 %i	以带符号的十进制形式输出整数

%o	以八进制无符号形式输出整数
%x 或 %X	以十六进制无符号形式输出整数。用 x 时，a~f 字母以小写形式输出；用 X 时，A~F 字母以大写形式输出
%u	以无符号十进制形式输出整数
%c	以字符形式输出，只输出一个字符
%s	输出字符串
%f	以小数形式输出
%e 或 %E	以指数形式输出实数，用 e 时指数以“e”表示，用 E 时指数以“E”表示。

表 1.3.5 常用的转义字符

转 义 字 符	含 义	转 义 字 符	含 义
\n	回车换行	\\	输出一个反斜杠 “\”
\t	横向跳到下一制表位置	%%	输出一个百分号 “%”
\v	竖向跳格	' '	输出一个单引号 “'”
\b	退格		
\r	回车		

variables 为输出列表，列出要进行输出的数据，可以是变量或表达式。

例如，在命令窗口中输入：

```
>> fprintf('the int is: %d\n',65);fprintf('the char is: %c\n',65);
```

得到输出：

```
the int is: 65
the char is: A
```

➤ MAT 文件的读写

MAT 文件是 MATLAB 格式的双精度二进制数据文件，由 MATLAB 软件创建。可以使用 MATLAB 软件在其它计算机上读取，同时也可以使用其它软件通过 MATLAB 的应用程序口来进行读写操作。如果只是在 MATLAB 环境中处理数据，那么使用 MAT 文件格式是最方便的，因为这样会省去文件格式转换的操作。

● MAT 文件的写入

通过调用 save 函数，可以将 Workspace 中的变量导出，储存为 MAT 文件。save 函数的调用语法如下：

```
save filename
```

表示将 Workspace 中的全部变量导出保存在 filename.mat 文件中，默认保存在 MATLAB 当前目录下。若忽略文件名，系统则会使用默认的 matlab.mat 文件名来保存文件。

另外，也可以只保存 Workspace 中的指定变量，语法如下：

```
save filename var1 var2 ... varN
```

将变量 var1、var2 ... varN 保存入名为 filename.mat 的文件中（文件名任意）。

● MAT 文件的读取

通过调用 load 函数，可以从硬盘把 MAT 文件导入到 Workspace 中。若将文件的全部变量导入 Workspace 中，调用语法如下：

```
load filename
```

另外，还可以只导入文件中指定的变量，语法如下：

```
load filename var1 var2 ... varN
```

在把数据导入 Workspace 时，如果导入的变量名与 Workspace 中原有的变量相同，MATLAB 将会以新导入的变量覆盖原有变量。

➤ audioread 与 sound 函数的使用

● audioread 函数的使用：

用于读取音频文件，具体调用格式为：

```
[Y,Fs] = audioread(filename)
```

其中 Filename 为音频文件带路径的名字（带文件后缀），需要用单引号括起来（若文件在工作路径中则只需列文件名）。支持 wav、mp3 等常用格式。

F_s 为音频信号的采样频率，可以利用Windows中的文件属性页查询。

Y 为输出的音频数字信号，默认是 n 行 2 列，其中 2 代表这个音频信号是两个声道的， n 的值等于该音频信号的时长乘采样频率 F_s 。

即通过运行 `audioread` 函数，获得数字音频文件的双声道数字信息和自带采样率。

- **sound 函数的使用：**

用于播放音频文件，需要给出两个参数，具体调用格式为：

`sound(Y, Fs)`

其中 Y 与 `audioread` 函数的参数 Y 含义一致，不过这里 Y 是输入参数，是要播放的音频文件。这里还需要 F_s 采样频率参数，改变 F_s 会得到加速或慢速播放的效果。

1.3.5 MATLAB 编程原则

- 百分号“%”后面的内容是程序的注解，要善于运用注解使程序更具可读性。
- 养成在主程序开头用 `clear` 指令清除变量的习惯，以消除工作空间中其它变量对程序运行的影响，但注意子程序中不要用 `clear`。
- 养成在主程序开头用 `clc` 和 `close all` 指令的习惯，作用分别是清空命令窗口和关闭所有的 `figure` 窗口，而在子程序中一般不使用。
- 参数的赋值要集中放在程序的开始部分，以便维护。
- 对于不需要显示的中间结果，所在语句行后要输入分号，使其不在命令窗口中显示，以提高执行速度。
- 注意 **MATLAB** 工作路径的设置，以便程序运行。
- 对于复杂的任务，程序设计要尽量做到模块化，即采用主程序调用子程序的方法执行任务。

1.4 M 文件

1.4.1 M 文件简介




在 **MATLAB** 系统中编写程序需要使用 **M** 文件，包括脚本式 **M** 文件和函数式 **M** 文件。**M** 文件是系统中统一应用的扩展名为 `.m` 的文件：**MATLAB** 系统大量定义了该文件作为自带函数；用户也可以定义自己的脚本式 **M** 文件完成特定的任务，或定义自己的函数式 **M** 文件供以后调用。

MATLAB 命令有三种执行方式：

- 在命令窗口中逐行输入命令，通过与 **MATLAB** 交互的方式执行命令。
- 编写 **M** 文件，整批执行 **M** 文件中写好的命令。
- 利用预编程的 **GUI** 图形用户界面执行命令。

交互式命令行工作方式适用于命令行比较简单，同时处理问题较少的情况。但是当需要处理复杂问题时，直接在命令行输入程序的方式就会比较吃力，这时用户就需要使用 **M** 文件进行编程，批量执行命令。至于 **GUI** 图形用户界面，其后台的程序指令也是编写在特定的 **M** 文件中的（关于 **GUI** 图形用户界面方面的内容将在后面相关章节中详细介绍）。所以，熟练掌握 **M** 文件的用法，对于掌握 **MATLAB** 程序设计至关重要。

1.4.2 M 文件的基本操作

- **M** 文件的创建：单击工具栏上的 **New Script** 按钮 ，或单击工具栏上的 **New** 按钮 ，再选择 **Script** 项 ，即可弹出如图 1.4.1 所示的 **M** 文件编辑器界面，默认文件名为 `Untitled.m`。能看出，按如上方法建立的 **M** 文件是空白的，可以利用 **M** 文件编辑器写入代码。

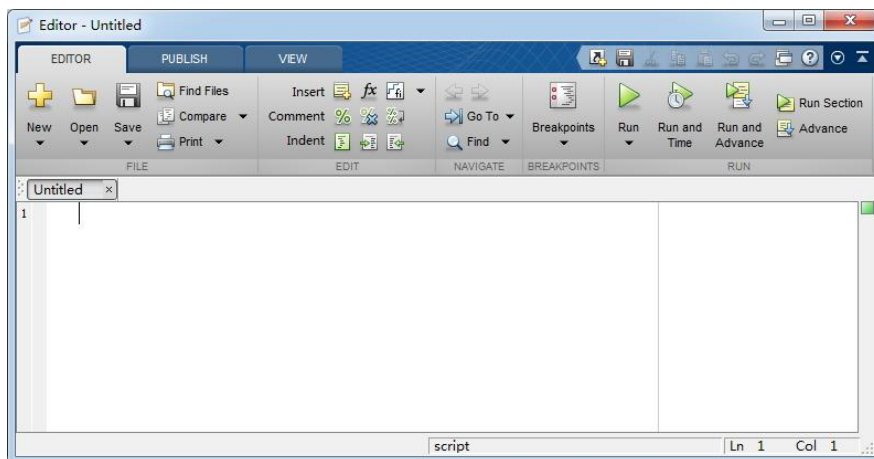




图 1.4.1 M 文件编辑器界面

此外，还可以创建函数式 M 文件专用的程序模板，点击工具栏上的 New 按钮 ，再选择 Function 项 ，即可弹出如图 1.4.2 所示的带有函数式 M 文件模板的 M 文件编辑器界面。默认文件名同样为 Untitled.m。但此时，M 文件创建后，预先已有一些代码。在后面的讲解中会介绍，这是建立函数式 M 文件所必须的函数体，只需在函数体模板内加入代码即可。

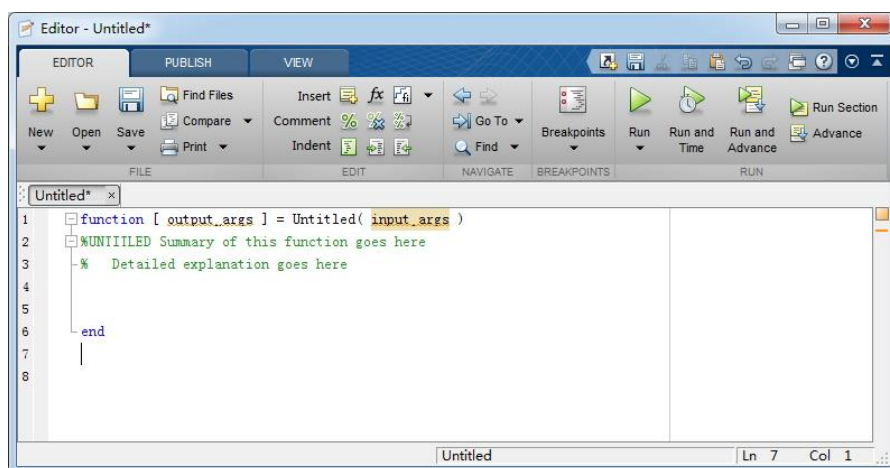






图 1.4.2 带有函数式 M 文件模板的 M 文件编辑器界面

➤ M 文件的保存：在 M 文件编辑器界面工具栏中点击 Save 按钮 ，再选择 Save As 项 ，在弹出的保存对话框中选择保存路径，设置 M 文件的文件名，然后点击确定进行保存。这里需要注意的是 M 文件的命名原则，需要注意以下几点：

- 1) 必须以字母开头；
- 2) 可由字母、数字和下划线混合组成（不能出现空格）；
- 3) 区分字母大小写；
- 4) 不可以用中文（保存路径中的文件夹名称可以用中文）；
- 5) 不要使用系统自带关键词（如保留变量），尽量不要使用系统已有的函数名。

保存 M 文件时使用的文件名如果与系统已有的函数名重复，主函数调用该名称的 M 文件时，是调用自己编写的 M 文件还是调用系统已有的 M 文件，需按照一定的规则，将在以后关于函数调用的相关章节中介绍。

➤ M 文件的打开：打开已有的 M 文件，需点击工具栏的 Open 按钮 ，在弹出的窗口中选择要打开的 M 文件，点击打开即可。

➤ M 文件的运行：单击打开 M 文件上方工具栏的 Run 按钮 ，若弹出如图 1.4.3 所示的对话框，表示当前 M 文件所在的目录不是 MATLAB 当前的工作路径（Current Folder），也不在 MATLAB 搜索路径列表中。要运行此 M 文件，需要将 MATLAB 当前的工作路径变更为此 M 文件所在的路径（点击 Change

Folder 按钮), 或者把此 M 文件所在的路径添加到 MATLAB 搜索路径列表中 (点击 Add to Path 按钮)。

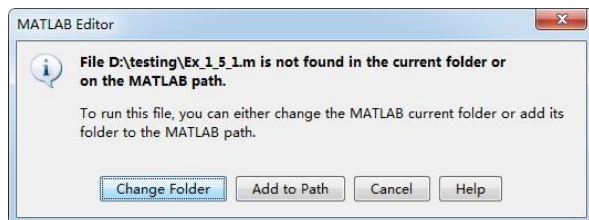


图 1.4.3 更改当前工作路径或添加路径到 MATLAB 搜索路径列表中

Current Folder 即是 MATLAB 当前的工作路径, 所有文件的保存和读取都是在这个路径下进行的。而 MATLAB 搜索路径列表是 MATLAB 程序运行中调用函数所搜索的路径, 关系到 MATLAB 在运行一句命令时如何选择函数, 包括搜索被运行函数的顺序。单击 MATLAB 工具栏中的 Set Path 菜单项 (如图 1.4.4), 可以弹出 Set Path 窗口 (如图 1.4.5), 从中可以设置该搜索路径。



图 1.4.4 Set Path 菜单项

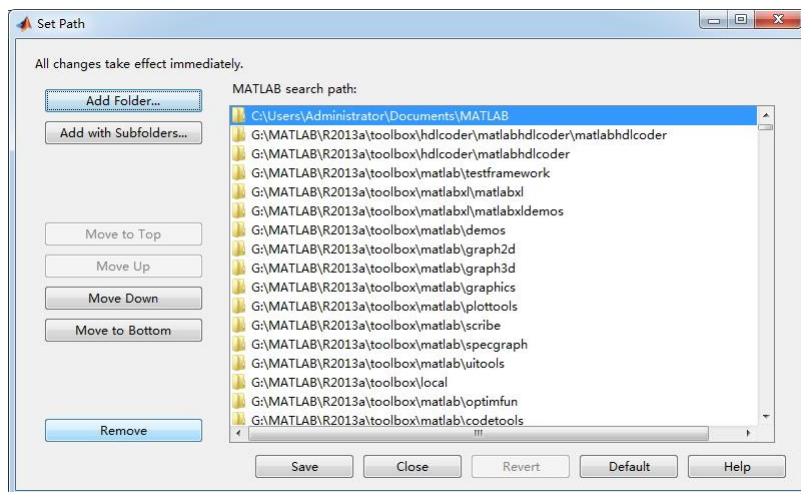


图 1.4.5 MATLAB 搜索路径列表设置窗口

若点击图 1.4.3 中的 Add to Path 按钮, 当前 M 文件所在的目录就被添加到图 1.4.5 所示搜索路径列表的最前端 (在此列表中的搜索优先级最高); 若点击图 1.4.3 中的 Change Folder 按钮, 当前 M 文件所在的目录不会被添加到图 1.4.5 路径列表中, 而是将 MATLAB 的当前工作路径 (Current Folder) 设为当前 M 文件所在的目录。此时, 当系统搜索函数时, 若函数重名, 优先搜索当前工作路径中的函数, 再搜索 MATLAB 搜索路径列表中的函数。虽然两种方法对于主程序和被调用函数在同一文件夹的情况是等效的, 但第一种情况, 即加入 MATLAB 搜索路径列表后, 在其他文件夹中的主程序也可以优先调用此路径下的函数文件。

另外, 运行 M 文件, 也可以在命令窗口中直接输入文件名 (不包括 .m), 会直接显示运行结果。

1.4.3 脚本式 M 文件

脚本式 M 文件是由 MATLAB 语句构成的文本文件, 运行命令文件的效果等价于从 MATLAB 命令窗口中按顺序逐条输入并运行文件中的指令, 类似于 DOS 下的批处理文件。脚本式 M 文件不能接受输入参数, 也不返回输出结果; 但脚本式 M 文件可以将变量保存在工作空间 (Workspace) 中, 脚本式 M 文件也可以访问 MATLAB 当前工作空间的变量, 其它脚本文件和函数可以共享这些变量。脚本文件常用于主程序的设计。下面将通过运行一个脚本式 M 文件完成对三个数字的排序。

【例 1.4-1】 编写脚本文件, 对数 a、b、c 进行排序, 并按从大到小的顺序输出。

新建一个 M 文件, 在编辑器中输入如下代码:

```
a = 10;  
b = 3;  
c = 7;  
disp(sort([a,b,c], 'descend'));
```


以文件名 Ex_1_5_1.m 保存该文件（脚本式 M 文件的文件命名比较自由，可以用其它任何满足要求的名称）。

例 1.4-1 中程序用到了排序函数（sort）和显示函数（disp），都是 MATLAB 的内置标准函数。前者用于对矩阵进行排序，'descend' 参数表示降序，后者将结果显示在命令窗口中。运行后在命令窗口中显示出排序结果如下：

```
>> Ex_1_5_1
    10     7     3
```

1.4.4 函数式 M 文件

函数式 M 文件是 M 文件的另一种类型，它也是由 MATLAB 语句构成的文本文件。函数式 M 文件的第一行以 function 关键词开始，说明此文件是一个函数。其实质为用户向 MATLAB 函数库中添加的自定义函数。默认情况下，函数式 M 文件中的变量都是局部变量，仅在函数运行期间有效，函数运行结束后，这些变量将从工作区中清除，在函数中使用全局变量的情况区除外，若使用全局变量，则在调用函数和被掉用函数中都需要用 global 关键字声明。下面将通过一个计算 n 的阶乘的函数说明函数式 M 文件的基本结构。

【例 1.4-2】 函数式 M 文件的结构

fact.m

```
function f = fact(n)                % 函数定义行
% Compute a factorial value.        % H1行
% FACT(N) returns the factorial of N, % Help 文本
% usually denoted by N!

% FACT(N) is PROD(1:N).            % 注释
f = prod(1:n);                      % 函数体
end
```

1) 函数定义行

函数定义行用来定义函数名称、输入输出变量的数量和顺序，必须写在函数式 M 文件的第一行，对于脚本式 M 文件没有此行。完整的函数定义语句如下：

```
function [out1, out2, out3...] = funName(in1, in2, in3...)
```

其中 function 是不可以改变的，代表该文件为函数式 M 文件。输入信号用圆括号括起来，变量间用英文逗号分隔，只有一个输入变量圆括号也不可以省略。输出变量用方括号括起来，同样用逗号分隔，只有一个输出变量时，方括号可以省略。另外当函数没有输出时，可按下面两种方式书写：

```
function [] = funName(in1, in2, in3...)
```

或

```
function funName(in1, in2, in3...)
```

funName 是用户自定义的函数名，与 MATLAB 变量的命名规则一样。另外，当保存函数式 M 文件时，MATLAB 会默认以函数的名字来保存，请不要更改此名称，否则调用所定义的函数时会发生错误。由于函数名与文件名的关联性，用户自定义的函数名还应遵循 M 文件的命名规则。

2) H1 行

H1 行紧跟着函数定义行。因为它是 Help 文本的第一行，所以叫它 H1 行，用百分号（%）开始，属于注释语句。强烈建议用户在编写函数式 M 文件时，将函数的功能、调用函数的参数等概括描述出来，以供自己和别人查看，方便函数的使用。在命令窗口中输入以下命令可以显示 H1 行文本。

```
lookfor filename
```

例如刚才的例 1.4-2 建立的函数式 M 文件为 fact.m，在命令窗口中利用 lookfor 命令可见以下输出。

```
>> lookfor fact
coninputfactor      - Input factor object for Constraints
cset_fullfact       - Full Factorial Design generator object
cgexprfactory       - Construct a new cgexprfactory object
mbcinputfactor      - input factor class
fact                - Compute a factorial value.          % H1 行
...                % 以下结果省略
```


`lookfor` 命令搜索所有函数 H1 行中包含 `fact` 字符串的函数，将这些函数列出来，并且将它们的 H1 行显示出来（函数的 H1 行中必须有函数名称的字符串）。从 `lookfor` 命令的结果中可以看到 4 个其它的包含 `fact` 字符串的函数，以及要找的 `fact` 函数，并且分别显示了它们的 H1 行。

3) Help 文本

Help 文本是为调用帮助命令而建立的文本，可以是连续多行的注释文本。在命令窗口中输入以下命令可以显示 Help 文本。

```
help filename
```

例如刚才的例 1.4-2 建立的函数式 M 文件为 `fact.m`，在命令窗口中利用 `help` 命令可见以下输出。

```
>> help fact
Compute a factorial value.                % H1 行
fact(N) returns the factorial of N,      % Help 文本
usually denoted by N!
```

以上命令的结果显示了 `fact` 函数文件的注释行，从 H1 行开始，直到第一个非注释行（包括空行）结束。

4) 注释

以 % 开始的注释行可以出现在函数的任何地方，当然，也可以出现在一行语句的右边。换句话说，上面的 H1 行也属于 Help 文本，而 Help 文本都属于注释语句。对程序进行注释可以方便以后的阅读和维护，程序运行时不会执行注释语句。

5) 函数体

函数体是函数式 M 文件与脚本式 M 文件中计算和处理数据的主体，可以包含进行计算和赋值的语句、函数调用、循环和流控制语句，以及注释语句、空行等。

如果函数体中的命令没有以分号 “;” 结尾，那么该行返回的变量将会在命令窗口显示其具体内容。如果在函数体中使用了 “`disp`”、“`fprintf`” 等输出函数，那么结果也将显示在命令窗口中。用户可以通过这些功能来查看中间计算过程或者最终的计算结果。

1.5 函数调用

1.5.1 MATLAB 函数的类型

MATLAB 的函数有多种类型，包括：主函数、子函数及私有函数等。

➤ 主函数

主函数在结构上与其他函数没有区别，之所以叫它主函数，是因为它在函数式 M 文件中排在最前面，其它子程序都排在它后面。主函数与其所在 M 文件同名，是唯一可以在命令窗口或者其它函数中调用的函数。主函数通过所在 M 文件名（也就是主函数名）来调用，本书前面涉及的函数文件都是主函数。

➤ 子函数

一个 M 文件中可以写入多个函数，排在第一个位置的是主函数，在主函数后面定义的函数都叫子函数，子函数的排列无规定顺序。子函数只能被同一个 M 文件上的主函数或同一个 M 文件上的其它子函数调用。每个子函数都有自己的函数定义行。

子函数所在的 M 文件必须以主函数开头，也就是说脚本式 M 文件中不可以包含子函数，运行会报错。

需要注意的是，虽然几个子函数在同一个文件夹中，但各有自己的变量储存空间，子函数之间不能相互存取其它子函数的变量。但若声明变量为全局变量，就另当别论了。

➤ 私有函数

私有函数在编写形式上和主函数相同，但它是私有的，只有父函数式 M 文件才能调用它。储存私有函数需要在当前目录下创建一个子文件夹，子文件夹名字必须为 `private`。存放于 `private` 文件夹内的函数即为私有函数，它的上层目录称为父目录，只有父目录中的函数式 M 文件才可以调用该私有函数。

将调用私有函数的条件概括如下：

- 1) 私有函数对于其父目录以外的目录中的 M 文件来说是不可见的。
- 2) 调用私有函数的 M 文件必须在 `private` 子目录的直接父目录内。
- 3) 调用私有函数的 M 文件必须为函数式 M 文件，脚本式 M 文件无法调用私有函数。

私有函数与子函数的区别在于：私有函数可以方便地供一批父 M 文件来调用；而子函数只能供其所在的主函数调用。

1.5.2 MATLAB 函数调用查找顺序

下面将通过一个例子来说明 MATLAB 在调用同名函数时的查找顺序。

【例1.5-1】 MATLAB对于子函数、私有函数、同一文件夹中的函数、系统内置标准函数的调用顺序。

这里应用系统内置标准函数disp，建立一个函数式M文件testing.m，在同一个文件夹下建立一个函数式M文件disp.m，与系统内置标准函数disp重名。在testing.m所在文件夹下建立private子文件夹，在子文件夹下建立私有函数disp.m，同样与系统内置标准函数disp重名。再在testing.m文件中编写子函数disp，也与系统内置标准函数disp重名。文件结构如图1.5.1所示。

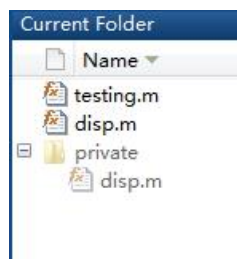


图 1.5.1 函数调用顺序测试文件结构

其中，testing.m 的代码如下：

```
function testing(x)      % 主函数
disp(x);
end

function disp(x)         % 子函数
x = '子函数';
fprintf('%s \n',x);
end
```

在 testing.m 文件中包含同名的主函数和名为 disp 的子函数，其中应用了格式化输出函数 fprintf，使用方法与 C 语言的 printf 类似。另外，在testing.m 文件主函数中调用了名为 disp 的函数。

与 testing.m 在同一文件夹中的 disp.m 文件代码如下：

```
function disp(x)
x = '同一文件夹中的函数';
fprintf('%s \n',x);
end
```

在 private 文件夹中的 disp.m 文件代码如下：

```
function disp(x)
x = '私有函数';
fprintf('%s \n',x);
end
```

可以看出，上面两个 disp.m 文件和 disp 子函数实际上并没有应用传入的参数值，而是重新定义了各自的输出的内容。此时，在MATLAB命令窗口中输入：

```
>> testing('系统内置标准函数');
```

如果输出“系统内置标准函数”表明调用了系统内置标准函数；如果输出“子函数”表明调用了子函数；如果输出“私有函数”表明调用了私有函数；如果输出“同一文件夹中的函数”表明调用了同一文件夹中的函数文件。

运行后，输出结果为“子函数”，说明子函数的搜索优先级最高。接下来，将 testing.m 文件中的子函数部分删除，再次运行：

```
>> testing('系统内置标准函数');
```

输出结果为“私有函数”。删除私有函数再次运行，输出结果为“同一文件夹中的函数”。删除同一文件夹中的函数文件，输出结果为“系统内置标准函数”。

通过上面的例子，证明了系统调用函数的优先级由高到低依次是：子函数、私有函数、同一文件夹中的

函数、系统内置标准函数。

需要说明的是,这里为了研究系统调用函数的优先级,我们定义了与系统内置标准函数同名的各种函数。但以上程序在运行中出现了警告,在实际应用中,我们应该尽量避免定义这种与系统内置标准函数同名的函数,方便程序的维护。另外,前面的函数式M文件中都没有H1行、Help文本和其它注释,是为了简洁的说明问题,而对于函数式M文件的编写,还是应该尽量规范。

1.6 常用命令

1.6.1 常用运算符

MATLAB 语言中的运算符和其它高级语言类似,不同之处在于,MATLAB 中有许多针对矩阵的运算符,在使用中还需要注意参加运算的项是否为矩阵,因为同一个运算符对于矩阵和对于一般数字,表达含义不同。和其它语言一样,在 MATLAB 中表达式可以将运算符自由组合,组成更为复杂的运算表达式。在对运算符进行组合的过程中,需要注意优先级问题。下面(1)至(9)列出了 MATLAB 语言的常用运算符,并按计算优先级从高到低的顺序进行排列。

- (1) 括号 ()。
- (2) 元素群转置 (.'), 元素群幂 (.^), 共轭转置 ('), 矩阵幂 (^)。
- (3) 代数正 (+), 代数负 (-), 逻辑非 (~)。
- (4) 元素群乘法 (.*), 元素群除法 (./), 矩阵乘法 (*), 矩阵右除 (/), 矩阵左除 (\)。
- (5) 加法 (+), 减法 (-)。
- (6) 冒号运算符 (:)。
- (7) 小于 (<), 小于等于 (<=), 大于 (>), 大于等于 (>=), 等于 (=), 不等于 (~=)。
- (8) 元素“与”(&)。
- (9) 元素“或”(|)。
- (10) 短路逻辑“与”(&&)。
- (11) 短路逻辑“或”(||)。

➤ 上面有关“元素群...”的提法,是指矩阵中的所有元素按单个元素进行运算,详见下一节 1.6.2 数组运算中的表 1.6.1 及相关说明。

➤ 再解释一下“&”与“&&”的用法与区别。

对表达式 A&B 的解释:

- 1) 首先判断 A 的逻辑值,然后判断 B 的值,然后进行逻辑与的计算。
- 2) A 和 B 可以为矩阵。

对表达式 A&&B 的解释:

- 1) 首先判断 A 的逻辑值,如果 A 的值为假,就可以判断整个表达式的值为假,就不需要再判断 B 的值,简称“短路”。
- 2) A 和 B 不能是矩阵,只能是标量。

同理可得“|”与“||”的用法和区别。

如果用户希望在判断的时候,对 A 和 B 表达式都进行计算,就应该使用标准的“&”或者“|”。Matlab 中的 if 和 while 语句中的逻辑与和逻辑或都是默认使用短路逻辑运算符,也就是说在 Matlab 中的 if 和 while 语句中即使使用“&”或“|”,结果也与使用“&&”和“||”一致,在下面的程序运行结果对比中可以证明此结论。

【例 1.6-1】 编程证明 if 判断表达式默认使用短路逻辑运算符。

```
function test1
global g1 g2
a=1;
b=2;
g1=0;
g2=0;
c = add(a,b)<0 & subtract(a,b)>0;    % & 可以换做 && 再看结果
```

```

if c
    disp('true')
else
    disp('false')
end;
fprintf('%d\n',g1)
fprintf('%d\n',g2)
end

function y = add(x1,x2)
global g1
y = x1 + x2;
g1 = y;
end

function y = subtract(x1,x2)
global g2
y = x1 - x2;
g2 = y;
end

```

按上面代码执行后显示:

```

>> test1
false
3
-1

```

另一种情况, 若在 if 判断表达式中直接判断, 按下面的代码执行:

```

function test2
global g1 g2
a=1;
b=2;
g1=0;
g2=0;
if add(a,b)<0 & subtract(a,b)>0      % & 可以换做 && 再看结果
    disp('true')
else
    disp('false')
end;
fprintf('%d\n',g1)
fprintf('%d\n',g2)
end

function y = add(x1,x2)
global g1
y = x1 + x2;
g1 = y;
end

function y = subtract(x1,x2)
global g2
y = x1 - x2;

```

```
g2 = y;  
end
```

结果为:

```
>> test2  
false  
3  
0
```

从上面的程序对比中可以看出“&”与“&&”运算符的区别以及 if 语句中判断语句的运行细节。

1.6.2 数组运算

在 MATLAB 中，矩阵和数组在一般情况下是没有区别的，矩阵就是二维数组。

➤ 元素群运算与矩阵运算

MATLAB 运用于矩阵（或数组）上的数学运算有两类：一类是以线性代数中的矩阵运算法则来进行计算的矩阵运算；另一类是两个矩阵对应元素之间的运算，即矩阵的元素群运算，两者是有区别的。

表 1.6.1 列出了元素群运算与矩阵运算在指令形式和功能方面的区别。

表 1.6.1 元素群运算与矩阵运算的区别

元素群运算		矩阵运算	
指令形式	功 能	指令形式	功 能
A.'	非共轭转置，相当于 (conj (A'))	A'	共轭转置
A+B 与 A-B	A 与 B 对应元素之间加减	A+B 与 A-B	A 与 B 对应元素之间加减
k.*A 或 A.*k	k 乘以 A 的每个元素	k*A 或 A*k	k 乘以 A 的每个元素
k+A 与 k-A	k 加 (减) A 的每个元素	k+A 与 k-A	k 加 (减) A 的每个元素
A.*B	两数组对应元素相乘，再组成矩阵	A*B	按线性代数的矩阵乘法规则计算矩阵 A 和 B 的乘积
A.^k	对 A 的每个元素进行 k 次方运算，再组成矩阵	A^k	k 个矩阵 A 相乘
A./k 和 k.\A	A 的元素分别除以 k，再组成矩阵	A/k 和 k\A	A 的元素分别除以 k，再组成矩阵
k./A 和 A.\k	k 除以 A 的每一个元素，再组成矩阵	—	—
左除 A.\B	B 的元素除以 A 对应位置的元素，再组成矩阵	左除 A\B	AX=B 的解
右除 B./A	B 的元素除以 A 对应位置的元素，再组成矩阵	右除 B/A	XA=B 的解

注：上表中 k 为标量，即单个数值；A 和 B 均为矩阵或数组。

从表 1.6.1 可以看出，部分元素群运算和矩阵运算结果是相同的，另一部分是完全不同的，在应用中要注意。

➤ 元素群的函数

元素群的函数运算规则定义如下：

对于 (m×n) 的二维数组 $X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} = [x_{ij}]_{m \times n}$ ，元素群函数 $f(\quad)$ 的运算规则是指：

$$f(X) = [f(x_{ij})]_{m \times n}$$

也就是说，元素群函数是将函数作用于矩阵中的每一个元素，并将最后的结果另存为与原矩阵行数和列数相同的矩阵。

下面两张表格列出了可以进行元素群函数运算的常用函数。包括：常用的基本数学函数见表 1.6.2，和常用的三角函数见表 1.6.3。

表 1.6.2 MATLAB 常用的基本数学函数

函 数	结 果	函 数	结 果
abs(X)	矩阵 x 中各元素绝对值组成矩阵	rat(X)	矩阵 x 中各元素（实数）转化为分数组成矩阵
angle(X)	矩阵 x 中各元素的相角组成矩阵	sign(X)	矩阵 x 中各元素按符号函数转化为-1 或 0 或 1 组成矩阵
sqrt(X)	矩阵 x 中各元素开平方后组成矩阵	rem(X,Y)	矩阵 x 各元素除以矩阵 y 对应位置元素得到的余数组成矩阵（类似元素群运算）
real(X)	矩阵 x 中各元素的实部组成矩阵	gcd(X,Y)	矩阵 x 与 y 对应元素求最大公约数后组成矩阵（类似元素群运算）
imag(X)	矩阵 x 中各元素的虚部组成矩阵	lcm(X,Y)	矩阵 x 与 y 对应元素求最小公倍数后组成矩阵（类似元素群运算）
conj(X)	矩阵 x 中各元素的共轭复数组成矩阵	exp(X)	矩阵中各元素求 e 指数后组成矩阵
round(X)	矩阵 x 中各元素四舍五入至整数组成矩阵	pow2(X)	矩阵中各元素求以 2 为底的指数后组成矩阵
fix(X)	矩阵 x 中各元素向 0 方向取整数组成矩阵	log(X)	矩阵中各元素求自然对数后组成矩阵
floor(X)	矩阵 x 中各元素用舍去法取整数组成矩阵	log2(X)	矩阵中各元素求以 2 为底的对数后组成矩阵
ceil(X)	矩阵 x 中各元素用进一法取整数组成矩阵	log10(X)	矩阵中各元素求以 10 为底的对数后组成矩阵

表 1.6.3 MATLAB 常用的三角函数

函 数	结 果	函 数	结 果
sin(X)	矩阵 x 中各元素取正弦后组成矩阵	asin(X)	矩阵 x 中各元素取反正弦后组成矩阵
cos(X)	矩阵 x 中各元素取余弦后组成矩阵	acos(X)	矩阵 x 中各元素取反余弦后组成矩阵
tan(X)	矩阵 x 中各元素取正切后组成矩阵	atan(X)	矩阵 x 中各元素取反正切后组成矩阵

- 矩阵运算函数
- 矩阵运算函数在运算后输出矩阵（或数组）的一些属性。表 1.6.4 列出的是当自变量为向量（一维数组）时的一些常用函数和输出结果；表 1.6.5 列出的是当自变量为矩阵时的一些常用函数和输出结果。

表 1.6.4 适用于向量的常用函数

函 数	结 果	函 数	结 果
length(x)	向量 x 的元素个数	norm(x)	向量 x 的欧式长度，也就是范数
min(x)	向量 x 的元素最小值	sum(x)	向量 x 的元素总和
max(x)	向量 x 的元素最大值	prod(x)	向量 x 的元素总乘积
mean(x)	向量 x 的元素平均值	dot(x,y)	向量 x 和 y 的内积
median(x)	向量 x 的元素中位数	cross(x,y)	向量 x 和 y 的外积
std(x)	向量 x 的元素标准差	sort(x)	对向量 x 的元素进行排序

表 1.6.5 适用于矩阵的常用函数

函 数	结 果	函 数	结 果
size(X)	获得矩阵 x 的行数和列数 (m,n)	min(X)	矩阵 x 各列元素最小值组成的行向量
length(X)	获得矩阵 x 的列数	max(X)	矩阵 x 各列元素最大值组成的行向量
ndims(X)	获得矩阵 x 的维数	mean(X)	矩阵 x 各列元素分别求平均值组成的行向量
numel(X)	获得矩阵的元素个数	median(X)	矩阵 x 各列元素中位数组成的行向量
inv(X)	计算矩阵 x 的逆矩阵	std(X)	矩阵 x 各列元素分别求标准差组成的行向量

sum(X)	矩阵 x 各列元素分别求和组成的行向量	sort(X)	对矩阵 x 的各列分别排序组成新矩阵
prod(X)	矩阵 x 各列元素分别求积组成的行向量	norm(X)	矩阵 x 的范数

1.6.3 流程控制

➤ 顺序结构

顺序结构是最简单的程序结构，用户编写好程序之后，系统将按照程序的物理位置从上到下顺序执行。

➤ for 循环结构

当程序需要进行有规律的重复运算时，就需要使用循环程序结构。MATLAB 提供了两种循环方式，即 for 循环和 while 循环。这里先介绍 for 循环。

for 循环的循环判断条件通常就是循环次数。也就是说，for 循环的循环次数是预先设定好的。for 循环的一般调用语法如下：

```
for 循环变量 = 初值 : 步长 : 终值
    循环体语句组
end
```

其中，“初值：步长：终值”代表一个向量。当“初值”<“终值”时，“步长”>0；当“初值”>“终值”时，“步长”<0。当“步长”为 1 时，向量可写作“初值：终值”的形式。另外，还可以直接将一个向量赋给“循环变量”，此时程序进行多次循环，直至穷尽该项量的每一个值。

➤ while 循环结构

与 for 循环不同，while 循环的判断控制是逻辑判断语句，因此，它的循环次数并不确定。while 循环的调用语法如下：

```
while 关系表达式
    循环体语句组
end
```

在这个循环中，只要关系表达式的值不为 false，程序就会一直运行下去。通常在执行语句中要有使表达式值改变的语句。必须注意的是，当程序设计出了问题，导致关系表达式总是 true，程序就陷入死循环。

➤ if 条件选择结构

在编写程序时，往往要根据一定的条件进行判断，然后选择执行不同的语句。此时需要使用判断语句来进行流程控制。

条件选择结构中最基本的是 if 条件选择结构语句。if 条件选择结构有 3 种基本格式：

(1)

```
if 条件表达式
    条件语句组
end
```

(2)

```
if 条件表达式
    条件块语句组 1
else
    条件块语句组 2
end
```

(3)

```
if 条件表达式 1
    条件块语句组 1
elseif 条件表达式 2
    条件块语句组 2
...
elseif 条件表达式 n-1
    条件块语句组 n-1
else
```

条件块语句组 n

end

对于第(3)种情况,如果程序运行到的某一条表达式为 true,则执行响应的语句。此时系统不再对其它表达式进行判断,即系统将直接跳到 end。另外,最后的 else 也可以不用,如果使用,则必须与 if 配对使用。需要注意的是,elseif 是连着写的,中间没有空格,如果写成 else if,那么系统会认为这是一个嵌套的 if 语句,这时每一个 if 都需要与 end 配对。

➤ switch 条件选择结构

在 MATLAB 语言中除了上面介绍的 if 条件选择结构外,还提供了另外一种条件选择结构,那就是 switch 条件选择结构,结构如下:

switch 开关语句

case 条件语句 1

执行语句组 1

case 条件语句 2

执行语句组 2

...

case 条件语句 n

执行语句组 n

otherwise

执行语句组 n+1

end

在 switch 分支结构中,如果某个条件语句的结果与开关语句的内容相匹配,系统将执行其后的语句;如果所有的条件语句与开关语句都不相符合,系统将执行 otherwise 后面的语句。和 C 语言不同的是,switch 语句中如果某一个 case 中的条件语句与开关语句匹配成功,则其它的 case 将不会再继续执行,程序将直接跳至 switch 语句结尾,故不需要 break。

【例1.6-2】使用 switch...case...end 语句。

```
switch var
```

```
case 1
```

```
% 判断var是不是 1
```

```
disp('1')
```

```
case {2,3,4}
```

```
% 判断var是不是 2,3,4
```

```
disp('2 or 3 or 4')
```

```
case 5
```

```
% 判断var是不是 5
```

```
disp('5')
```

```
otherwise
```

```
% 其它情况
```

```
disp('something else')
```

```
end
```

➤ continue 命令

continue 命令经常与 for 或 while 循环语句一起使用,作用是结束本次循环,即跳过循环体中尚未执行的语句,接着进行下一次循环。该命令的调用方法为:

```
continue
```

➤ break 命令

MATLAB 中 break 语句用于终止 for 或 while 循环的执行,当在循环体内执行到该语句的时候,程序将会跳出循环,继续执行循环语句下面的语句。该命令的调用方法为:

```
break
```

1.6.4 绘图

MATLAB 除了强大的数值分析功能外,还具有方便的绘图功能。利用 MATLAB 丰富的二维、三维图形函数和多种修饰方法,就可以绘制出理想的图形。这里只介绍 MATLAB 二维图形的绘制。

➤ 基本绘图函数

MATLAB 中最常用的绘图函数为 plot() 和 stem(), 它们用于绘制二维曲线的。其中, plot() 用来绘制连续曲线; stem() 用来绘制离散火柴杆图。根据函数输入参数不同, 常用的几种调用格式如表1.6.6 所示。其中, 'option'用来设置曲线属性的选项, 其内容主要包括诸如颜色、线型、标记类型等曲线属性。'option'选项并不是必须项, 若缺少该项, MATLAB 将按系统默认格式统一安排各条曲线的属性值。

表1.6.6 绘图函数plot()与stem()的常用调用格式

函数调用格式	说 明
plot(x,y,'option') stem(x,y,'option')	以向量x为横坐标, 向量y为纵坐标, 按属性选项'option'绘制连续曲线(plot)或离散火柴杆图(stem)。
plot(x1,y1,'option',x2,y2,'option',...) stem(x1,y1,'option',x2,y2,'option',...)	分别以向量x1, x2, ...为横坐标, 以向量y1, y2, ...为纵坐标绘制多条连续曲线(plot)或多组离散火柴杆图(stem)。每条曲线或火柴杆图的属性由相应的选项'option'确定。

MATLAB 提供了三种 'option' 选项以供选择: Line style 线性类, Marker symbol 标记符号, Color 颜色。表1.6.7 列出了 'option' 选项的可选属性。

表1.6.7 'option' 选项的可选属性列表

符 号	Line style	符 号	Marker symbol	符 号	Color
'-'	实线 (默认)	'+'	加号	'r'	红色
'--'	虚线	'o' (字母o)	圆圈	'g'	绿色
'.'	点线	'*'	星号	'b'	蓝色 (默认)
'-.'	点划线	'.'	点	'c'	蓝绿色
		'x'	叉号	'm'	洋红色
		'square' 或 's'	方块	'y'	黄色
		'diamond' 或 'd'	钻石	'k'	黑色
		'^'	向上的三角	'w'	白色
		'v'	向下的三角		
		'>'	向右的三角		
		'<'	向左的三角		
		'pentagra' 或 'p'	五角星		
		'hexagram' 或 'h'	六角星		

'option' 选项使用时可以把三种属性一起设置, 如 '-*r', 三种属性可以列1项、2项或3项, 三项的顺序可以调换, 且所有属性选项必须在一个单引号内。需要注意的是, 在使用 plot 函数绘图时, 若在属性设置中设置了Marker symbol (标记符号) 而省略了Line style (线性类) 的设置, 会得到离散的点图, 若想得到连线图还需设置线类型。

➤ 图形修饰

有时用户会对图形的绘图进行一些修饰, MATLAB 提供了多种图形函数, 用于图形的修饰。常用的图形修饰函数名称及其功能说明如表1.6.8所示。

表1.6.8 常用图形修饰函数名称及其功能说明

函 数	功能说明
Axis([Xmin,Xmax,Ymin,Ymax])	x、y坐标轴范围的调整
xlim([Xmin,Xmax])	x坐标轴范围的调整
ylim([Ymin,Ymax])	y坐标轴范围的调整
title('string')	标注图形标题为 string 字符串
xlabel('string')	标注x轴名称为 string 字符串
ylabel('string')	标注y轴名称为 string 字符串
legend('string1','string2',...)	标注图例标注为 string1、string2... 字符串
grid on	给图形增加网格
grid off	给图形取消网格
gtext('string')	在图形中加入普通文本标注

表1.6.8 中所列的修饰函数应用时应该放在所修饰的 plot 语句或 stem 语句的后面。

➤ 图形窗口控制

MATLAB 提供了一系列专门的图形窗口控制函数，通过这些函数，可以创建或者关闭图形窗口，还可以同时打开几个窗口。这些函数及功能说明如表 1.6.9 所示。

表1.6.9 MATLAB图形窗口控制函数及其功能说明

函 数	功能说明
figure	每调用一次就打开一个新的图形窗口
figure(n)	创建或打开第n个图形窗口，使之成为当前窗口
hold on	保留当前窗口的图形不被后继图形覆盖，可实现在同一坐标系中多幅图形的重叠
hold off	解除 hold on 命令，一般与 hold on 成对使用
close	关闭当前图形窗口
close all	关闭所有图形窗口

关于 close 函数，如果要关闭当前的图形窗口，可以用 “close” 命令；要关闭第 n 个图形窗口，可以用 “close figure n” 命令，或 “close (figure(n))” 命令；要关闭所有图形窗口，用 “close all” 命令。

注意，如果想在多幅图中显示图像，每次使用绘图命令 plot 或 stem 之前，都要先使用 figure(n) 命令，以打开第 n 个图形窗口；如果要在一张图中重叠显示多个函数，则需要使用 hold on 和 hold off，把所有的绘图命令放置于hold on 和 hold off 语句之间即可。具体的，第一个绘图命令运行后，将自动创建一个名为 Figure 1 的图形窗口，这个窗口将被当做当前窗口，接下来的所有绘图命令（包括绘图修饰和再一次的绘图指令）均在该图形窗口中执行，后续绘图指令会覆盖原图形或者叠加在原图形上。在 MATLAB 2013a 版本中，若在同一图形窗口中多次绘图，系统会自动更改颜色以区分结果。

【例1.6-3】取三个不同的 t，t1 = 0 : 0.025 : 0.5，t2 = 0.25 : 0.025 : 0.75，t3 = 0.5 : 0.025 : 1，在同一坐

标系下绘制 $y_1 = \sin(2\pi f t_1)$ ， $y_2 = \sin(2\pi f t_2 - \pi/2)$ ， $y_3 = \sin(2\pi f t_3 - \pi)$ ，其中 $f = 1\text{ Hz}$ ，具体要求如下：

- (1) 将图形的 x 轴大小范围限定在[0，1]之间，y 轴的大小范围限定在[-1，1]之间；
- (2) x、y 轴分别标注为“时间（s）”、“函数值”；
- (3) 图形标题标注为“三个不同相位的正弦曲线”；
- (4) 添加图例标注，标注字符分别为 y1，y2，y3；
- (5) 给三条曲线分别添加属性 ‘-.r*’， ‘--mo’， ‘: bs’；
- (6) 在三条曲线上分别标注文本 $y_1 = \sin(0.5\pi t_1)$ ， $y_2 = \sin(0.5\pi t_2 - 0.5\pi)$ ， $y_3 = \sin(0.5\pi t_3 - \pi)$ ；
- (7) 给图形添加网格。

编写脚本M文件如下：

```
clc; % 清空命令窗口
clear; % 清空Workspace变量存储空间
close all; % 关闭所有图形窗口
% 图形绘制
figure(1)
f=1;
t1=0:0.025:0.5;
t2=0.25:0.025:0.75;
t3=0.5:0.025:1;
y1=sin(2*pi*t1);
y2=sin(2*pi*t2-pi/2);
y3=sin(2*pi*t3-pi);
hold on
plot(t1,y1,'-.r*');
plot(t2,y2,'--mo');
```



```

plot(t3,y3,':bs');
hold off
% 图形修饰
axis([0,1,-1,1]);
xlabel('时间 (s) ');
ylabel('函数值');
title('三个不同相位的正弦曲线');
legend('y1','y2','y3');
grid on;
gtext('y1=sin(0.5πt)');
gtext('y2=sin(0.5πt-0.5π)');
gtext('y3=sin(0.5πt-π)');

```

程序运行结果如图1.6.1所示。

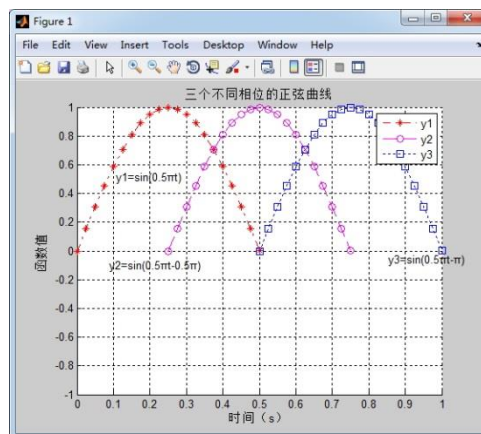


图1.6.1 例1.6-3绘图结果

➤ 子图

MATLAB 允许用户在同一个图形窗口内布置几幅独立的子图，具体调用语法如下。

- `subplot(m,n,p)`: 将当前绘图窗口分割成 m 行、 n 列，并在第 p 个区域内绘图。子图的编号顺序是左上为第 1 幅，向右向下依次排序。
- `subplot('Position',[left bottom width height])`: 在指定位置上绘制子图，并成为当前图。其中，`left` 代表子图左边框距离窗口左边框相对距离，窗口总宽为1，`left` 为0到1之间的相对比值。`bottom` 代表子图下边框距离窗口下边框相对距离，窗口总高为1，`bottom` 为0到1之间的相对比值。`width`, `height` 也都为0到1之间相对值，代表子图相对宽度和相对高度。

【例1.6-4】 调用 `subplot` 函数绘制子图

```

clc;
clear;
close all;
t=0:0.001:0.5;
f1=1;
f2=10;
y1=sin(2*pi*f1*t);
y2=sin(2*pi*f2*t);
y12=y1.*y2;
subplot(2,2,1),plot(t,y1);
axis([0,0.5,-1,1]);
subplot(2,2,2),plot(t,y2);
axis([0,0.5,-1,1]);
subplot('position',[0.2,0.05,0.6,0.45]),
plot(t,y12,'b-',t,[y1;-y1],'r:');

```

```
axis([0,0.5,-1,1]);
```

以上代码的运行结果如图 1.6.2 所示。

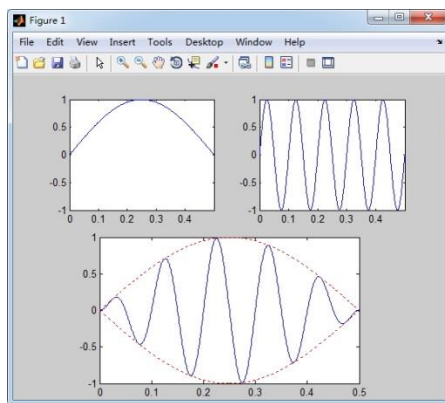


图1.6.2 例1.6-4中绘制的子图

1.6.5 信号分析常用信号表示

➤ 方波信号

可使用函数 `square`，具体使用方法为：`square(w*t+b)`，可对比函数 `sin(w*t+b)`，进而发现两者的区别与联系，其中 t 代表自变量数组， w 代表角频率， b 代表初始相位。

➤ 方波脉冲信号

可使用函数 `rectpuls`，具体使用方法为：`rectpuls(t-a,b)`，其中 t 代表自变量数组， a 代表方波脉冲的中心位置横坐标， b 代表脉冲宽度。绘制的方波脉冲幅度为 1。

➤ 单位脉冲

可使用函数 `rectpuls`，具体使用方法为：`rectpuls(n-a,0)`，其中 n 代表自变量数组， a 代表单位脉冲的横坐标位置。

➤ 单位阶跃

可使用函数 `stepfun`，具体使用方法为：`stepfun(n,a)` 或 `stepfun(t,a)`，其中 n 或 t 代表自变量数组， a 代表跳变点横坐标。

➤ 三角波脉冲信号

可使用函数 `tripuls`，具体使用方法为：`tripuls(t-a,b)`，其中 t 代表自变量数组， a 代表三角波脉冲的中心位置横坐标， b 代表脉冲宽度。绘制的三角波脉冲幅度为 1。

1.7 GUI 设计

1.7.1 GUI 简介

图形用户界面（Graphical User Interface，简称 GUI，又称图形用户接口）是一种用户与计算机通信的直观显示方式，该方式包含图形对象的交互界面，如窗口、图标、菜单和文本等。与早期计算机使用的命令行界面相比，图形界面对于用户来说在视觉上更易于接受。

图形用户界面允许用户使用鼠标等输入设备操纵屏幕上的图标或菜单选项，以选择命令、调用文件、启动程序或执行其它一些日常任务。

MATLAB 为用户开发图形界面提供了一个方便的集成开发环境：MATLAB 图形用户界面开发环境（MATLAB Graphical User Interface Development Environment），简称 GUIDE。GUIDE 主要是一个界面设计工具集，MATLAB 将所有 GUI 支持的用户控件都集成起来，同时提供界面外观、属性和行为回调（CallBack）的设置方法。

下面将通过一个具体的图形化程序设计（图形化显示正弦和方波曲线）说明 MATLAB GUI 的一些基本用法，包括主菜单、弹出式菜单、下拉菜单、命令按钮的创建和使用。

1.7.2 打开 Layout 编辑器

启动 GUI，可以单击 New | Graphical User Interface 菜单，即可打开 GUIDE Quick Start 对话框，如图 1.7.1 所示。利用此对话框可以创建新的 GUI（选 Create New GUI），或者打开已有的 GUI（选 Open Existing GUI）。

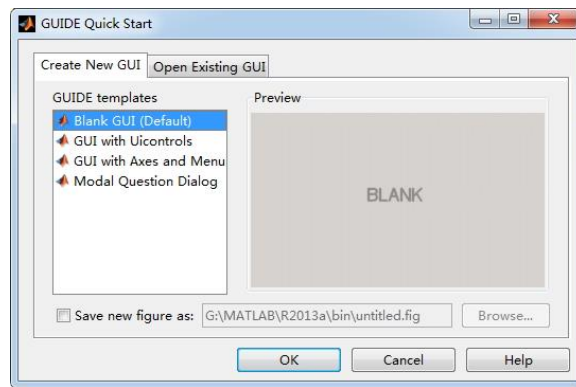


图 1.7.1 GUIDE Quick Start 对话框

这里我们选择 Blank GUI (Default) 选项，新建一个 GUI，该 GUI 将显示在 Layout 编辑器中，如图 1.7.2 所示。用户可以使用鼠标拖动模板左边的控件（按钮、坐标轴、列表框等）到中间的设计区域，在 GUI 编辑区域右下角，可以通过鼠标拖拽的方式改变 GUI 的大小。默认情况下控件面板中的控件只显示图标，不显示名称。可通过点击 Layout 编辑器中 File | Preferences 命令弹出显示对话框，然后选中 Show names in component palette 复选框，以便显示出控件名称，如图 1.7.3 的控件面板中所示。

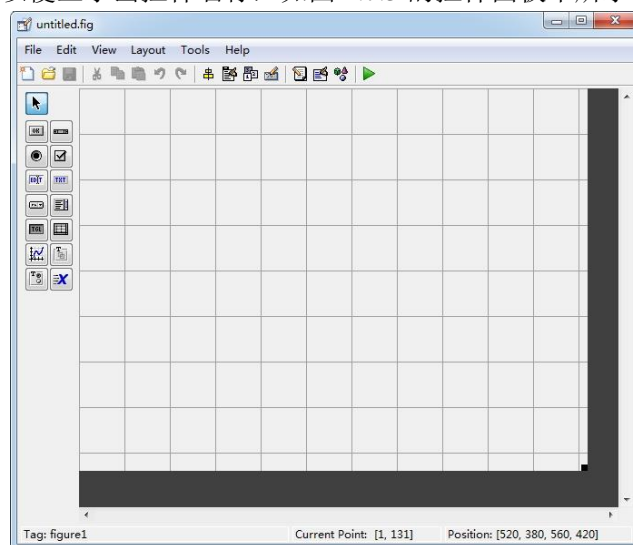


图 1.7.2 Layout 编辑器

1.7.3 添加控件

下面在 Layout 编辑器中为 GUI 添加控件，从左侧控件面板中拖拽一个轴对象（Axes）到设计区域中，用来显示函数图像。使用同样的方法拖拽两个按钮控件（Push Button）和一个下拉列表控件（Pop-up Menu）到设计区域，设计结果如图 1.7.3 所示。

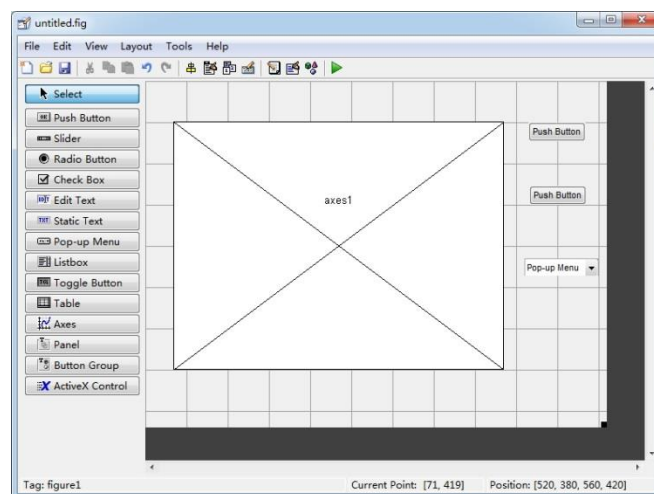


图 1.7.3 添加控件

1.7.4 设置按钮控件图形对象

接下来，改变两个按钮控件的标签分别为 thick 和 thin，代表曲线加粗和变细。通过以下步骤来实现标签设置。

右键选中需要更改标签的命令按钮，选中 **Property Inspector** 选项，在弹出的属性窗口中设置 **String** 属性为需要的标签内容，两个按钮控件的标签更改前均为 **Push Button**，需要分别改为 **thick** 和 **thin**，如图 1.7.4 所示，代表的是改为 **thick** 的情况。

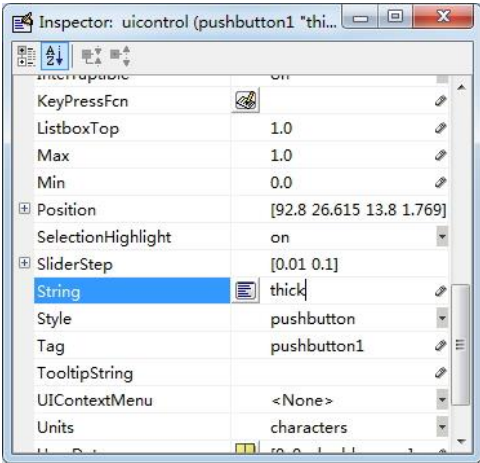



图 1.7.4 设置标签

1.7.5 设置下拉菜单图形对象

还需设置下拉菜单的列表项，本例中的下拉列表项需要设置为 **blue**、**red** 和 **green** 三项，代表曲线的不同颜色标签。首先，右键选中下拉列表控件，选中 **Property Inspector** 选项。然后，单击 **String** 属性旁边的按钮 ，弹出 **String** 对话框，如图 1.7.5 所示。将现有的 **Pop-up Menu** 替换为 **blue**、**red** 和 **green**，三个选项需要排列为三行，每一行是下拉列表的一个选项，设置的结果如图 1.7.6 所示。

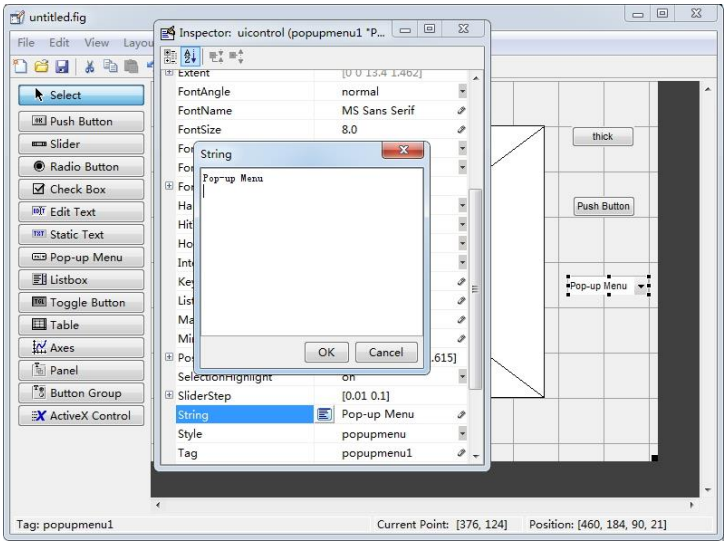


图 1.7.5 下拉列表项设置

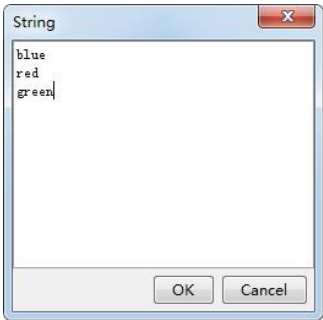


图 1.7.6 下拉列表项设置结果

1.7.6 设置主菜单图形对象

单击 **Layout** 编辑器中的 **Tools | Menu Editor** 菜单命令，会弹出 **Menu Editor** 窗口，可以给 **GUI** 增加主菜单和弹出式菜单。先进行主菜单的设置，功能是改变曲线的对应函数，选择 **Menu Editor** 窗口中的 **Menu Bar** 选项卡，单击左侧选项卡窗口，选中 **Untitled 1**，如图 1.7.7 所示。

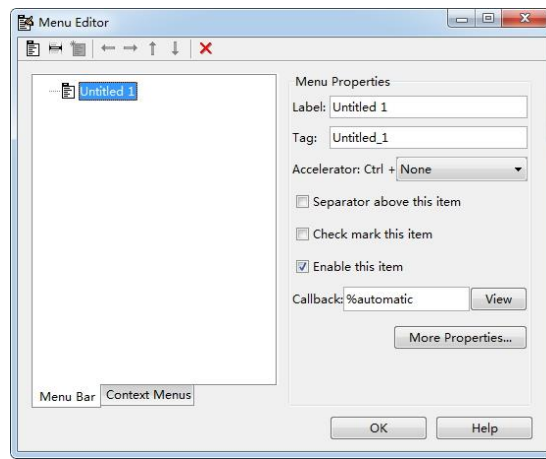



图 1.7.7 Menu Editor 窗口

更改菜单的 Label、Tag 属性，Label 为菜单显示的标签名，Tag 为代码程序中引用该菜单的名称。设置 Label 的目的是为了在运行图形化程序时便于操作；设置 Tag 的目的是为了增加程序代码的可读性。这里把主菜单项的 Label 由 Untitled 1 改为 Function，Tag 改为 Main_Menu。进一步，通过点击 Menu Editor 窗口工具栏中的  图标，给当前菜单增添子菜单项。这里增加两个子菜单：第一个设置 Label 和 Tag 属性为 sine 和 Main_Menu_sine；第二个设置 Label 和 Tag 属性为 square signal 和 Main_Menu_square，如图 1.7.8 所示。

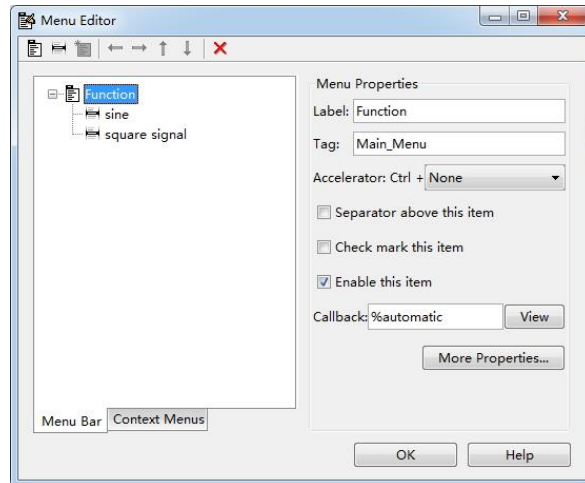


图 1.7.8 给 GUI 增添主菜单项及其附属子菜单项

1.7.7 设置弹出式菜单图形对象

再进行弹出式菜单的设置，功能是改变曲线的线型，选择 Menu Editor 窗口中的 Context Menus 选项卡，然后参考上文给主菜单添加子菜单项的方法来给弹出式菜单添加菜单项。此处弹出式菜单主体只包含 Tag 属性，设置为 LineStyle；弹出式菜单的子菜单共三项，包含 Label 和 Tag 属性，第一项设置为 dashed 和 Context_Menu_dashed，第二项设置为 dotted 和 Context_Menu_dotted，第三项设置为 solid 和 Context_Menu_solid，如图 1.7.9 所示。

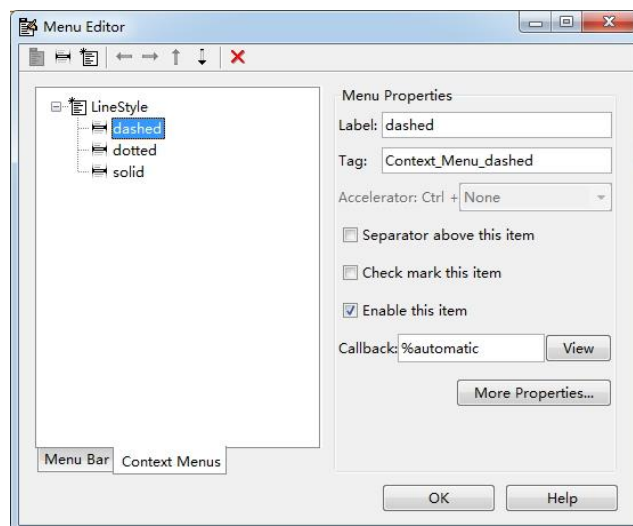


图 1.7.9 给 GUI 增添弹出式菜单项及其附属子菜单项

对于弹出式菜单，还需要确定在哪个控件中点击右键后弹出菜单，这里设置在轴对象（Axes）中点击右键弹出菜单。在 Layout 编辑器中，右键选中轴对象（即运行后显示图像的区域对象），选中 Property Inspector 选项，在对应 UIContextMenu 选项中选择刚才设置的弹出式菜单的 Tag 属性，即 LineStyle，如图 1.7.10 所示。这样就完成了弹出式菜单与轴对象的链接。

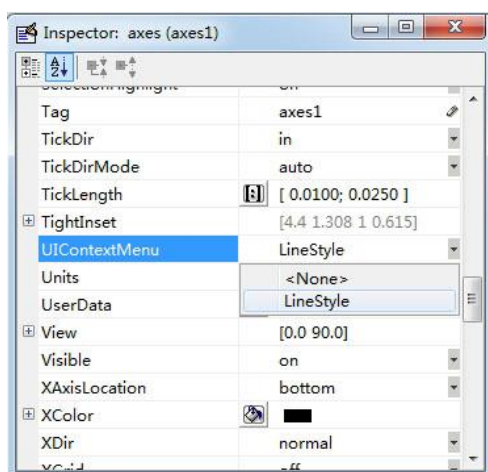


图 1.7.10 将弹出式菜单与对象链接

1.7.8 完成布局设计并保存

通过上面的操作，可以得到图 1.7.11 所示的结果，通过菜单或者工具栏可以进行保存，这里将文件名改为 SimpleGUI.fig。

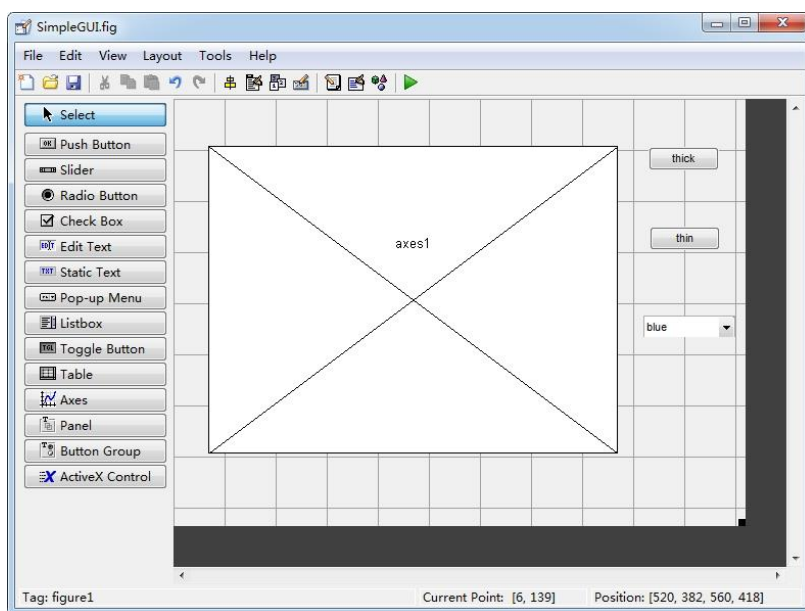


图 1.7.11 布局设计结果

1.7.9 添加 M 文件代码

保存 GUI 布局设置之后，GUIDE 会创建两个文件：SimpleGUI.fig 和 SimpleGUI.m。保存后，MATLAB 会自动将保存的 M 文件打开。其中 SimpleGUI.fig 保存的是 GUI 的布局设计，而 SimpleGUI.m 保存的是控制 GUI 动作的代码。之前的设计并没有完成代码，这样运行 GUI 只能得到一个图形窗口，按钮等控件没有任何功能，为此需要向 M 文件中添加相应的代码。

1.7.10 生成绘图数据

本例中的 GUI 通过主菜单选择不同的函数图形，通过其他控件选择函数图形的不同属性，而绘图数据是在打开函数（OpeningFcn）中产生的。打开刚才的 SimpleGUI.m 文件，可以看到系统已经生成了程序代码的基本框架，只需在框架内填入具体的代码。找到 SimpleGUI_OpeningFcn 子函数，可以看到该函数中已经有一下内容。

```
% --- Executes just before SimpleGUI is made visible.
function SimpleGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to SimpleGUI (see VARARGIN)

% Choose default command line output for SimpleGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SimpleGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

```

然后在 % varargin 行下面添加如下代码。

```

handles.x = -2:0.01:2;
handles.sine = sin(2*pi*handles.x);
handles.square = square(2*pi*handles.x);
handles.current_data = handles.sine;
handles.current_Color = 'b';
handles.current_LineStyle = '--';
handles.current_LineWidth = 1;
line = plot(handles.x, handles.current_data);
ylim([-2,2]);
set(line, 'Color', handles.current_Color);
set(line, 'LineStyle', handles.current_LineStyle);
set(line, 'LineWidth', handles.current_LineWidth);

```

代码中出现的 handles 为一个结构体数组，这里还要引入一个概念：“句柄”。MATLAB 在创建每一个图形对象时，都会为该对象分配唯一的值，称其为图形对象句柄。handles 和 hObject 都是句柄。可以简单的把 handles 句柄理解为特殊的结构体变量，用来指代 GUI 中所有组件，handles 即是当前整个界面（整个程序）所有的句柄。由于 handles 为该 GUI 的整体句柄，所以在程序中的每一个函数或回调函数都可以引用和保存 handles 句柄，相当于全局变量，起到在各函数之间变量传递和访问 GUI 数据的作用。而 hObject 句柄可以理解为一个程序入口，相当于局部变量，用来指代当前的这个控件（比如按钮，或编辑框等），hObject 标识了当前控件的存储空间。在当前控件回调函数中出现的 hObject 句柄只能表示当前控件，在另一个控件回调函数中出现的 hObject 句柄代表另一个控件，类似 C 语言中不同作用域中使用相同的变量名称，虽然变量名称相同，但由于作用域不同，为不同变量。

比如在本 GUI 中，Label 为 thick 的按钮，其 Tag 为 pushbutton1，故 handles.pushbutton1 代表该按钮的唯一句柄标识，与本按钮回调函数作用范围内的 hObject 句柄相同作用。使用 handles 句柄可以按照 handles.Name 的方式引用，其中 Name 可以是特定控件的 Tag 值，如前文所述。Name 还可以是自定的变量名，在一个函数中自定义的 handles.Name 句柄属性，也可以在另一个函数中引用（全局变量）。

在上文添加的代码中，handles.x、handles.sine、handles.square、handles.current_data、handles.current_Color、handles.current_LineStyle、handles.current_LineWidth 都是自定义的句柄属性。前七行生成绘图所需要的数据，并完成了绘图区图形属性的初始化，第八行 plot 函数完成绘图，后三行完成对所绘图行的颜色、线型、线粗的设定。这里用到了 set 函数，用于设置某句柄对应图形对象的属性值。即 set(h, 'PropertyName', PropertyValue)，设置对象句柄 h 的 PropertyValue 属性为 PropertyValue。这里先用 plot 函数完成绘图，即创建了一个图形对象，将该对象的句柄保存赋值给 line，之后调用 set 函数完成对该句柄对应图形对象的属性设置。此处 line 的作用域只限于当前函数。

在 GUIDE 自动生成的代码中，语句 guidata(hObject, handles) 完成对改变后的 handles 结构数组的保存，这样这些数据就可以被所有的回调函数调用。其中句柄 hObject 是为了找到特定的储存空间。SimpleGUI_OpeningFcn 函数头部中需要传入参数 handles，传入的 handles 值可以理解为实参。在函数中对 handles 进行赋值的语句中，函数体内部的 handles 可以理解为形参。语句 guidata(hObject, handles)

的作用就是在此子函数结束时用形参改变实参，即保存对全局变量的更改。

语句 `handles.output = hObject` 和 `SimpleGUI_OutputFcn` 子函数中语句 `varargout{1} = handles.output` 的作用是在不同 GUI 之间传递参数，本例只涉及一个 GUI，可以不予理会，在这里我们也不再更详细的说明。

1.7.11 编写主菜单程序

主菜单中包含两个子菜单，分别是绘制正弦曲线和方波函数曲线。绘制正弦曲线的程序位于 `Main_Menu_sine_Callback` 函数中：

```
function Main_Menu_sine_Callback(hObject, eventdata, handles)
% hObject    handle to Main_Menu_sine (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

在后面添加代码如下：

```
handles.current_data = handles.sine;
line = plot(handles.x,handles.current_data);
ylim([-2,2]);
set(line, 'Color', handles.current_Color);
set(line, 'LineStyle', handles.current_LineStyle);
set(line, 'LineWidth', handles.current_LineWidth);
guidata(hObject, handles);
```

加入代码的第一行将图形函数设置为正弦函数。

同样，在函数 `Main_Menu_square_Callback` 最后添加如下代码：

```
handles.current_data = handles.square;
line = plot(handles.x,handles.current_data);
ylim([-2,2]);
set(line, 'Color', handles.current_Color);
set(line, 'LineStyle', handles.current_LineStyle);
set(line, 'LineWidth', handles.current_LineWidth);
guidata(hObject, handles);
```

以上我们对主菜单控件完成了代码的编写，所写的代码是在选取主菜单中相应子菜单时才运行，所以我们称这种触发控件后执行的代码为回调函数。

1.7.12 编写弹出式菜单程序

找到 `Context_Menu_dashed_Callback` 函数体。可通过单击 **M** 文件编辑器工具栏中的 **Go To** 按钮来定位函数的位置，如图 1.7.12 所示。

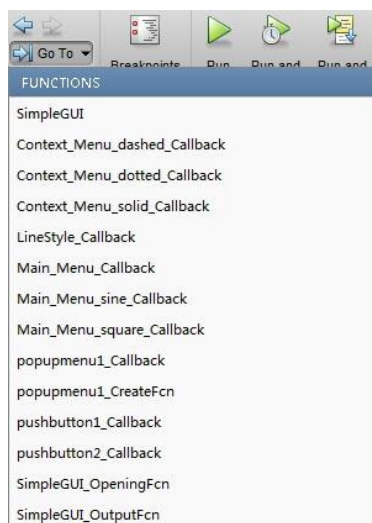


图 1.7.12 回调函数定位

打开 `Context_Menu_dashed_Callback` 函数体后，在末尾添加代码如下：

```
line = plot(handles.x,handles.current_data);
```

```
ylim([-2,2]);
handles.current_LineStyle = '--';
set(line, 'Color', handles.current_Color);
set(line, 'LineStyle', handles.current_LineStyle);
set(line, 'LineWidth', handles.current_LineWidth);
guidata(hObject, handles);
```

同理，在 Context_Menu_dotted_Callback 函数体末尾添加代码如下：

```
line = plot(handles.x,handles.current_data);
ylim([-2,2]);
handles.current_LineStyle = ':';
set(line, 'Color', handles.current_Color);
set(line, 'LineStyle', handles.current_LineStyle);
set(line, 'LineWidth', handles.current_LineWidth);
guidata(hObject, handles);
```

在 Context_Menu_solid_Callback 函数体末尾添加代码如下：

```
line = plot(handles.x,handles.current_data);
ylim([-2,2]);
handles.current_LineStyle = '-';
set(line, 'Color', handles.current_Color);
set(line, 'LineStyle', handles.current_LineStyle);
set(line, 'LineWidth', handles.current_LineWidth);
guidata(hObject, handles);
```

1.7.13 编写按钮的回调函数

在 pushbutton1_Callback 函数体末尾添加代码如下：

```
line = plot(handles.x,handles.current_data);
ylim([-2,2]);
handles.current_LineWidth = 4;
set(line, 'Color', handles.current_Color);
set(line, 'LineStyle', handles.current_LineStyle);
set(line, 'LineWidth', handles.current_LineWidth);
guidata(hObject, handles);
```

在 pushbutton2_Callback 函数体末尾添加代码如下：

```
line = plot(handles.x,handles.current_data);
ylim([-2,2]);
handles.current_LineWidth = 1;
set(line, 'Color', handles.current_Color);
set(line, 'LineStyle', handles.current_LineStyle);
set(line, 'LineWidth', handles.current_LineWidth);
guidata(hObject, handles);
```

1.7.14 编写下拉菜单程序

在 popupmenu1_Callback 函数体末尾添加代码如下：

```
str = get(hObject, 'String');
val = get(hObject, 'Value');
switch str{val};
    case 'blue'
        handles.current_Color = 'b';
    case 'red'
        handles.current_Color = 'r';
```

```

case 'green'
    handles.current_Color = 'g';
end
line = plot(handles.x,handles.current_data);
ylim([-2,2]);
set(line, 'Color', handles.current_Color);
set(line, 'LineStyle', handles.current_LineStyle);
set(line, 'LineWidth', handles.current_LineWidth);
guidata(hObject, handles);

```

这里前两句用到了get语句,用法为get(h, 'PropertyName'),即返回对象句柄h的PropertyName属性的值。第一句得到的是当前图形控件 (popupmenu1) 的'String'属性, 这个属性是之前编辑过的, 分为三行: blue、red、green。在这里 str 便是具有这三行文本的元组数据。第二句得到的是当前图形控件的'Value'属性, 并赋值给val。当被选中的是 blue 时, val 为 1; 当被选中的是 red 时, val 为 2; 当被选中的是 green 时, val 为 3。第三句利用switch语句进行选择, 其中 str{val} 为取出str元组中序号为val的元素。

1.7.15 运行GUI

通过以上操作, 本例的 GUI 设计完成。保存后, 运行 fig 或者 M 文件显示如下图 1.7.13 所示界面:

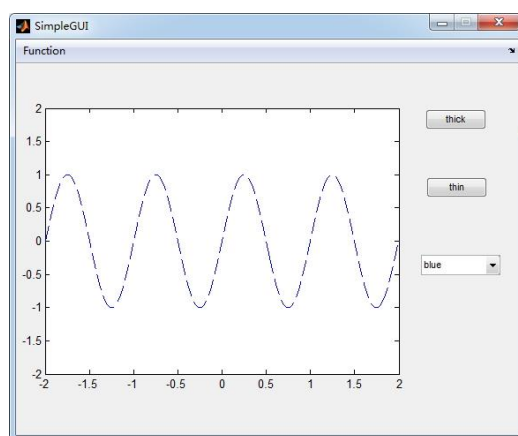


图 1.7.13 GUI 设计结果

可以通过点击左上角的主菜单 **Function** 选择不同的函数曲线; 可以通过点击两个按钮改变曲线的粗细; 可以通过点击右下角的下拉菜单选择不同的曲线颜色。

最后可以在曲线显示窗口内点击右键, 在弹出的菜单中选择不同的线型, 如图 1.7.14 所示。

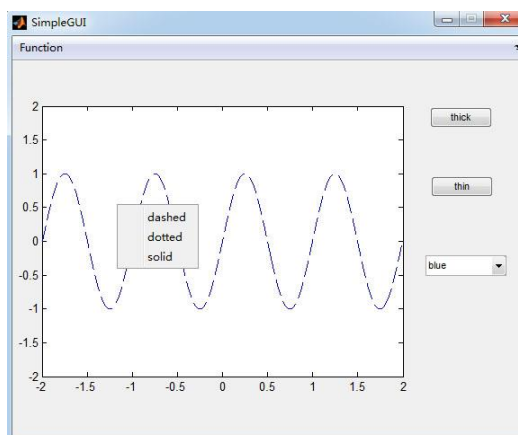


图 1.7.14 利用弹出式菜单进行线型选择

这里将 GUI 的 FIG 文件和 M 文件做一个总结:

- **FIG 文件:** 该文件包含 GUI 及其子对象的完全描述, 包含所有相关对象的属性信息。FIG 文件是一个二进制文件, 它包含系列化的图形窗口对象, 该文件最主要的功能是保存对象句柄。
- **M 文件:** 该文件包含 GUI 设计、控制函数及控件的回调函数。该文件基本上可以分为 GUI 初始

化和回调函数两个部分，控件的回调函数根据用户与 GUI 的具体交互行为来选择调用。该文件不包含用户界面设计的代码，对应的代码由 FIG 文件保存。

1.7.16 MATLAB GUI 封装成 .exe 可执行文件

前面的可视化程序的运行需要 SimpleGUI.fig 和 SimpleGUI.m 两个文件，而且必须在 MATLAB 的程序环境下运行。为了使程序能够脱离 MATLAB 程序环境在 Windows 下单独运行，需要将 GUI 程序封装成 .exe 可执行文件，即生成一个 SimpleGUI.exe 独立运行文件。

首先，在 MATLAB 的命令窗口输入 deploytool，回车，弹出窗口如下图 1.7.15 所示，可设置 Name 为 SimpleGUI.prj，然后点 OK。

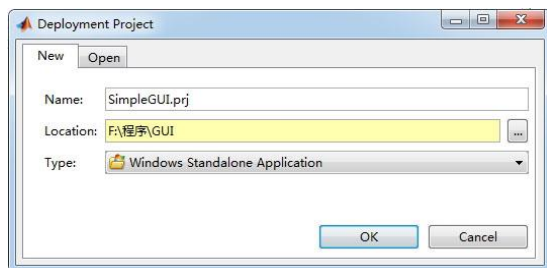



图 1.7.15 打开 Deployment Project

弹出如图 1.7.16 所示窗口，点击 Add main file，选择 SimpleGUI.m 文件，点击“打开”，点击 Windows Standalone Application 窗口的 Build 按钮 ，开始生成可执行文件。

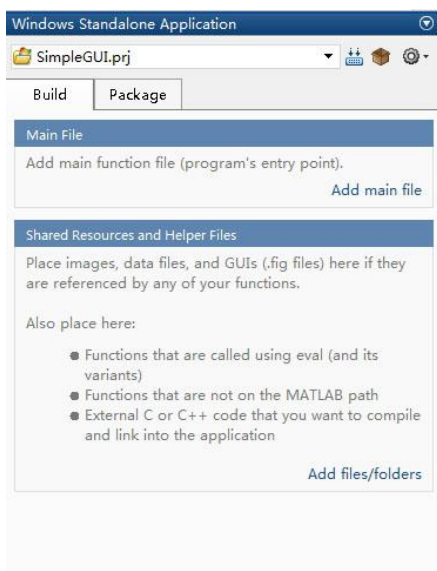


图 1.7.16 Windows Standalone Application 窗口

等待可执行文件生成完成，如下图 1.7.17 所示：

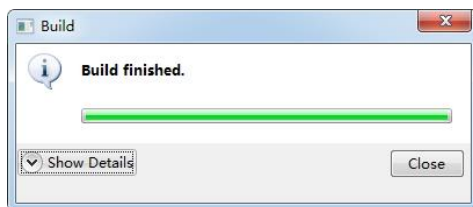


图 1.7.17 Build 窗口

至此，在文件夹 SimpleGUI 中的 distrib 子文件夹中可以找到 SimpleGUI.exe 文件，可以在 Windows 下独立运行。

1.8 Simulink 仿真

1.8.1 Simulink 简介


Simulink 是 MATLAB 环境下一个进行动态系统建模、仿真和综合分析的集成软件包。它可以处理：线性、非线性系统；离散、连续及混合系统等。Simulink 已经成为学术和工业领域中在对工程问题的模型化

及动态模拟仿真方面应用最广的软件包。

作为 MATLAB 的一个工具箱，Simulink 具有良好的图形交互界面，体现了模块化设计和系统级仿真的思想。用户只要进行鼠标的简单拖拽操作就可以构造出复杂的仿真模型。其外表以方块图形呈现，模块间连接数据线，使得建模仿真如同搭积木一样简单。

1.8.2 Simulink 的启动

在启动 Simulink 软件包之前，首先要启动 MATLAB 软件。在 MATLAB 中有以下两种启动 Simulink 浏览器的方法。

- 单击工具栏上的 Simulink Library 按钮。
- 在命令行中键入 simulink。

操作后，会弹出 Simulink Library Browser（Simulink 模型库浏览器），如图 1.8.1 所示。单击此浏览器左侧的模块组，在右侧就会显示该模块组内的所有模块。在图 1.8.1 左侧模块组列表表中可以看到第一个模块组库 Simulink，在 Simulink 模块库下包了丰富的模块组，表 1.8.1 表示的是 Simulink 中所包含的基本模块组子库。如果知道使用模块的名称，还可以在模型库浏览器上方的搜索框里搜索。

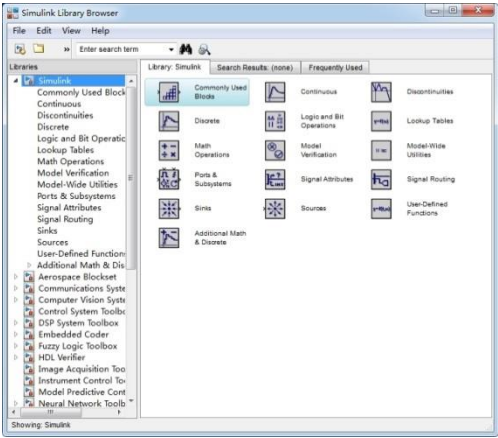



图 1.8.1 Simulink Library Browser 界面

表 1.8.1 Simulink 模块库中基本模块组子库列表

模块组子库	对应英文名
常用模块组	Commonly Used Blocks
连续模块组	Continuous
非连续模块组	Discontinuties
离散模块组	Discrete
逻辑与二进制操作模块组	Logic and Bit Operations
寻表操作组	Lookup Tables
数学运算模块组	Math Operations
模型验证操作模块组	Model Verification
公用模块组	Model-Wide Utilities
端口与子系统模块组	Ports & Subsystems
信号属性模块组	Signal attributes
信号传输选择模块组	Signal Routing
接收器模块组	Sinks
信号源模块组	Sources
自定义函数模块组	User-Defined Functions
附加数学和离散模块组	Additional Math & Discrete

1.8.3 Simulink 文件打开与保存

单击模型库浏览器工具栏上的 New Model 按钮（图 1.8.1），可以新建空白的 Simulink 模型，如图 1.8.2 所示。还可以单击模型库浏览器工具栏上的 Open Model 按钮（图 1.8.1），打开一个现有的 Simulink

模型。Simulink 仿真模型文件有两种格式，分别是 .slx 和 .mdl 格式，slx 文件是二进制格式文件，mdl 文件是文本格式文件，slx 格式主要优势是文件小一些。用户在保存模型文件时，单击打开的 Simulink 仿真模型窗口上方的 Save 按钮即可（图 1.8.2），两种格式都可以选择。

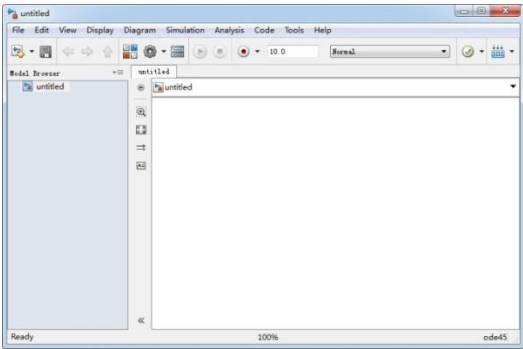


图 1.8.2 Simulink 建模仿真窗口

1.8.4 模块的基本操作

在 Simulink 仿真模型窗口中（如图 1.8.2），可以通过对模块进行各种操作来完成模型的搭建，具体如下表 1.8.2。

表 1.8.2 Simulink 中模块基本操作方法

操作内容	操作方法及相关说明
模块的添加	用鼠标指向模块库内所需模块（图 1.8.1），按下鼠标左键，把它拖至建模仿真窗口内（图 1.8.2），或者右击该模块并从上下文菜单中选择 Add block to model 命令添加到仿真窗口。
模块的选定	要选定单个模块，用鼠标指针指向待选模块，单击即可。选定多个模块的操作方法如下：按下 Shift 键，同时依次单击所需选定的模块；或按住鼠标任意一键，拉出矩形虚线框，将所有待选模块包在其中，于是矩形里所有的模块均被选中。此方法适合于选取位置相近的模块。
模块的移动	选中需移动的模块，按下鼠标左键，将模块拖到合适的地方即可。在移动的过程中按下 Shift 键，可以固定横向或者纵向移动。模块移动时，与之相连的连线也会随之移动。
模块的删除	选中待删除模块后，可以采用以下几种方法删除模块。按键盘上的 Delete 键，或者使用 Ctrl+X 快捷键，将选定的模块剪切到剪贴板上。
模块的内部复制	先按住 Ctrl 键，再单击模块，拖拽模块到合适的位置，松开鼠标按键，或选中模块，使用 Edit→Copy 及 Edit→Paste 命令。
模块的旋转	方法 1：选中模块，选择菜单命令 Diagram→Rotate&Flip→Clockwise，模块顺时针旋转 90°；选择菜单命令 Diagram→Rotate&Flip→Counterclockwise，模块逆时针旋转 90°；选择菜单命令 Diagram→Rotate&Flip→Flip Block，模块将旋转 180°。 方法 2：右键单击目标模块，在弹出的快捷菜单中进行与方法 1 同样的菜单项选择。
模块间的连线	模块间的连线是指从某一模块的输出端开始到另一个模块的输入端的有向线段。要另起一段绘制过程，将光标指向模块的输出端，待光标编程十字形后，按下鼠标左键，拖动鼠标，移动光标到另一个模块的输入端，然后释放鼠标按钮即可。此时，Simulink 就会自动生成一条带箭头的线段，把两个模块连接起来，箭头的方向表示信号流向。如果输入端和输出端不在同一水平线上，Simulink 会自动生成折线来连接两端。若连线没有连接上输入端就松开鼠标按钮，此时连接线就会变成一条红色的虚线来提醒用户连接有误。
模块名的设置	要修改模块名，单击模块名，光标会在文本间闪动，便可以进行修改。要设置模块名字体，单击 Diagram→Format→Font Style 命令，打开字体对话框后便可以设置。
标注信号线	双击要修改的信号线，会在连线旁边显示一个编辑框，在编辑框里输入标注即可。要设置标注字体，单击 Diagram→Format→Font Style 命令进行设置。

1.8.5 离散系统仿真分析举例

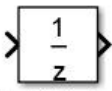
【例 1.8-1】 人口变化系统。设某一年的人口数目为 $x[n]$ ，其中 n 表示年份，它与上一年的人口数目 $x[n-1]$ 、人口繁殖速率 r 以及新增资源所能满足的个体数目 K 之间的动力学方程由如下的差分方程给出：

$$x[n] = r x[n-1] \left\{ 1 - \frac{x[n-1]}{K} \right\}$$

从此差分方程可以看出，此人口变化系统为一非线性离散系统。设人口初始值 $x[0]=100000$ ，人口繁殖速率 $r=1.008$ ，新增资源所能满足的个体数目 $K=1000000$ ，要求建立此人口变化系统的系统模型，并分析人口数目在 0 至 100 年的变化趋势。

(1) 建立系统模型。根据系统的数学描述选择合适的 Simulink 系统模块，建立此人口变化系统的 Simulink 模型，并对信号线添加标注，如图 1.8.3 所示。其中 Unit Delay 模块可以在 Simulink | Discrete 模块组子库中找到，其他模块可以在 Simulink | Commonly Used Blocks 模块组子库中找到。

这里说明一下相关的技巧。由于差分方程里包含 $x[n]$ 和 $x[n-1]$ ，所以可以先加入单位延迟模块

，若输入该模块信号为 $x[n]$ ，则输出信号为 $x[n-1]$ ，将 $x[n-1]$ 分为两路，一路先与 $-\frac{1}{K}$ 相乘

再与常数 1 加和，另一路不变，将两路相乘，最后经过比例系数 r 后与信号 $x[n]$ 相连，即完成模型的构建。

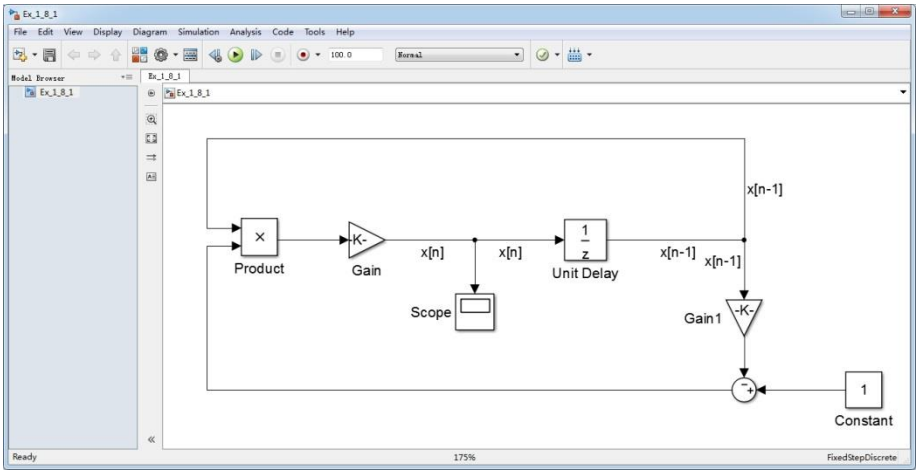
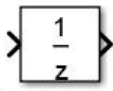


图 1.8.3 人口变化系统 Simulink 模型

(2) 模块参数设置。

➤ 单位延迟模块  是本系统的核心模块，首先设置该模块的参数，双击该模块打开其参数设置窗口进行参数设置，如图 1.8.4 所示。其中第一个参数 Initial condition（离散模块的初始值），是程序运行的起点，是必须设置的，这里即是人口初始值 $x[0]=100000$ 。第三个参数 Sample time（采样间隔）

如果设置为 -1，代表继承前面的采样间隔，由于本模块是程序的起点（其他模块可以继承此模块的采样间隔，此模块不可以继承其他模块的采样间隔），所以必须设置采样间隔。由于本例为离散系统，所以可以设置采样间隔为 1。

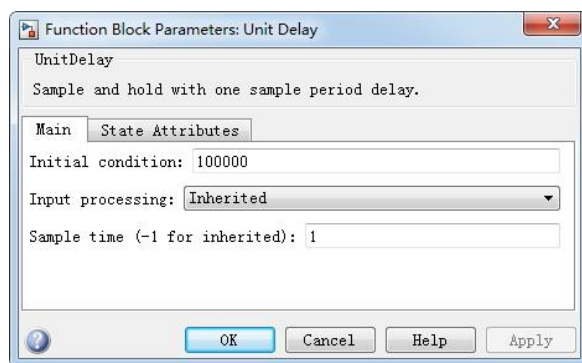



图 1.8.4 单位延迟模块的参数设置

- Gain 模块表示人口的繁殖速率，所以主参数 Gain（放大倍数）设置为 1.008，Sample time（采样间隔）设置为 -1 即可（继承前面的模块）。
- Gain1 模块表示新增资源所能满足的个体数目的倒数，故取主参数 Gain（放大倍数）设置为 0.000001，Sample time（采样间隔）设置为 -1。
- Product 模块表示信号相乘，Sample time（采样间隔）设置为 -1。


- Sum 模块  的 List of signs 参数设为 “-+|”（不同个数不同位置的加号、减号和竖线排列

组合会得到不同的效果，可以自行尝试）。Sample time（采样间隔）设置为 -1。

- Scope 示波器模块是 Simulink 仿真中非常重要的模块，可以显示时域仿真结果（离散或者连续均可），并且可以同时保存波形数据，是人机交互的重要手段。在本例中不需对示波器模块进行参数设置，其详细参数设置方法将在后面连续系统仿真分析举例中介绍。

（3）系统仿真参数设置

选择 Simulink 仿真平台窗口（图 1.8.3）菜单命令 Simulation→Model Configuration Parameter 或单击工

具栏中的  按钮可以进行仿真参数及算法的设置，弹出的仿真参数对话框如图 1.8.5 所示。图 1.8.5 中显示

的属性页面为求解器（Solver）属性页面（在图中左侧选项卡片中选取），主要功能是设置仿真的起始和终止时间，设置求解器类别和具体算法类型。

求解器主要参数设置：

- Simulation time 仿真时间

- Start time（仿真起始时间栏）：默认为 0，单位为秒。
- Stop time（仿真结束时间栏）：默认为 10，单位为秒。

本例中起始时间设为 0，结束时间设为 100。这里 1 秒对应 1 年，共 100 年，前提是单位延迟模块的采样间隔设为 1（如果设为 2，仿真时间 0 至 100 秒对应 50 年）。

- Solver option 求解器选项

- Type（求解器类别复选框）：求解器分为变步长（Variable-step）求解器和定步长（Fixed-step）求解器两大类，默认设置是 Variable-step。
- Solver（求解器具体算法类型复选框）：设定求解器的具体算法类型，对于变步长求解器包括：discrete、ode45、ode23、ode113、ode15s、ode23s、ode23t、ode23tb；对于定步长求解器包括：discrete、ode8、ode5、ode4、ode3、ode2、ode1、ode14x。其中 discrete 为离散求解器，其他都为连续求解器。

本例为离散系统的仿真分析，所以 Solver 栏选择离散求解器（discrete），对于离散系统的仿真，无论是采用定步长求解器还是采用变步长求解器，都可以进行精确的求解。这里选择定步长求解器，定步长与变步长的区别，将在后面连续系统仿真分析举例中介绍。

本例仿真参数设置后的窗口如图 1.8.6 所示。

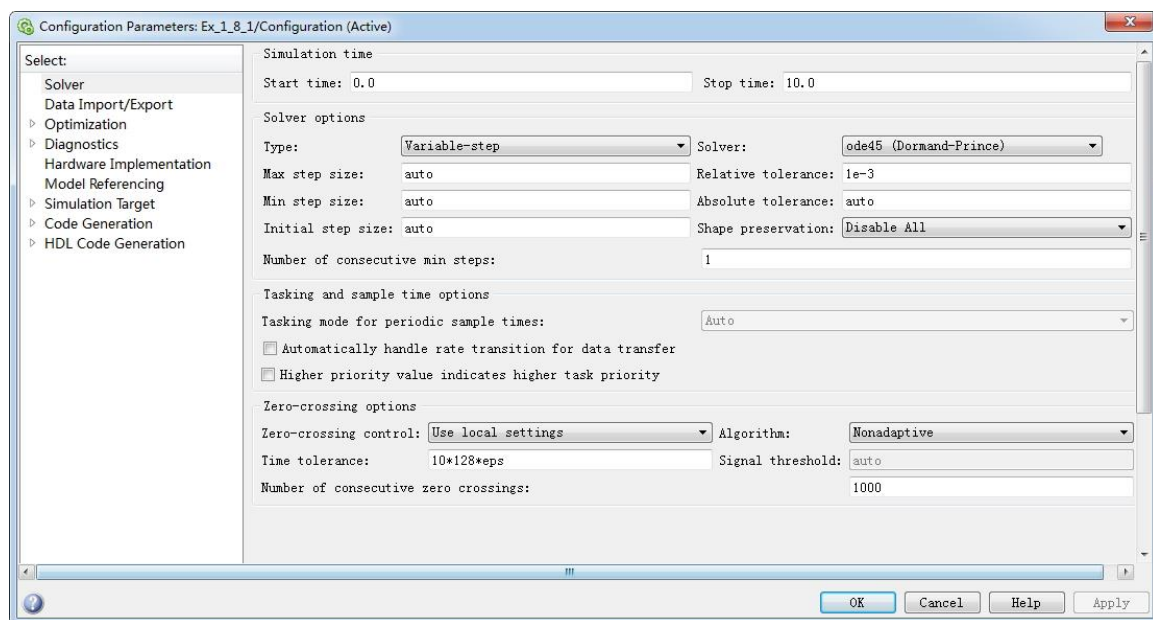


图 1.8.5 仿真参数设置对话框

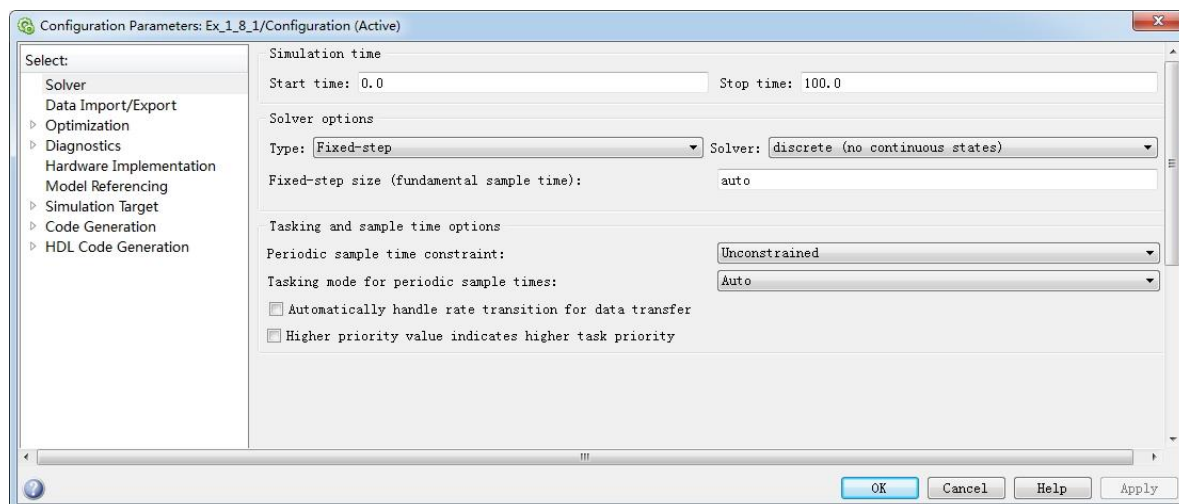




图 1.8.6 人口变化系统仿真参数设置

(4) 仿真运行

选择 Simulink 仿真平台窗口（图 1.8.3）菜单命令 **Simulation**→**Run** 或单击工具栏中的运行按钮 ，即

可运行仿真。仿真结束，双击系统模型中的 Scope 模块，按下自动尺寸按钮 ，显示系统仿真结果如图 1.8.7 所示。

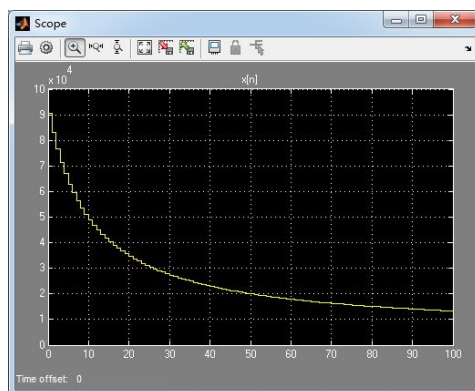


图 1.8.7 人口变化系统仿真输出结果

从图 1.8.7 中可以看到，人口数目在 0 到 100 年间逐渐减少，变化趋势是先急剧下降，然后趋于平缓。

1.8.6 连续系统仿真分析举例

【例 1.8-2】 本例将演示当输入为一个方波脉冲时的某二阶连续时间系统的输入输出过程，该二阶连续

时间系统的传递函数为 $H(s) = \frac{4}{s^2 + 4s + 4}$ ，需要分析输入与输出的时域信号曲线和频谱曲线，以及系统的

频率响应 $H(j\omega)$ 。在本例中将介绍 Scope（示波器模块）、Spectrum Analyzer（频谱分析仪模块）以及保存

To Workspace（数据至工作区模块）的详细用法。

（1）建立系统模型。

仿真系统的搭建比较简单，用到了 Step（阶跃响应模块）和 Transfer Fcn（连续传递函数模块），搭建后的系统模型框图如图 1.8.8 所示，图 1.8.8 表示的是输入输出时域曲线测量图。

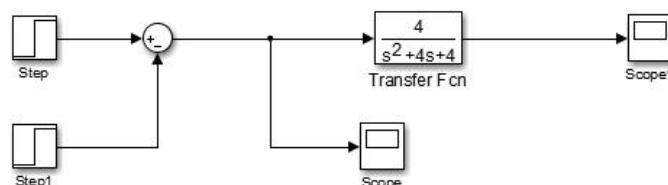


图 1.8.8 二阶连续时间系统模型之输入输出时域曲线测量

（2）模块参数设置。

Step（阶跃响应模块）可以在 Simulink | Sources 模块组子库中找到，两个阶跃响应模块做差合成一个方波脉冲信号，其中系数为正的阶跃函数的跳变位置为 10 秒，其属性设置如图 1.8.9 所示，其中 Step time（跳变点）设为 10，Initial value（跳变前值）设为 0（默认），Final value（跳变后值）设为 1（默认），Sample time（采样间隔）设为 0（默认），采样间隔值代表将此信号离散化的采样间隔，设为 0 代表此信号连续。另一个系数为负的阶跃函数跳变位置为 13 秒，所有参数设置方法与前一个阶跃函数模块相同。

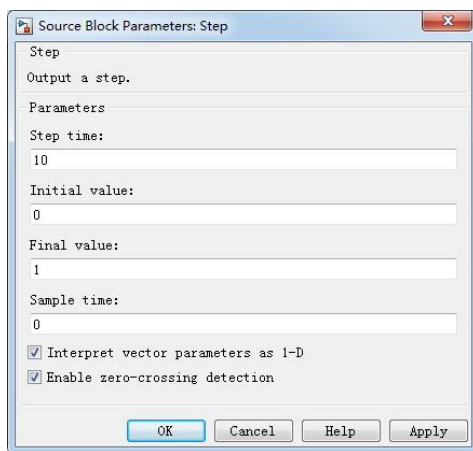


图 1.8.9 阶跃响应模块的参数设置

Transfer Fcn（连续传递函数模块）可以在 Simulink | Continuous 模块组子库中找到，可以代表一个二阶连续时间系统，从输入端输入信号在输出端可以得到输出信号。其属性设置如图 1.8.10 所示，Numerator coefficients（分子系数向量）代表系统传递函数分子 s 多项式（关于 s 降幂排列）的系数向量，这里我们设置为 [4]，代表关于 s 零次幂的系数为 4，其他项均为零，即分子只有一项，为常数 4；Denominator coefficients

（分母系数向量）代表系统传递函数分母 s 多项式（关于 s 降幂排列）的系数向量，这里我们设置为 [1 4 4]，

代表分母多项式为 $s^2 + 4s + 4$ 。其余参数使用默认。

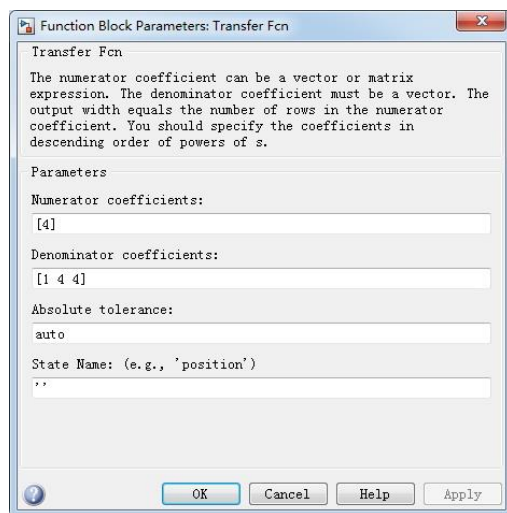



图 1.8.10 连续传递函数模块的参数设置

本例中同样应用到了 Scope（示波器模块），这里将对示波器模块的使用方法做详细的说明。

➤ 示波器参数

双击 Scope（示波器模块），即可弹出示波器的窗口界面，单击窗口中的示波器参数按钮 ，弹出如图 1.8.11 所示的示波器参数对话框，该对话框含有三个选项卡，分别是 General、History、Style 选项卡。

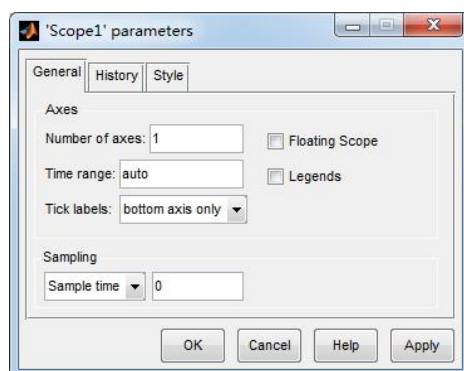


图 1.8.11 示波器参数对话框

● General 常规选项卡（图 1.8.11）

Number of Axes（坐标个数栏）：用于设定示波器的输入信号端口个数，默认值为 1，即表示本示波器只观察一路信号。

Time range（时间范围）：用于设定示波器时间轴的最大值，一般可选 auto（自动），这样 x 轴可以自动以系统的仿真时间范围作为示波器的时间显示范围。

Tick label（刻度标记）：用于选择是否加入坐标轴刻度标记。

Sampling（采样下拉框）：用于选择数据取样方式，包括 decimation（抽取）和 sample time（采样间隔）两种方式，系统默认使用“decimation”方式。选择“decimation”表示当右边文本框输入数字 N 时，从每 N 个输入数据中抽取一个用来显示，适用于离散数据输入的情况。当选择“decimation”时，右边文本框默认为 1，表示所有输入数据均显示。选择“sample time”表示按右边文本框所输入的数字为采样间隔采样取得数据进行显示，适用于连续数据输入的情况。当选择“sample time”时，右边文本框默认为 0，此时系统将参照模型中其他采样器件的采样间隔进行采样后画图，若模型中没有任何采样器件，则参照仿真参数设置中的 Max step size（最大步长）进行采样后画图。

本例中的两个示波器参数对话框的 General 选项卡设置如图 1.8.11 所示。

● History 历史选项卡（图 1.8.12）

Limit data point to last（仅显示最新的数据复选框）：用于数据点数设置，默认是选中的。选中后，其右侧的文本框被激活，默认值为 5000，表示示波器显示 5000 个数据，若超过 5000 个数据，也仅显示最后 5000 个数据。若不选该项，表示所有数据都显示。

Save data to workspace（保存数据至工作区复选框）：数据在显示的同时被保存到 MATLAB 工作区中，默认是不选中。若选中该项，将激活下方的 Variable name（变量名）、Format（格式）两

个参数设置项。Variable name 用于设置保存数据的变量名称，以便在 MATLAB 工作区中调用；Format 用于设置数据的保存格式，具体为 Structure with time（带时间变量的结构体数据）、Structure（结构体数据）、Array（数组数据）。需要说明的是通常不用 Scope 模块的该功能保存数据，保存数据可以使用 To Workspace（数据至工作区模块），用 To Workspace 模块保存数据更准确。

本例中的两个示波器参数对话框的 History 选项卡设置如图 1.8.12 所示。

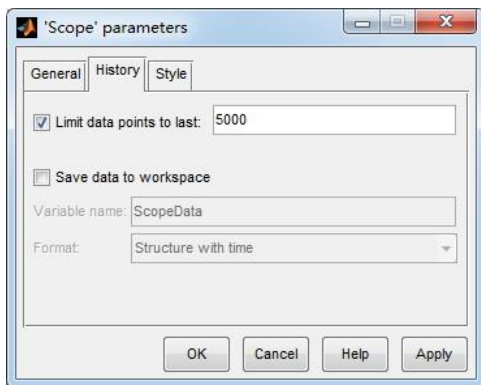


图 1.8.12 示波器参数 History 选项卡

- **Style 风格选项卡**

Figure color（图形颜色下拉框）：选择图形绘制的背景颜色。

Axes colors（坐标轴下拉框）：第一个下拉框选择绘图区域的填充颜色，第二个选择坐标轴和刻度的颜色。

Properties for line（曲线属性）：Properties for line 后面的下拉框包括了 1~6 六个选项，表示 Scope 显示多维信号时依次绘制图像时的颜色和规格，本例中信号维度为 1，所以只设定 Properties for line 后下拉为 1 的情况。当 Properties for line 后面的下拉框选定后，可以设定对应的 Line（线条属性）和 Marker（标注属性）。Line 属性代表所画图像的线型，Marker 属性代表所画图像数据点的标注图形。

本例中的两个示波器参数对话框的 Style 选项卡设置如图 1.8.13 所示。

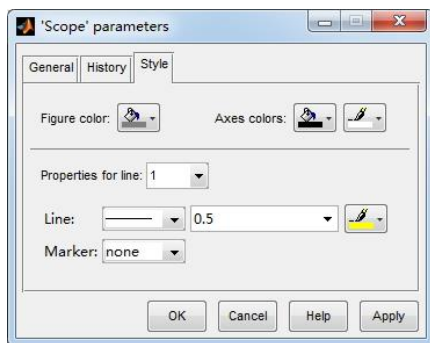






图 1.8.13 示波器参数 Style 选项卡

➤ **图形缩放**

- **区域放大按钮** ：首先在示波器的窗口界面工具栏中单击该按钮，然后在窗口中需要放大的区域上按住鼠标左键并拖拽一个矩形框，松开鼠标左键，该区域被放大显示。

- **x 轴放大按钮** ：首先在示波器的窗口界面工具栏中单击该按钮，然后在窗口中需要放大的区域上按住鼠标左键，并沿 x 轴方向拖拉即可。

- **y 轴放大按钮** ：首先在示波器的窗口界面工具栏中单击该按钮，然后在窗口中需要放大的区域上按住鼠标左键，并沿 y 轴方向拖拉即可。

- **自动尺寸按钮** ：能自动调整示波器的横轴和纵轴，调整时会参照仿真时间域以及对应的结果值域。

➤ **坐标轴范围**

在示波器窗口的图形区域内单击鼠标右键，在弹出的快捷菜单中选择 Axes parameters 坐标轴参数选项，出现一个名为 'Scope' properties: axis1 的坐标轴属性对话框，如图 1.8.14 所示。其中的 Y-min

和 Y-max 用来设置纵轴显示数值范围，Title 项用来给显示信号命名。

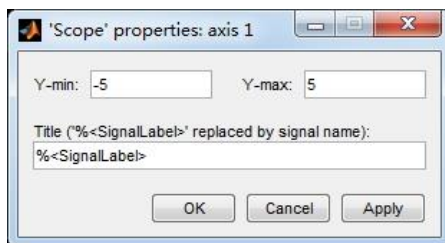


图 1.8.14 示波器坐标轴属性对话框

(3) 系统仿真参数设置

本例中需要将仿真时间设为 0 至 50 秒。下面对仿真求解器的设置进行说明。

MATLAB 对离散系统可以做到精确求解（应用离散求解器），但对于连续系统只能做到近似求解（应用连续求解器）。针对连续系统微分方程的不同数值求解方法，Simulink 有不同的连续求解器如下：

定步长连续求解器：ode8、ode5、ode4、ode3、ode2、ode1、ode14x

变步长连续求解器：ode45、ode23、ode113、ode15s、ode23s、ode23t、ode23tb

定步长求解器使用固定的仿真步长对连续系统进行求解，但使用定步长求解器不能对系统中的积分误差进行控制；而变步长求解器则能够根据用户指定的积分误差自动调整仿真步长，因此，变步长求解器能够对积分误差进行控制。积分误差分为如下两种。

绝对误差：积分误差绝对值。

相对误差：绝对误差除以状态值。

在仿真参数对话框中，用户可以对绝对误差和相对误差进行设置。一般来说，当状态值较大时，相对误差小于绝对误差，此时由相对误差控制求解器的运行；而当状态值接近零时，绝对误差小于相对误差，此时由绝对误差控制求解器的运行。在实际应用中，需要综合考虑系统仿真精度与仿真效率，然后选择合适的积分误差限制。

此外，在使用变步长求解器时，用户需要设置合适的初始仿真步长与最大仿真步长。不合适的初始仿真步长有可能使系统进入不稳定状态。最大仿真步长则可以为 Scope 等需要采样的模块提供采样间隔参考。

对于本例，选择 Variable-step（变步长）求解器，求解算法设置为 ode45，最大仿真步长设置为 0.01 秒，相对误差设置为 $1e-3$ ，具体仿真参数设置见图 1.8.15。

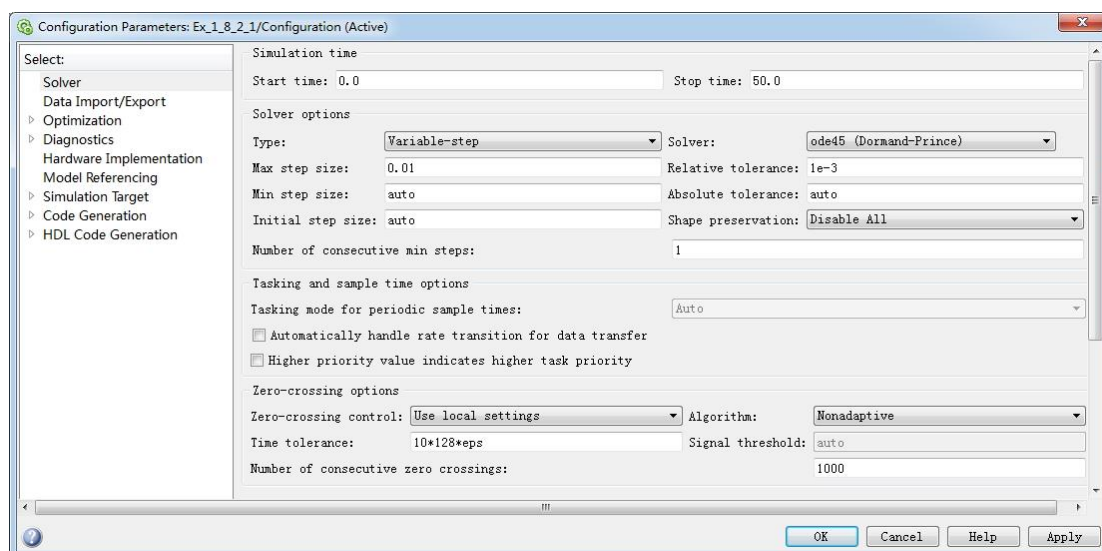


图 1.8.15 二阶连续时间系统模型仿真参数设置

(4) 仿真运行

运行后，输入和输出端的信号时域图像如图 1.8.16 所示，其中左侧为输入信号时域图像，右侧为输出信号时域图像。

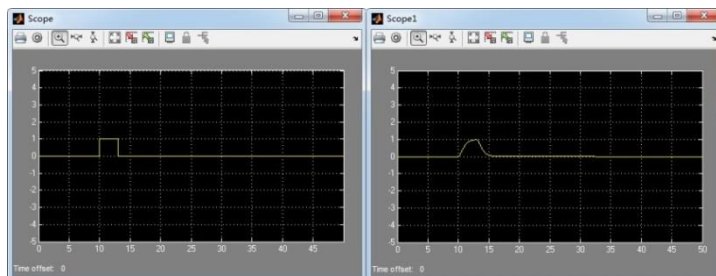


图 1.8.16 二阶连续时间系统模型之输入（左）输出（右）时域曲线测量结果

针对此二阶连续时间系统模型，还可以测量系统的频率响应曲线（如图 1.8.17）、测量输入输出信号的功率谱曲线（如图 1.8.23）、将输入输出信号（时域采样）作为变量储存入工作区（如图 1.8.25）。对于每一种情况的仿真参数设置均可参照图 1.8.15。

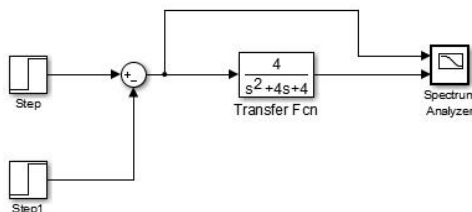


图 1.8.17 二阶连续时间系统模型之系统频率响应测量

图 1.8.17 表示的是二阶连续时间系统模型的系统频率响应测量，模型中用到了 Spectrum Analyzer 模块，可以在 Simulink Extras | Additional Sinks 中找到，需要说明的是有两个 Spectrum Analyzer 模块，这里用到的是双输入的 Spectrum Analyzer 模块。该模块可以直接输出系统的频率响应。关于双输入端口，位于上面的输入端连接系统输入信号，位于下面的输入端连接系统的输出信号。该模块共有 4 个参数：Length of buffer（计算缓冲区长度），代表计算过程中及最终的时域数据长度；Number of points for fft（fft 计算点数），代表系统进行的是多少点的 fft 计算（决定频率分辨率），取值时，fft 计算点数通常与缓冲区长度取相同的值；Plot after how many points（每过多少点更新频谱图像），此值可以取默认值 64，若取值过小则程序运行较慢；Sample time（采样间隔），为 Spectrum Analyzer 模块的采样间隔。

本例中，Length of buffer 和 Number of points for fft 取 5000，Plot after how many points 取 64，Sample time 取 0.01 秒，如图 1.8.18 所示。

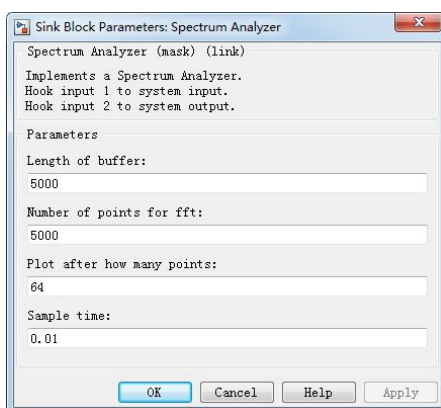


图 1.8.18 双输入 Spectrum Analyzer 模块参数设置

二阶连续时间系统模型的系统频率响应测量结果如图 1.8.19 所示

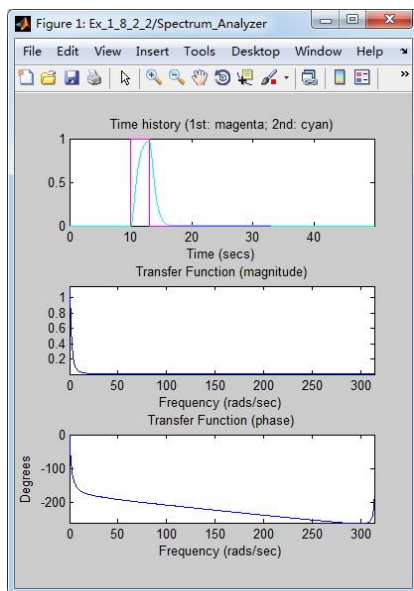


图 1.8.19 二阶连续时间系统模型的系统频率响应测量结果

图 1.8.19 中第一张子图表示缓冲区中的时域信号（包括输入和输出）；第二张子图表示系统频率响应的幅频特性曲线；第三张子图表示系统频率响应的相频特性曲线。观察幅频特性曲线，当横坐标为线性坐标时并不十分清晰，需要换为对数坐标显示。在图 1.8.19 中选择 **Edit**→**Figure Properties**，弹出图片属性设置框如图 1.8.20 所示。单击幅度谱图像，再拖动下方的滚动条，在 **X Scale** 后面的下拉列表中选择 **Log**。如图 1.8.21 所示。设置后的系统频率响应幅频特性曲线如图 1.8.22 所示。

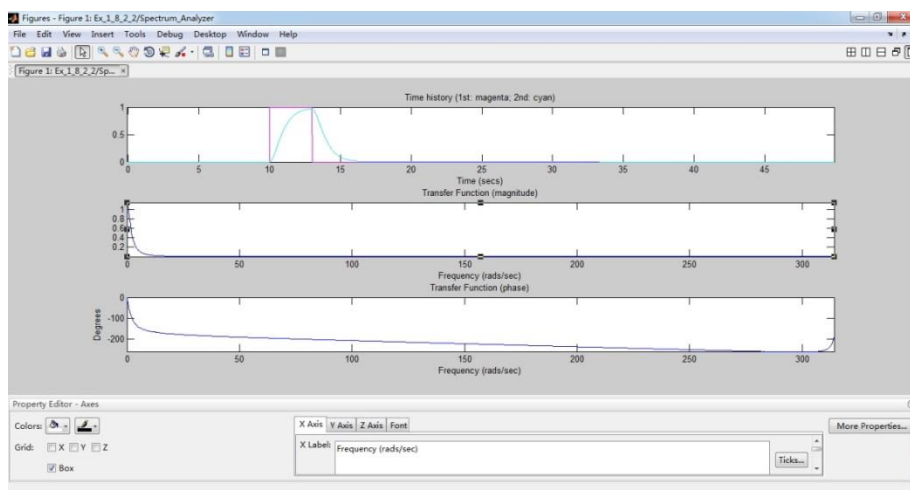


图 1.8.20 系统频率响应图片属性设置框

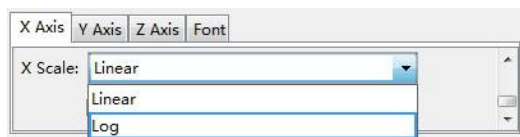


图 1.8.21 系统频率响应幅频特性横坐标显示模式设置

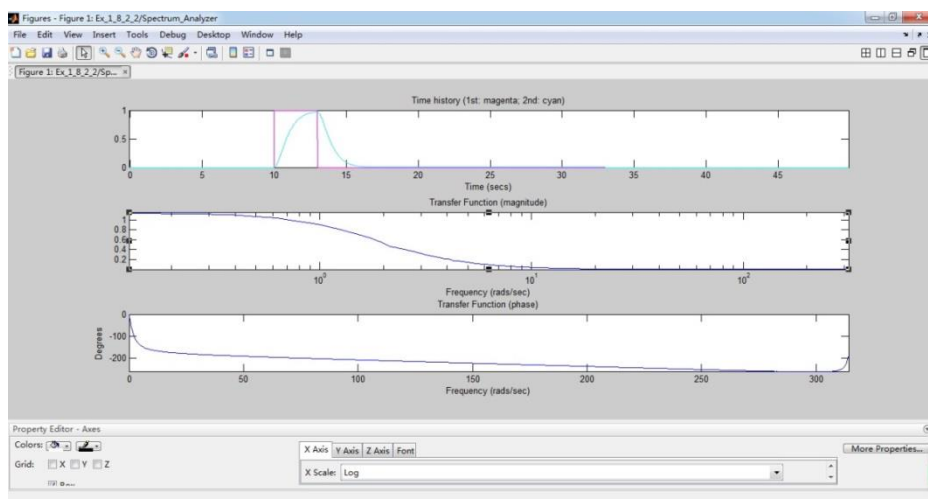


图 1.8.22 系统频率响应幅频特性横坐标取对数显示

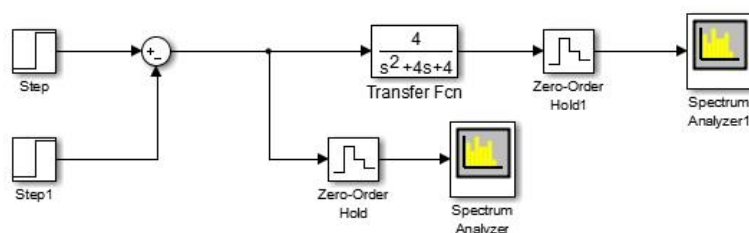


图 1.8.23 二阶连续时间系统模型之输入输出信号功率谱曲线测量

图 1.8.23 表示的是二阶连续时间系统模型的输入输出信号的功率谱曲线测量，模型中用到了 **Spectrum Analyzer** 模块，可以在 **DSP System Toolbox | Sinks** 中找到，这里用到的是单输入的 **Spectrum Analyzer** 模块。该模块可以输出信号的功率谱曲线。信号的功率谱的量纲是信号幅频特性的平方，且与信号的相频特性无关，所以此模块表示的是信号幅频特性的变化趋势。此模块要求输入端必须为离散信号，所以在输入前要对信号进行采样，这里用到了 **Zero-Order Hold**（零阶保持模块）进行采样，其参数只有一项，**Sample time**（采样间隔），本例中我们设置为 0.01 秒。

二阶连续时间系统模型的输入输出信号功率谱测量结果如图 1.8.24 所示，其中上面为输入信号功率谱图像，下面为输出信号功率谱图像。

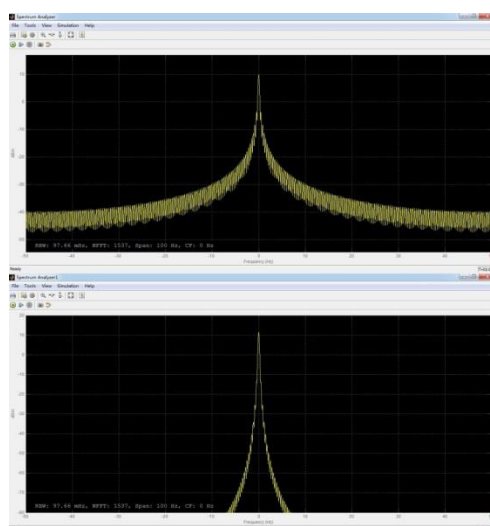


图 1.8.24 二阶连续时间系统模型的输入（上）输出（下）信号的功率谱测量结果

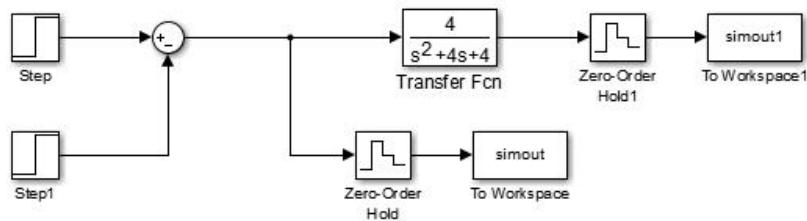


图 1.8.25 二阶连续时间系统模型之输入输出信号
(时域采样) 作为变量储存入工作区

图 1.8.25 表示的是二阶连续时间系统模型输入输出信号(时域采样)作为变量储存入工作区的情况, 这里用到了 To Workspace (数据至工作区模块), 可以将仿真中信号作为变量保存在工作区, 其输入必须为离散信号, 所以也需要 Zero-Order Hold (零阶保持模块), 采样间隔也设置为 0.01 秒。To Workspace 的参数设置窗口如图 1.8.26 所示, Variable name (变量名) 用于设置保存数据的变量名称, 默认变量名为 simout; Limit data points to last (保存最新的数据数量) 设置更新保存的数据点数, 默认值 inf 表示保存全部数据; Decimation (抽取) 设置每多少个仿真采样点保存一个点到变量中, 默认为 1 即全部保存; Sample time (采样间隔), 设置采样间隔, 由于 To Workspace 模块自带采样功能, 所以 Zero-Order Hold 可以省去, 这里我们保留 Zero-Order Hold 模块, 将 To Workspace 模块中的 Sample time 属性设为 -1 (继承前面的采样间隔); Save format (保存格式) 设置数据保存的格式, 具体为 Structure with time (带时间变量的结构体数据)、Structure (结构体数据)、Array (数组数据)、Timeseries (对应时间序列的数组数据), 这里我们选择 Array; Log fixed-point data as a fi object (将固定点数据类型作为一个 fi 对象保存) 默认不勾选, 代表将数据作为 double 型保存。To Workspace 的参数设置结果参考图 1.8.26。

运行模型, 输入信号的时域采样作为变量 simout 储存入工作区; 输出信号的时域采样作为变量 simout1 储存入工作区。

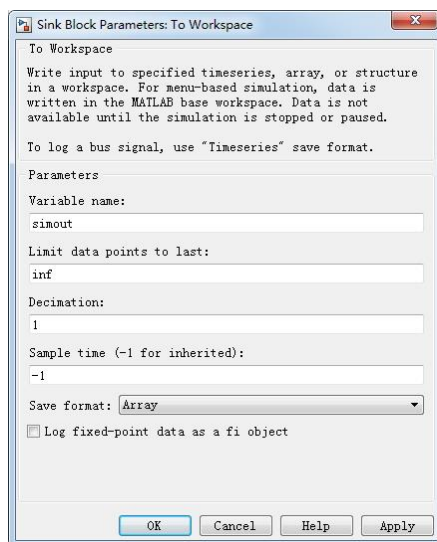


图 1.8.26 To Worspace 模块的参数设置

在同一工作路径下建立 M 文件, 编辑代码如下, 运行。

```
f = (-2500:2500)*100/5001;
y1 = fft(simout)*0.01;
y2 = fft(simout1)*0.01;
Z1 = abs(y1);
Z2 = abs(y2);
figure(1)
plot(f,fftshift(Z1));
xlim([-5,5]);
figure(2)
plot(f,fftshift(Z2));
xlim([-5,5]);
```

figure(3)

```
plot(f,fftshift(Z2./Z1));
```

```
xlim([-5,5]);
```

可以得到 figure 1-3 三张图, figure 1 代表输入信号的幅频特性曲线; figure 2 代表输出信号的幅频特性曲线; figure 3 代表系统的频率响应幅频特性曲线, 如图 1.8.27 所示, 从左至右分别为 figure 1、figure 2 和 figure 3。

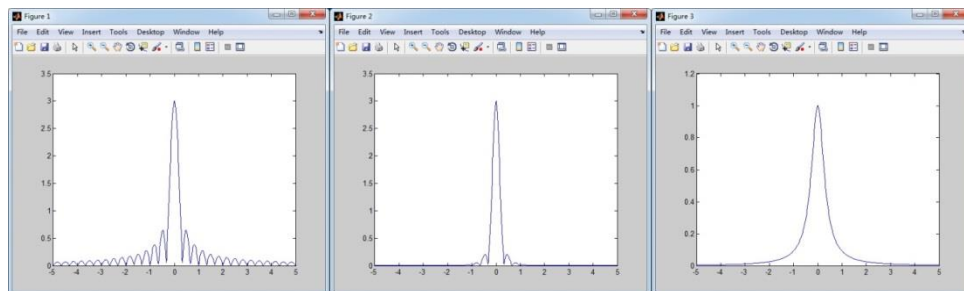


图 1.8.27 利用仿真中保存的数据进行的频谱分析曲线

1.8.7 子系统的建立

最后, 说明一下子系统的建立方法。如果被研究的系统比较复杂, 那么直接用基本模块构成的模型就比较庞大, 模型中信息的主要流向就不容易辨认。此时, 若把整个模型按时限功能或对应物理器件的存在划分成块, 将有利于理顺整个系统的逻辑关系。

1. 如果模型本身不包含组成子系统的模块, 在模型中新建立一个子系统可以按下列步骤进行。
 - 1) 在 Simulink Library Browser 的 Ports & Subsystems 模块组中选取合适的 Subsystem 模块并施至模型窗口中。
 - 2) 双击 Subsystem 模块, 打开 Subsystem 窗口。
 - 3) 把要组合进子系统的模块拖拉到 Subsystem 窗口中, 然后在该窗口中加入 Inport 模块, 表示从子系统外部到内部的输入; 加入 Outport 模块, 表示从子系统内部到外部的输出。把这些模块按设计顺序连接起来, 子系统就建立成功了。
2. 组合已有的模块建立子系统

如果用户创建完了一些模块, 又想把这些模块变成子系统, 其操作步骤如下: 用方框同时选中待组合的模块。或者按住 Shift 键逐个选中, 在菜单栏中单击 Edit→Creat Subsystem 命令, 或者右击, 从弹出的菜单中选择 Creat Subsystem 命令, 子系统就建成了。此操作非常简单方便, 请读者自行尝试, 这里不再举例说明。