Maple 中文用户手册



版本：试用版 v0.1

编制：Maplesoft China

2020 年 9 月 20 日

前言

本教程是 Maplesoft 为广大 Maple 软件用户编制的入门级使用教程，供大家免费使用和参考。

本教程使用的建模软件是 Maple 2020 版。

如需申请试用版软件，请登录 沅航科技官网申请（https://www.simtech-sh.com/Maple

同时也可以登陆Maple官网审请

由于水平有限，教程中错误之处在所难免，敬请读者指正！

联系邮箱：info@simtech-sh.com

目录

第一章 Maple 使用环境 1
1.1 软件简介 1
1.2 安装的系统配置要求 1
1.3 启动 Maple 软件界面 1
1.4 新建 Maple 文件 2
1.4.1 文件模式 3
1.4.2 工作表模式 4
1.4.3 保存文件 5
1.5 快速参考卡 6
1.6 函数和函数包的使用 9
1.7 变量名 11
1.8 数字格式 11
1.9 常规运算 13
1.9.1 数的表示 14
1.9.2 基本的运算符号 14
1.9.3 数字运算规则 14
1.9.4 比较算符 14
1.9.5 求算式的值 15
1.9.6 多项式计算 17
第二章 数据结构 18
2.1 序列 (Sequence) 18
2.2 列表 (List) 19
2.3 集合 (Set) 20
2.4 数组 (Array) 21
2.5 矩阵 (Matrix) 和向量 (Vector) 23
2.6 表 (Table) 25
第三章 常规的数学计算 27
3.1 代数 27
3.2 求方程与不等式 30
3.3 矩阵计算与线性代数 32
3.3.1 矩阵计算 33
3.3.2 矩阵 LU 分解 35
3.3.3 矩阵 QR 分解 36
3.3.4 大型矩阵的数值计算 36
3.3.5 线性代数函数列表 37
3.4 微积分 43
3.4.1 极限 43
3.4.2 微分 44
3.4.3 方向导数 45
3.4.4 级数 46

3.4.6 多变量和向量微积分.....47
3.4.7 Student 微积分函数包和助教.....48
3.4.8 微积分问题求解示例.....48
3.5 优化.....53
3.5.1 线性规划.....55
3.5.2 非线性规划.....58
3.5.3 最小二乘优化问题.....59
3.5.4 全局优化.....61
3.6 概率论与数理统计.....68
3.6.1 概率分布和随机变量.....69
3.6.2 统计计算.....70
3.6.3 统计图形.....70
3.4.5 积分.....72
3.6.4 Student Statistics 学生学习函数包.....73
3.7 内置工程及科学常数数据库.....74
3.8 含单位和公差的科学计算.....74
3.8.1 输入单位.....74
3.8.2 国际规范.....75
3.8.3 单位计算.....76
3.8.4 绘图中使用单位.....77
3.8.5 在方程、积分、优化等计算时使用单位.....77
3.9 公差计算.....79
第四章 图形和动画.....81
4.1 使用关联菜单绘图.....81
4.2 二维图形和三维图形命令.....82
3.2.1 单变量表达式绘图.....82
3.2.2 函数绘图.....82
3.2.3 两个变量函数的二维图形.....83
3.2.4 两个变量函数的三维图.....83
4.3 plots 函数包.....83
3.3.1 多个图形和动画的合并显示.....83
3.2.2 微分方程解的绘图.....85
3.2.3 对数作图.....85
3.2.3 极坐标图.....86
3.3.4 隐函数的极坐标图形.....86
3.3.5 波特图.....87
3.3.6 数据表的绘图.....88
3.3.7 点云图.....88
3.2.8 柱状图.....88
3.2.9 双 y 轴图形.....89
4.4 提取图形的数据并保存为 Excel 文件.....90
4.5 动画.....91

第五章 数据处理.....92
5.1 导入数据.....92
5.1.1 使用图形化用户界面导入数据文件.....92
5.1.2 使用命令导入 Excel 文件中的数据.....93
5.2 拟合数据.....94
5.3.1 Fit 命令.....95
5.3.2 拟合噪声信号的模型.....98
5.3.3 最小二乘拟合.....100
5.3.4 三维数据的多项式拟合.....102
5.4 提取图形的绘图数据并输出 EXCEL 文件.....104
第六章 微分方程求解.....106
6.1 定义微分方程.....106
6.2 使用 dslove 命令求解析解和数值解.....107
6.2.1 定义微分方程.....108
6.2.2 求解析解.....108
6.2.3 求数值解.....108
6.2.4 求级数解.....109
6.3 微分方程的判定.....110
6.4 求解一阶常微分方程 ODE.....112
6.5 求解常微分方程组 ODEs.....115
6.6 求解偏微分方程 PDE.....117
6.6.1 一个简单的例子.....117
6.6.2 求解热传导方程.....117
6.6.3 求解波动方程.....119
第七章 创建计算书.....128
7.1 样式.....128
7.2 创建文件.....130
7.2.1 输入数学和文字.....130
2.2.2 插入图片.....131
7.2.3 插入视频.....131
7.2.4 章节管理.....132
7.2.5 插入表格.....133
7.2.6 输出为 PDF 文件.....133
7.3 Maple Workbook 管理项目中的多个文件.....134
7.3.1 保存为 Maple Workbook 工作薄文件格式.....135
7.3.2 导航面板.....135
7.3.3 从 Workbook 读取文件.....136
7.3.4 变量管理器.....137
7.4 文件内容加密.....137
7.5 分享 Maple 文件.....138
第八章 编程.....139
8.1 编写代码的几种方式和工具.....139

8.2 编写 Procedures 过程程序 140
8.2.1 编写一个简单的过程程序 Procedure 140
8.2.2 一些简单的示例和说明 143
8.3 条件和循环语句 146
8.3.1 程序流控制 146
8.3.2 返回结果: return 和 error 语句 147
8.4 迭代: for 循环和 while 循环 148
8.4.1 for/from 循环 149
8.4.2 for/in 循环 150
8.4.3 while 循环 150
8.4.4 嵌套循环 151
8.5 创建一个模块 Module 152
8.6 创建一个函数包 Package 154
8.7 代码调试和分析 155
8.7.1 编程习惯：注释 155
8.7.2 调试 155
8.7.3 代码分析 159
第九章 图形用户界面 GUI 应用开发 162
9.1 常用的 GUI 组件 162
9.2 GUI 组件操作 165
9.2.1 插入 GUI 组件 165
9.2.2 编辑组件的属性 166
9.2.3 删除 GUI 组件 168
9.2.4 在文件中使用 GUI 组件 168
9.3 示例 – 创建包含 GUI 组件的文件 170

第一章 Maple 使用环境

1.1 软件简介

Maple 软件是 1980 年由加拿大滑铁卢大学两位教授 Keith Geddes 和 Gaston Gonnet 领导的科研小组开发，并以加拿大的国树枫叶（Maple）命名的数学软件。1988 年，Maplesoft 公司成立，开始面向全球销售 Maple 软件（以下简称 Maple）。

Maple 是一个数学软件，有“数学家的软件”之称。Maple 提供世界领先的符号和数值计算工具，可以处理您的所有数学需求，从执行简单计算到高级计算、可视化、数据分析和算法开发。目前，Maple 已成为世界上最为通用的数学和工程计算软件之一，被广泛地应用于教育、研究和工程等领域。此外基于数学，Maple 提供了数十个附加工具箱和仿真模块 MapleSim，覆盖科学计算、在线学习、数字孪生、车辆仿真、航空航天系统设计、基于模型的系统工程等不同的应用领域和行业应用。

1.2 安装的系统配置要求

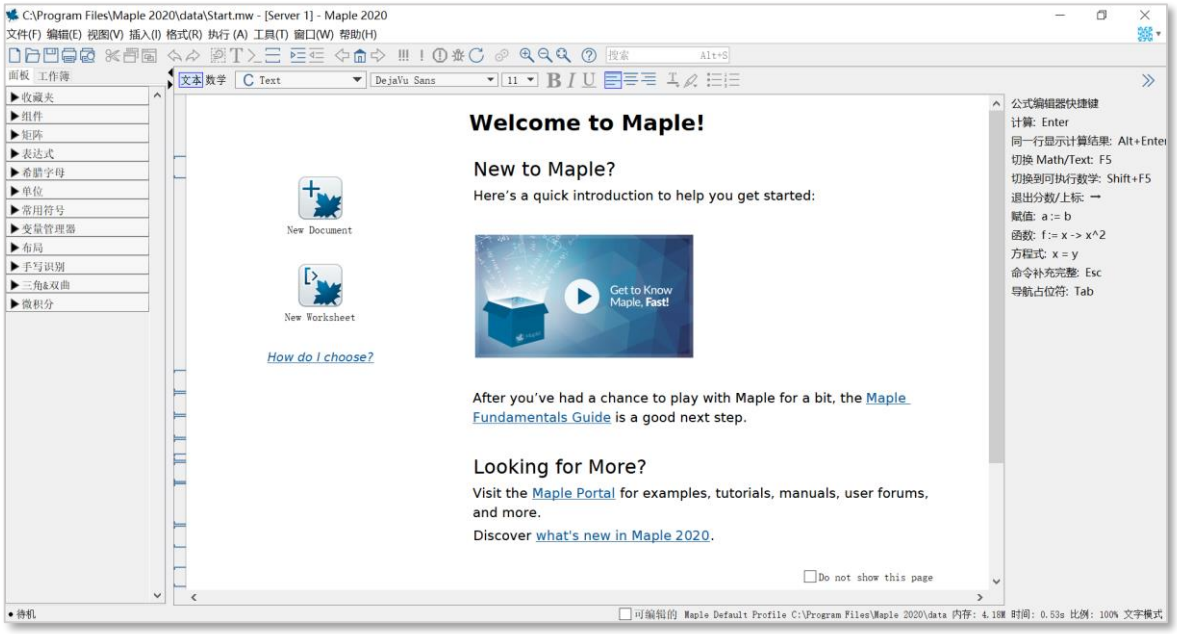
Maple 是一个跨平台软件，适用于各种主流操作系统，包括 32 位或 64 位 Windows、64 位 Linux、以及 Mac 系统。Maple 2020 版本计算机推荐配置：内存大于 4G，不小于 10G 的硬盘空间。

1.3 启动 Maple 软件界面

启动 Maple 软件：

Windows	从开始菜单，选择所有所有程序 → Maple 2020 → Maple 2020. 或者： 双击桌面上的 Maple 2020 快捷键图标。
Mac	1. Finder → Applications → Maple 2020. 2. 双击 Maple 2020.
Linux	输入完整的路径，例如 /usr/local/maple/bin/xmaple

	或者： 1. 将 Maple 目录（例如，/usr/local/maple/bin）添加到命令搜索路径。 2. 输入 xmaple.



图：Widows 下 Maple 软件启动界面

Maple 软件打开后，会显示一个启动页面，其中包含一些指向任务和主题的快捷键。

1.4 新建 Maple 文件

可通过以下方式新建 Maple 文件：

- 在启动页面，选择 New Document 或者 New Worksheet，点击后会出现一个空白的文件。

或者，

- 从文件菜单，选择新建，然后选择工作表模式或者文件模式。会出现一个空白文件。

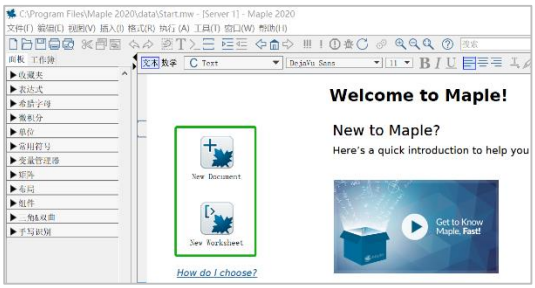
你也可以设置启动 Maple 软件后直接显示一个空白文件，替代默认的启动页面。你也可以创建一个客户化的启动页面，替换现有的默认启动页面。

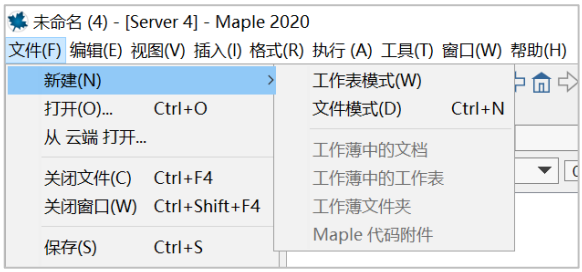
在任意时间可以打开启动页面，直接点击工具栏上的主页按钮。

注意：后面有文件模式和工作表模式差异的详细说明，但实际上两者并无功能的差异，你可以任选一种模式。推荐先使用文件模式。

1.4.1 文件模式

Maple 提供两种工作模式：文件模式和工作表模式，可从启动页面或者<文件>菜单创建一个新的文件或者工作表。





文件模式适用于输入数学表达式，快速求解问题；也可用于创建美观的技术文件。Maple 有三种工作方式，其一是点击工具栏上的按钮 T，进入纯文字模式，在光标后面只能输入纯文字；其二是点击按钮 ≥_，进入命令行模式；前三是点击按钮 ☰，进入文件块模式。点击上述三个快捷键可实现方式的转换。

这三种方式中都可以输入数学表达式和文字，单点击工具栏上的“文字”和“数学”按钮 文本 数学 （或者按 F5 键）可进行文字格式和数学格式的切换。命令行模式中的数学表达式，无论是文字格式还是数学格式都可以用 Maple 运算；纯文字模式和文件块模式中的数学表达式，只有在数学格式下输入的可用 Maple 运算。另外，数学格式中编写的公式是 2-D 的，而文字格式中编写的数学表达式是 1-D 的。

在文件方式中输入文字和 1-D 数学表达式的方法是选择文字格式，光标显示为垂直线，随后可进行输入问题。通常这种模式用于输入描述性文字。如下图所示。

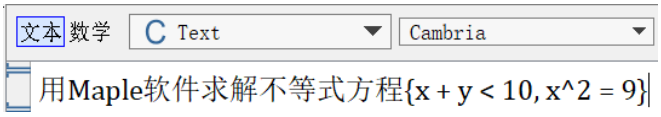


图: 1-D 文字格式

在文件方式中输入二维数学表达式的方法是选择数学格式，光标显示为斜体，周围有虚线框。数学格式下输入的内容，都被视为可执行命令，按回车键后会另起一行显示结果。如键入(1-*x*^2)/(1+*x*^2)可得到就会出现如下图所示结果。

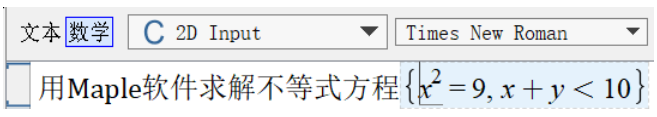
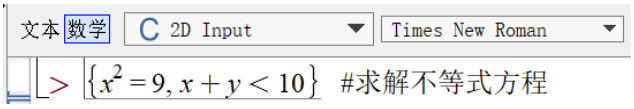


图: 2-D 数学格式

使用命令行方式输入 2-D 数学表示式的方法是点击工具栏上的按钮，在光标所在位置上插入一行命令行提示符[>，默认格式是 2-D 数学格式。如键入(1-*x*^2)/(1+*x*^2)就会出现如下图所示结果。另外，我们还可以使用软件界面左边工具栏中快捷键，方便地输入 2-D 数学表达式。



图：命令行方式中输入 2-D 数学符号

使用命令行方式输入一维数学表示式的方法是点击工具栏上的按钮，在光标所在位置上插入一行命令行提示符[>，点击工具栏上的“文字”按钮 文本 数学 （或者按 F5 键），在命令行提示符后可输入 1-D 代码。如键入(1-*x*^2)/(1+*x*^2)，就会出现如下图所示效果。

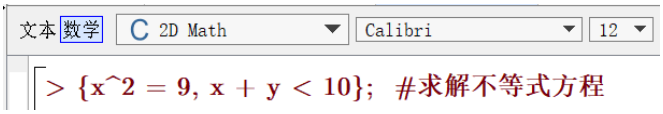


图: 命令行方式中输入 1-D 数学符号

Maple 默认输入格式是数学格式。我们可以通过菜单工具 → 选项 → 显示，设置默认模式为 Maple 符号（文字格式），然后点击全局应用按钮。

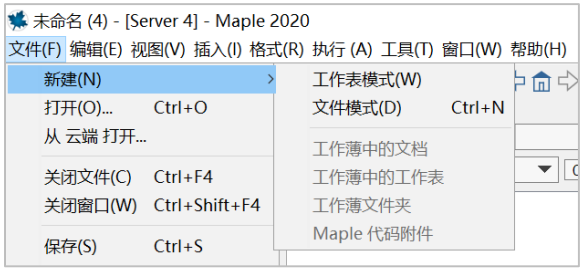


图: 设置默认输入格式为文字格式

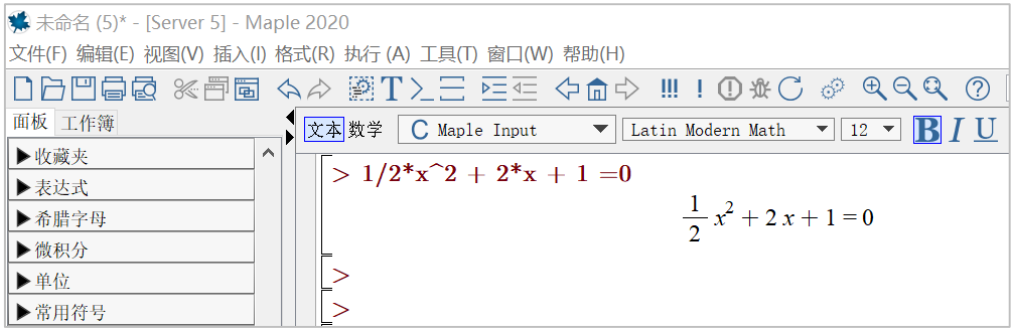
2-D 数学表达式转换为 1-D 数学表达式的方法是用从关联菜单中选择 2-D 数学 → 转换为 → 1-D 数学输入格式。

1.4.2 工作表模式

工作表模式适用于编程，其有关文字和数学表达式的操作与文件模式界面相同。



图：使用菜单新建一个工作表界面

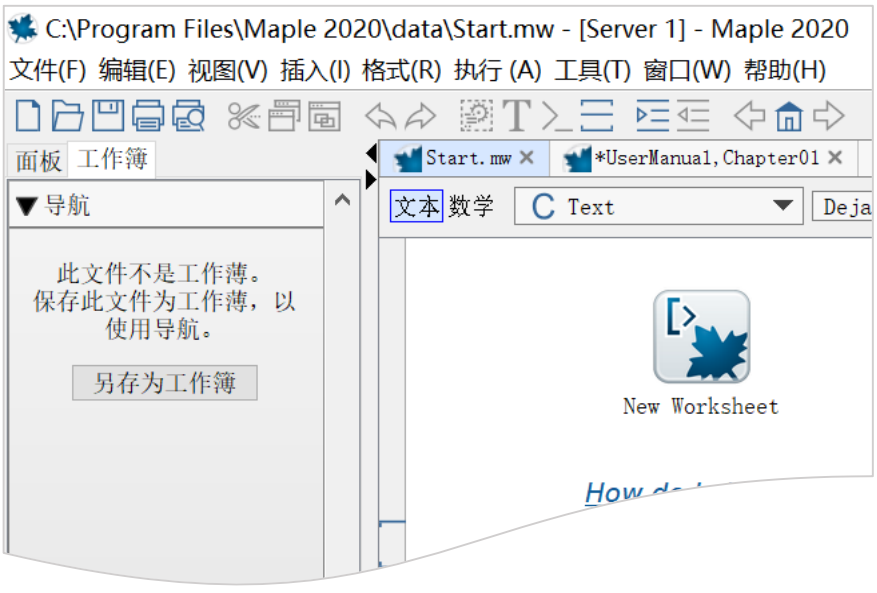


图：工作表界面

1.4.3 保存文件

保存前面创建的例子，选择菜单文件 → 保存，或者点击工具栏上保存。文件模式和工作表模式的默认保存格式后缀名都是.mw。

也可以保存为 Maple workbook 工作薄格式，单个 workbook 文件可以包含多个不同格式的 Maple 文件或者第三方软件文件。保存方式：文件 → 另存为...，然后再弹出的另存为窗口选择文件类型 Maple workbook (.maple)。或者，从左侧“工作薄”面板，点击“另存为工作薄”。



1.5 快速参考卡

<table>
<tr><th colspan="2">Maple 软件操作快速参考卡
Windows 版本</th></tr>
<tr><td colspan="2">文件模型 vs. 工作表模式</td></tr>
<tr><td colspan="2">Maple 提供了两种主要的问题输入和内容创建模式：文件模式(Document mode)和工作表模式(Worksheet mode)，两种模式都有各自的优势，您可以轻松地从一种模式切换到另一种模式，以实现最大的灵活性。
备注：从文件模式中可以使用工作表模式下的所有功能，反之亦然，这两种模式的区别仅仅是初始的状态不同。因此，您使用 Maple 时，可以选择任一种模式。</td></tr>
<tr><td>文件模式
• 适用于快速求解问题，格式比较自由，可以创建美观的技术文件
• 不显示输入命令提示符 (>)
• 输入的数学公式显示为 2-D
• 可以用鼠标点击输入或输出，利用关联菜单求解数学问题</td><td>工作表模式
• 类似于经典版 Maple 软件界面
• 输入提示符 (>) 后输入问题
• 输入的数学公式显示为 2-D 或 1-D
• 可以用鼠标点击输入或输出，利用关联菜单求解数学问题</td></tr>
<tr><td>文件模式允许用户创建美观的内容，类似于 Word 环境。例如，不使用命令求 x 的值：
$\frac{(x-2)}{\alpha}=1 \xrightarrow{\text{solutions for x}} \alpha+2$</td><td>在工作表模式下，使用命令完成相同的操作。
2-D（数学）输入：
> $solve\left(\frac{x-2}{\alpha}=1, x\right)$
$\alpha+2$
或者在 1-D (Maple) 输入：
> solve((x-2)/alpha=1,x);
$\alpha+2$</td></tr>
<tr><td>切换输入的数学或文字格式：按[F5]键，或者工具栏按钮 文本 数学</td><td>切换 2-D 或者 1-D 数学输入格式：按[F5]键，或者工具栏按钮 文本 数学，2-D 显示为黑色字体，1-D 显示为红色字体</td></tr>
<tr><td>求值并在同一行显示结果：[Ctrl][=]</td><td>求值，结果显示在新的一行上：[Enter]</td></tr>
<tr><td>求值，结果显示在新的一行上：[Enter]</td><td>不执行当前语句，新增加一行：[Shift][Enter]</td></tr>
<tr><td>在可执行数学和惰性数学之间切换：[Shift][F5]</td><td>隐藏命令，仅显示结果：选中要隐藏的命令，点击菜单编辑 → 文档块 → 创建文档块</td></tr>
<tr><td>切换到工作表模式（插入命令输入提示符）：点击工具栏上图标 ≥_</td><td></td></tr>
<tr><td>显示隐藏的命令：点击菜单编辑 → 文档块 → 显示命令</td><td></td></tr>
<tr><td colspan="2">文件模式和工作表模式下都适用的常规操作</td></tr>
<tr><td>使用方程标签引用前面的结果：</td><td>按[Ctrl][L]键，然后在对话框中输入方程编号</td></tr>
</table>

重新计算一行内的计算：	点击工具栏按钮 !
重新执行文件中的所有计算：	点击工具栏按钮 !!!
符号补全，例如：ϵ (epsilon)	输入几个首字母，按[Esc]键（英文操作系统下也可使用[Ctrl][Space]），例如：eps[Esc]
命令补全，例如：Lambert W 函数	输入手写几个字母，按[Esc]键（或者[Ctrl][Space]），例如：Lamb[Esc]
对数学表达式完成关联菜单操作：	使用关联面板
插入输入提示符：	点击工具栏图标 ≻_
插入文字：	点击工具栏图标 T
用鼠标拖动方式拷贝表达式到新的位置：	选中要拷贝的文字或数学，按住[Ctrl]键，拖放到新的位置。

2-D 数学编辑操作，快捷键

移动光标	[←][→][↑][↓]
将光标移动到表达式的不同部位，例如离开指数	[→]
从一个占位符移到下一个占位符	[Tab]
添加、隐藏、重新排列面板：	视图 → 面板，或者鼠标右键点击面板
分数 $\frac{x}{y}$，上标 x^n，下标 x_n	x/y, x^n, x_ _n
微分加撇符号，例如 $y''+y'=0$，等价于 $\frac{d^2y}{dx^2}+\frac{dy}{dx}=0$	y'' + y' = 0
平方根 $\sqrt{x}$， n 次根 $\sqrt[n]{x}$	输 sqrt 的前几个字母，然后按[Esc]键, nthroot 按[Esc]键
号上标，例如 $\vec{x}$	x [Ctrl][Shift]["] 然后从左侧的箭头面板中插入 →
输入文字字符(^, /, 等.), 在字符前使用 \(反斜杠)	例如 foo\^bar 产生 foo^bar
特殊字符和符号：输入头几个字符，然后按[Esc]键	π, e, i — pi, e, i ∞ — infin α, λ — alpha, lambda ≥, ≤, ≠, ± — geq, leq, ne, pm
更多信息见帮助系统：?2DMathShortcutKeys	

绘图和动画

绘制已有表达式的图形	从关联菜单选择图形生成器
对新表达式绘图	工具 → 助手 → 图形生成器
添加新的表达式到已有的图形	选中该表达式，然用鼠标左键拖放到已有图形中
在图形中添加注释	点击图形然后从工具栏选择画图 

多变量函数的动画和参数化图形	从关联菜单选择图形生成器，选择动画
更多信息见帮助系统：?PlottingGuide	

数学操作

常规操作 (简化、因式分解、展开，…)	使用关联菜单
解方程	输入方程，然后从关联菜单选择求解
求数值解 (浮点数)	输入方程，然后从关联菜单选择求解→求数值解
常微分方程求解	输入微分方程，然后从关联菜单选择交互求解微分方程
积分、微分	输入表达式，然后从关联菜单选择积分或微分
求表达式在一点上的值	输入表达式，然后从关联菜单选择求一点上的值
创建一个矩阵或向量	左侧面板：矩阵 → 选择 → 插入
逆矩阵、转置矩阵、求解矩阵	输入矩阵，然后从关联菜单选择标准运算→逆、转置、...
求值为浮点数	从关联菜单选择近似值
各种操作和任务	使用任务模板：工具 → 任务 → 浏览

重要的 Maple 语法

:= 赋值	a := 2; b := 3 + x; c := a + b; c 的结果是 $5+x$
= 数学方程	solve(2*x + a = 1, x); 结果是 $x=\frac{1-a}{2}$
= 布尔等式	if a = 0 then …
不显示输出	命令结尾使用冒号，例如 1000! :
[ ] 列表 (自动排序)	z:=[c, b, a]; z[1]; 结果是 c
{ } 集合 (不排序，去除重复项)	{a, b, a, c}; 结果是 {a, b, c}
显示主题的帮助	?topic
更多信息见帮助系统：?syntax	

表达式 vs. 函数

操作	表达式 x^2+y^2	函数 (算子) $g(x,y)=x^2+y^2$
定义	f := x^2 + y^2;	g := (x,y) -> x^2+y^2;
在 x=1, y=2 上求值	val(f, [x=1,y=2]); 结果是 5	g(1,2); 结果是 5
绘制 x 从 0 到 1，y 从 0 到 1 范围上的三维图形	plot3d(f,x=0..1,y=0..1);	plot3d(g(x,y),x=0..1,y=0..1);

相互转换	f2 := unapply(f,x,y);
	f2(1,2);
	结果是 5
		g2 := g(x,1);
		g2 + z;
		结果是 x^2+1+z

单位和公差计算

为数值或表达式添加单位	将光标放在物理量的右边，使用面板或者关联菜单：

	单位→附加单位
添加任意的单位	点击单位面板中〚unit〛，然后输入期望的单位
化简表达式中的单位	点击关联菜单：单位→化简
将当前单位转换为其他单位制	点击关联菜单：单位→转换
启用自动单位化简	with(Units)[Standard];
启用公差计算	with(Tolerances);
2-D 数学中的公差量	9 pm [Esc] 1.1 产生 9 ± 1.1
1-D 数学中的公差量	9 &+- 1.1; 产生 9 ± 1.1
输入和输出	
交互式数据输入助手	工具 → 助手 → 导入数据
输入声音或图片文件	工具 → 助手 → 导入数据
在工作表中保留数据文件等	文件→ 另存为 → 选择 Workbook 类型，然后使用插入 → 附加附件
代码生成(C, C#®, Fortran, Java, JavaScript®, MATLAB®, Perl, Python®, R, Visual Basic®等)	从关联菜单选择交语言转换
输出为其他文件格式：HTML, PDF, LaTeX, 或者 Word-RTF	文件 → 输出 → 选择 HTML, PDF, LaTeX, 或者 Rich Text Format
交互式工具	
预置的任务模板	工具 → 任务 → 浏览
图形生成器	从关联菜单选择图形生成器，或者菜单工具 → 助手 → 图形生成器
单位转换工具	工具 → 助手 → 单位转换
数值格式	关联面板中选择数字格式
使用 MapleCloud 分享 Maple 文件	MapleCloud 工具栏菜单
Maple Portal	进入主页：Start → Getting Started → Maple Portal
手册	帮助系统中
交互式助教工具，包括微积分、预科微积分、线性代数、数值分析等	工具 → 助教

1.6 函数和函数包的使用

Maple 将其函数或命令分为两类：主函数库(main library)和函数包(Package)。主函数库包含最常用的函数或命令，例如 sin，taylor，int，exp，dsolve，solve，fsolve，rhs 和 eval 等等，其余的函数或命令，则按照领域打包成不同的函数包，如代数、统计、微分几何、优化函数等包。为了节约内存，当启动 Maple 时，主函数库会自动加载，函数包并不会自动加载，使用时需使用命令调用。调用函数包的格式有短格式和长格式两种，具体调用如下：

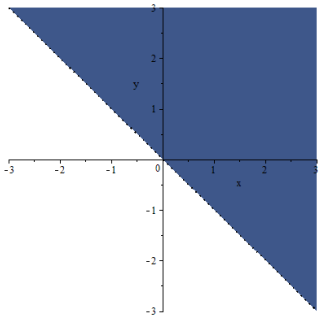
短格式：首先用 with 语句加载函数包 PackageName，然后直接使用命令进行运算；

长格式：PackageName[CommandName]或者 PackageName:-CommandName。

例1.1 绘制 0<*x*+*y* 的图形。

解 我们先使用短格式，在 Maple 中做如下运算：

```
[>restart:     #清除内存
[>with(plots);      #加载 plots 函数包
[>inequal(0<x+y,x=-3..3,y=-3..3);      #画出由不等式定义的区域
```



我们也可以使用长格式，在 Maple 中做如下运算：

```
[>plots[inequal](0<x+y,x=-3..3,y=-3..3);
```

或者

```
[>plots:-inequal(0<x+y,x=-3..3,y=-3..3);
```

运行以上命令可得到同样图形。

重要常数和基本初等函数

Maple 具有几百个常用的数学常数及函数，包括基本初等函数和一些特殊函数，下面介绍一些常用的常数和函数的表达方式。

例1.2 从下面几个例子来观察输入方式与输出格式之间的联系。

```
[>exp(2.);      #求数值近似解
```

7.389056099

```
[>exp(2);
```

e^2

```
[>evalf(exp(2));      #求数值近似解
```

7.389056099

从以上实例可看到：如希望使用 Maple 系统内的重要常数和基本初等函数进行数值计算，则必须以小数形式输入数值或使用 evalf 命令。

1.7 变量名

变量名是由字母开头的字母和数字的字符串，**切记不能以数字开头。**变量名中不能带有空格、标点符号、算号等。例如：yy、x3s 是变量名，3s、s*r、d e 就不是变量名（其中 表示空格）。

1.8 数字格式

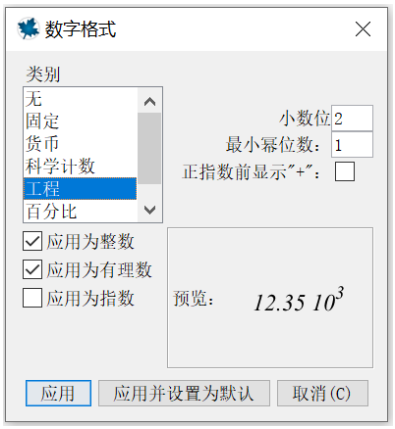
可以通过菜单或关联面板控制 Maple 中的数字格式，Maple 支持以下标准数字格式：

- 固定
- 货币
- 科学
- 工程学
- 百分比

你也可以创建自定义格式，例如自定义财务数据、日期、分数、时间等。

数字格式设置的两个途径：

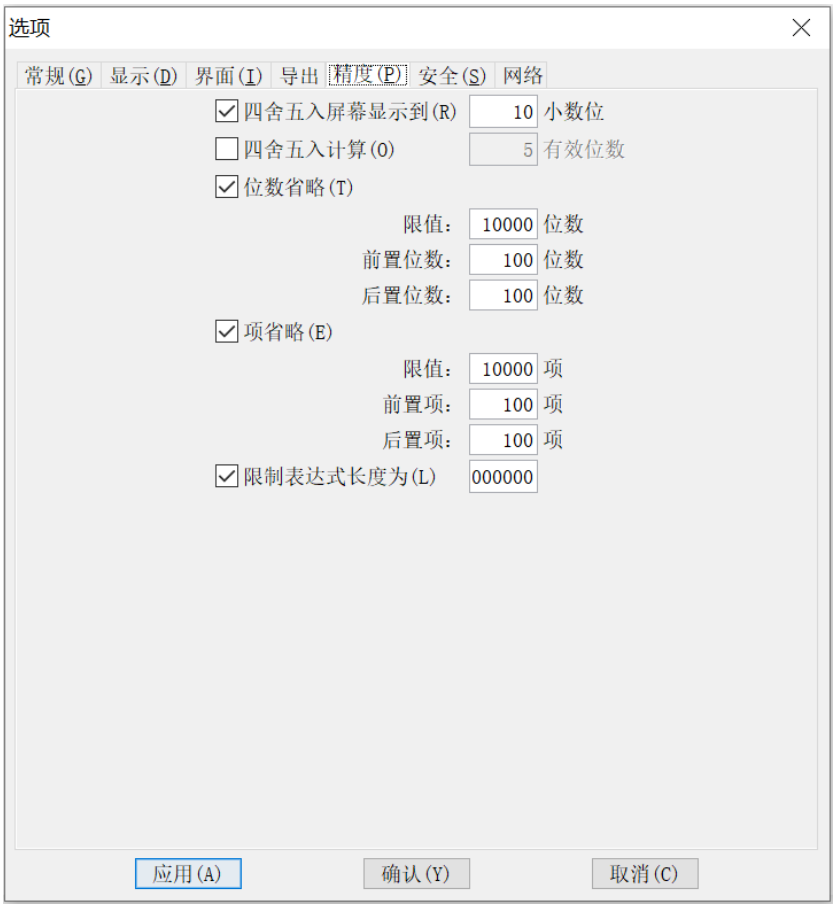
1. 菜单：格式 > 数字格式



2. 关联菜单，点击数字后，右侧面板会自动显示数字格式菜单。



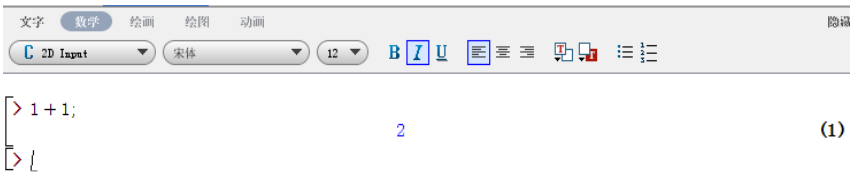
此外，Maple 中显示的数字格式和位数，系统默认设置有关。可以从菜单工具 > 选项 > 精度，修改其中的系统默认设置。如下图所示。



1.9 常规运算

进入 Maple 窗口后，可通过“帮助”菜单了解 Maple 的操作和使用方法。另外，也可以通过 Maple 软件公司的官方网站 www.maplesoft.com 查询 Maple 操作方法和应用案例，输入数学表达式后，如果要进行数学运算，将光标放在要运算的数学表达式上，按回车键，点击工具栏上的执行按钮 ！，或点击鼠标右键，使用弹出的右键菜单求解数学问题。

Maple 将每次输入记录在案，输出将另起一行居中显示，后面自动附加一个标签（如下图所示）。



图：加法运算

本书以后在叙述到上述运算时，写成：

[>1+1;

2

注意：[>是 Maple 自动显示的命令行提示符，无需我们手工输入。如要显示输出结果，运算表达式后加“;”；如不要显示输出结果，运算表达式后加“:”。

如果要删除单个文字，可以使用“Del”键；如果需要删除整行，可以使用 Ctrl+Del 组合键，Maple 的这一“超级删除”功能键可用于复杂对象的整行删除操作。

当输入的数学表达式较长时，为了在窗口中看到整个数学表达式，可将光标停在任一运算符后面按 **Shift+Enter** 组合键，便可使数学表达式换行。

如要同时计算几个数学表达式，方法有二。其一是每个数学表达式后面加“;”，然后按回车键或点击工具栏上的执行按钮 ！。例如：

[>1+1;2-6;

2

-4

其二是分别输入数学表达式点击工具栏上的按钮 !!!，Maple 将执行文件中的所有运算。例如：

[>1+1;

2

[>2-6;

-4

Maple 的许多操作和菜单与 Word 是一样的。在以后操作中使用较多的打开、关闭、复制、存盘等与 Word 操作完全一致，这里就不一一介绍。

1.9.1 数的表示

Maple 中，数的表示有精确数与浮点数两种形式。除几个常用的数学常数（如 Pi 表示π，exp(1)表示$e=2.718289\cdots$等）外，与通常的表示基本相同。

1.9.2 基本的运算符号

在 Maple 中，加、减、乘、除、幂分别用+、-、*、/、^表示。

1.9.3 数字运算规则

Maple 的数字运算规则与通常数的运算规则相同，其由高到低的优先级依次为：乘方，乘（除），加（减），连续几个同级运算（除乘方外）从左到右进行，用圆括号可改变运算的次序。例如：

表 1-1　四则运算输入法

数学表达式	键盘输入方法
$\frac{2^2}{2}+1$	2^2/2+1
$\frac{x}{2}+1$	x/2+1
$\frac{x}{y+z}$	x/(y+z)

1.9.4 比较算符

Maple 共有六个比较算符，其表达方式如表 1-2 所示。

表 1-2　比较算符输入法

比较算符	数学意义	键盘输入方法
=	等号	=
>	大于	>
<	小于	<
≤	小于等于	<=
≥	大于等于	>=
≠	不等于	<>

注意：在 Maple 中，等于号=也表示方程等式。在 Maple 中，x=x 表示一个代数方程，而不是一个布尔表达式。如要判别 x 与 x 是否相等，需使用 evalb(x=x)或 is(x=x)。Maple 自动求值为布尔表达式发生在以下情况下：

- 包含逻辑算子；
- if 语句中的条件判断；
- while 语句中的循环语句。

1.9.5 求算式的值

算式是指由数字、变量、+、-、*、/、^及圆括号构成的合理算式。

例1.3 计算 $2*(3+4)-2^{2+1}$。

解 在 Maple 中做如下运算：

```
[>2*(3+4)-2^(2+1);
```

$$6$$

解得：$2*(3+4)-2^{2+1}=6$。

例1.4 计算 2^{100}。

解 在 Maple 中做如下运算：

```
[>2^100;
```

$$1267650600228229401496703205376$$

以上方法与下面方法比较。

```
[>2^100;
```

$$1.267650600\times 10^{30}$$

解得： 2^{100}=1267650600228229401496703205376≈1.26765×10^{30} 。

例1.5 计算 $\sqrt[3]{-2}$ 的数值。

解 在 Maple 中做如下运算：

```
[>(-2.)^(1/3);
```

0.6299605250+1.091123636*I

其中 I 表示单位虚数。$\sqrt[3]{-2}$ 的数值理应是实数，而 Maple 计算结果却是复数，这是因为 Maple 是将任一数的指数运算化成 *e* 的指数进行运算。就上述例题而言，是将 $\sqrt[3]{-2}$ 化成 $e^{\log(-2)^{1/3}}$ 进行计算，从而导致复数的结果。为了得到正确结果，我们应这样计算：

```
[>-2.0^(1/3);
```

-1.259921050

解得：$\sqrt[3]{-2}$ ≈-1.259921050。

4.5.2 evalf(表达式)计算法

用 evalf(表达式)方法计算的结果是有效数为十位的近似值。

用 evalf[n](表达式)方法计算的结果是有效数为 *n* 位的十进制数。

例1.6 计算 $\frac{1}{35}+\frac{3}{136}$ 。

解 在 Maple 中做如下运算：

```
[>1/35+3/136;
```

$$\frac{241}{4760}$$

```
[>evalf(1/35+3/136);      #求数值近似解
```

0.05063025210

解得：$\frac{1}{35}+\frac{3}{136}=\frac{241}{4760}$ ≈0.05063025210。

例1.7 计算 $\frac{1}{300}$ 的数值，要求有效数为 20 位。

解 在 Maple 中做如下运算：

```
[>evalf[20](1/300);      #求数值近似解
```

0.00333333333333333333

或者

```
[>Digits:=20;        #控制 Maple 使用软件浮点数进行计算时使用的数字数
                                        Digits:=20
[>evalf(1/300);
                                  0.00333333333333333333
```

解得：$\frac{1}{300} \approx 0.00333333333333333333$ 。

1.9.6 多项式计算

Maple 中可进行多项式计算，例如因式分解、展开和化简可分别用命令 factor，expand 和 simplify 实现，它们的功能由以下实例说明。

例1.8 因式分解 a^2-b^2，展开 $(a-b)(a+b)$ 及，展开并化简。

解 在 Maple 中做如下运算：

```
[>restart:   #清除内存命令
[>factor(a^2-b^2);       #因式分解
                                      (a+b)(a-b)
[>expand((a+b)*(a-b));       #多项式展开
                                       a^2-b^2
[>expand((a+b)^2/(a^2+2*a*b+b^2));       #多项式展开
                                  a^2/(a^2+2ab+b^2)+2ab/(a^2+2ab+b^2)+b^2/(a^2+2ab+b^2)
[>simplify(expand((a+b)^2/(a^2+2*a*b+b^2)));       #符号化简
                                          1
```

第二章 数据结构

Maple 有多种数据结构，用于编程或交互式使用函数，这些数据结构包括：矩阵、数组、列表、集合、序列、表、模块等。其中许多数据结构提供相似的功能，但是一些数据结构可以更好地适用于某些类型的操作，因此熟悉数据结构和它们之间的区别非常重要，这样可以针对不同的问题选择恰当的结构和操作。

Maple 中的数据结构可分为两个基本类型：可变的(mutable)和不可变(immutable)。可变数据结构可以修改，即它们存储的值可以更改。不可变的数据结构在创建后无法更改。此外对这些数据结构的操作也会产生不同的结果。这种行为上的差异可能会对使用这些数据结构的代码性能产生重大影响。

2.1 序列 (Sequence)

Maple 中最基础的数据结构是序列。序列就是一组用逗号隔开的表达式列。例如当 Maple 中的函数有多个参数项，这些参数项用逗号分隔开，这实际上就是由一组参数项构成的序列。

```
restart:
a := 1, 2, 3;
```

一个序列可以加入其他序列，产生一个新的序列：

```
b:=a,4,5;
c:=NULL;
a,c,b;
```

$$1, 2, 3, 1, 2, 3, 4, 5$$

其他应用场景：函数的参数项通常是序列。

```
solve(3*x+5 = 0, x);       #求解方程
```

$$-\frac{5}{3}$$

```
d := 3*x+5 = 0, x:         #生成一个序列
solve(d);        #使用求解函数 solve，参数项使用序列 d
```

$$-\frac{5}{3}$$

提取序列中的元素项，可以使用索引：

```
a[2];
```

```
b[2 .. 4];
```

```
b[2 .. 4][3];
```

编程方式生成一个序列，使用 seq 函数：

```
seq(x^i,i=1..11,2);
```

$$x, x^3, x^5, x^7, x^9, x^{11}$$

2.2 列表 (List)

列表实际上是用一对方括号[ ]封装的序列，例如：[2, 3, 4]。与序列不同的是，列表中的嵌入列表展开方式不同。列表存储有序的表达式序列。列表中元素项的顺序是固定的。与集合不同，列表将保留重复的元素项。

```
a := [1, 2, 3];
```

```
whattype(a);          #判断 a 的类型
```

list

```
op(a);          #返回列表中的表达式序列
```

```
whattype(%);
```

exprseq

```
b := [a, 4, 5];
```

$$b := [[1, 2, 3], 4, 5]$$

```
c := [];
```

```
[a, c, b];
```

$$[[1, 2, 3], [\,], [[1, 2, 3], 4, 5]]$$

```
d := [op(a), 4, 5];
```

可以使用索引获取列表中的元素项。

```
b[1];
```

$$[1, 2, 3]$$

可以使用 member 函数测试一个表达式是否是列表的成员。

member(4, b);

true

has(b, 2);

true

使用 numelems 函数得到列表中元素项的数量。

numelems(d);

10

2.3 集合 (Set)

集合是用花括号{ }封装元素项的数据结构，与列表不同的是，集合中不可以有相同的元素，创建集合时，Maple 执行自动简化，将删除所有重复元素，并对其余元素重新排序。集合中的元素不分先后顺序。

a := {1, 2, 3};

$$a := \{1, 2, 3\}$$

b := {4, 5, a};

$$b := \{4, 5, \{1, 2, 3\}\}$$

c := {};

$$c := \varnothing$$

{a, b, c};

$$\{\varnothing, \{1, 2, 3\}, \{4, 5, \{1, 2, 3\}\}\}$$

跟列表一样，用户可以对集合使用 op 命令（提取操作数），或者可以使用 convert 命令在列表和集合之间转换。

op(a);

1, 2, 3

[op(a)];　　#加上[ ]变为列表类型

[1, 2, 3]

convert(a, 'list');

[1, 2, 3]

过滤列表或集合中的元素，使其满足一定的条件，可以使用 select。

```
d := [2, 3, 4];
select(issqr, d)
issqr(2), issqr(3), issqr(4)
select(x -> x^2 = 4, [-3, -2, 2, 1, 2, a])
```

想要获得不满足条件的元素项，使用 remove。或者使用 selectremove 得到两者。

```
remove(issqr, d);
```

[2, 3]

```
selectremove(issqr, d);
```

[4], [2, 3]

想对列表或集合中的所有元素应用一个函数，使用 map。

```
map(sin, d);
map(issqr, d);
map(f, [1, 2, 3]);
```

$$[f(1), f(2), f(3)]$$

2.4 数组 (Array)

在前面的段落中，我们并没有讨论如何改变表达式序列 sequence、列表或者集合。这是因为这些数据结构在 Maple 中属于不可变的(immutable)。用户可以创建一个新的这类数据结构，但不能改变已有的。数组（Array）属于可变的数据结构。

数组，以及后续讨论的矩阵、向量，在 Maple 中表示为相同的结构 - rtable，存储为可变密度数组。当需要改变元素值时，使用数组或其他基于 rtable 的数据结构。

```
a := Array([3, 2, 1])     #创建一个数组 a
```

$$\begin{bmatrix} 3 & 2 & 1 \end{bmatrix}$$

```
b := Array([a, 4, 5])      #创建一个数组 b
```

$$\begin{bmatrix} \begin{bmatrix} 3 & 2 & 1 \end{bmatrix} & 4 & 5 \end{bmatrix}$$

如果我们改变 a，那么 b 会随之改变。我们也可以通过 b 改变 a。

a[1] := 5;

b;

$$\left[\begin{array}{cccc}\left[\begin{array}{ccc}5 & 2 & 1\end{array}\right] & 4 & 5\end{array}\right]$$

b[1][3] := 16;

a;

$$\left[\begin{array}{ccc}5 & 2 & 16\end{array}\right]$$

可以定义数组索引的范围。

a := Array(-1 .. 1, [3, 2, 1]);

$$Array(-1..1, \{(1) = 3\})$$

a[-1]; a[0]; a[1];

$$3$$

$$2$$

$$1$$

b := Array(1 .. 5, 1 .. 3);

$$\left[\begin{array}{ccc}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{array}\right]$$

b[3, 2] := 5:

b;

$$\left[\begin{array}{ccc}0 & 0 & 0\\0 & 0 & 0\\0 & 5 & 0\\0 & 0 & 0\\0 & 0 & 0\end{array}\right]$$

用户可以使用 numelems 命令得到数组中元素的数目，使用 upperbound 和 lowerbound 命令得到索引的上下界。

numelems(a), numelems(b);

$$3, 15$$

```
lowerbound(a), upperbound(a);
```

$$-1, 1$$

考虑这样的情况，给出含有n个数的有序数据集（列表或数组）。假设你希望计算累计和，也就是$s_k = \sum_{i=1}^{k} x_i$，对所有的$k \le n$。$k \ge 2$时$s_k = s_{k-1} + x_k$。如果你使用列表（列表），你需要在每一步创建一个新的列表，代码如下。（这里我们使用了for循环语句；我们将在下一段落进行稍微详细的研究。）

```
n := 10;
x := [seq(1 .. n)];
s := [x[1]];
```

$$10$$

```
for k from 2 to n do
    s := [op(s), s[k - 1] + x[k]]; end do;
s;
```

$$[1, 3, 6, 10, 15, 21, 28, 36, 45, 55]$$

使用数组（Array），仅需要更新其中的值，而不是重新创建新的数据对象。由于这个原因，在这类应用场景中，使用数组相比列表计算性能更高，推荐使用数组。

```
s := Array(1 .. 10, [x[1]]);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
for k from 2 to n do
    s[k] := s[k - 1] + x[k];
end do;
s;
```

$$[1, 3, 6, 10, 15, 21, 28, 36, 45, 55]$$

2.5 矩阵 (Matrix) 和向量 (Vector)

Maple 提供多种方式创建矩阵和向量，主要有：

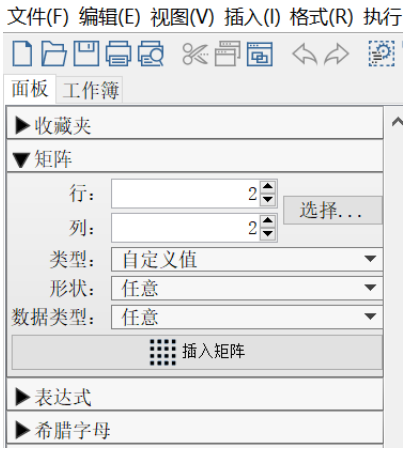
(1) 矩阵面板

(2) Matrix() 和 Vector() 命令

(3) <<>>

(4) 不同数据类型之间的转换

方式（1）：矩阵面板



例：使用矩阵面板创建一个矩阵，手工填入数值，或选择选择矩阵类型为随机值。

$$mymatrix := \begin{bmatrix} -32 & -4 & 8 & 99 \\ -74 & 27 & 69 & 29 \end{bmatrix}$$

方式（2）：Matrix() 和 Vector() 命令

当使用 Matrix 或 Vector 命令时，有多个输入格式。例如，输入一列嵌套列表。矩阵或向量的维数由给出元素项的数目确定。

```
M := Matrix([[1, 2, 3], [4, 3, 4], [3, 5, 3]]);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 3 & 4 \\ 3 & 5 & 3 \end{bmatrix}$$

```
V := Vector([2, 3, 4]);
```

$$\begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

使用索引函数创建矩阵或向量：

```
M := Matrix(10, 10, (i, j) -> i);
V := Vector(10, i -> i^2);
```

方式（3）：可以使用尖括号 <>

使用逗号分隔元素项，用垂直线 | 分割列。

```
V := <<2, 4, 5>>;
```

$$\begin{bmatrix} 2 \\ 4 \\ 5 \end{bmatrix}$$

```
M := <<<3, 4, 5> | <3, 4, 6> | <9, 2, 6>>>;
```

$$\begin{bmatrix} 3 & 3 & 9 \\ 4 & 4 & 2 \\ 5 & 6 & 6 \end{bmatrix}$$

方式（4）：由其他数据类型转换为矩阵或向量

```
L := [2, 4, 5, 5, 34, 2, 3, 4, 5];        #创建一个列表
v := convert(L, Vector);       #转换为向量
L2 := [[2, 3, 4], [3, 45, 65], [1, 2, 34]];
M := convert(L2, Matrix);       #转换为矩阵
```

2.6 表 (Table)

Maple 中表的数据类型：索引-值的存储机制，其中索引和值可以是任意的 Maple 对象。

```
t := table([3 = 5, foo = bar]);
t[3];
t[foo];
t[17, xyz];
t[17, xyz] := 1, 2, 3;
t[17, xyz];
```

当给出命名时，表不会自动显示其中的内容，仅显示命名。如果用户想查看其中的内容，可以使用 eval 命令获取表格内容。

表的另一个角色：如果变量没有值，例如 aaa，我们对 aaa[3]（或者其他关于 aaa 的索引形式）赋值，那么 Maple 会创建一个新的表，名称是 aaa，然后对其中的项赋值。

```
eval(aaa);
aaa[3] := 5;
```

eval(aaa);

用户可以使用 indices 命令获取已赋值的索引序列，使用 entries 命令获取相应的项。- 如果使用了 pairs 参数项，可以同时得到两者。注意 indices 和 entries 命令的结果是序列，原因是默认情况下它们封装为列表，但可以通过 nolist 参数项解除封装列表。

indices(t);

entries(t);

indices(t, pairs);

indices(t, nolist);

可以使用 entries(t, nolist) 命令得到列表形式的项，或者使用下面的命令：

convert(t, list);

第三章 常规的数学计算

本章内容介绍如何使用 Maple 求解特定领域的数学问题，求解问题时使用了我们经常使用的一些函数，但这些领域并不代表 Maple 所有的功能。其中的示例将引导我们如何使用不同的方法完成计算，包括向导、助手、命令、任务模板、绘图、以及右键菜单。

3.1 代数

Maple 提供大量的命令完成整数操作，例如因式分解和模数运算。

多项式代数：

一个 Maple 多项式是一个关于未知量幂的表达式。单变量多项式是关于一个变量的多项式，例如$x^3 - 2x + 13$。多元多项式是关于多个变量的多项式，例如$x^3y - \frac{3}{2}xy^2 + 7x$。系数可以是整数、有理数、浮点数、复数、变量、或者这些类型的组合。

```
[>a*x^2 + 7*x - b/2;
```

$$a x^2 + 7 x - \frac{1}{2} b$$

运算：

多项式运算符是标准的 Maple 运算符，但不包括除法运算（/）。（除法运算符接受多项式参数，但不能完成多项式除法。）

多项式除法是一个重要的操作。quo 和 rem 命令可以求多项式除法运算的商和余数。

操作	运算符	示例
加	+	[>(x^2 + 1) + (3*x^3 - 5*x + 2)
减	-	[>(x^2 + 1) - (3*x^3 - 5*x+ 2)
乘	*	[>(x^2 + 1)*(3*x^3 - 5*x + 2)
除：商和余数	quo rem	[>quo(2*x^2 + x - 3, 3*x + 5, x) [>rem(2*x^2 + x - 3, 3*x + 5, x)
幂	^	[>(x^2 + 1)^3

使用 expand 命令，展开一个多项式。

[>expand(3*x^2*(3*x + 5) + (-x^2 + 2));

如果你希望判断一个多项式能否除以另一个多项式，但不想要商，那么可以使用 divide 命令。divide 命令测试精确的多项式除法。

[>divide(x^4*y^2 + x^3*y^2 - x^2*y^2 + x^2*y + 13*x^2 + x*y + 13*x - y - 13, x^2 + x - 1)

true

注意：你必须在相邻的变量名之间插一个空格符号或乘法运算符 (*)。否则，Maple 会将它们解析为单个变量。

例如，如果没有空格，xy，将被解析为单个变量，x 不能被单变量 xy 整除。

[>divide(xy, x)

false

但是，x 可以被 x 和 y 的积相除。

[>divide(x*y, x);

true

多项式的项排序：

使用 sort 命令对多项式中的项进行排序。

[>p1 := x^2+x^4 + x^3 – x:

[>sort(p1);

$$x^4 + x^3 + x^2 - x$$

指定多项式中要排序的未知量和它们的排序，包括变量的列表。

[>sort(a^2*x^3 + a*x + x^2 + a + b, [a]);

$$x^3 a^2 + x a + a + x^2 + b$$

[>sort(a^2*x^3 + a*x + x^2 + a + b, [x, b])

$$a^2 x^3 + x^2 + a x + b + a$$

默认情况下，sort 命令依据项的总次数递减排序。

[>p2 := x^2*y^2 + x^3 + y^3;

[>sort(p2, [x, y]);

$$x^2 y^2 + x^3 + y^3$$

第一个项的总次数是 4。其他项总的次数是 3。最后两个项的排序是根据它们在列表中的

变量名排序的。

如果想要按照纯字母排序，也就是说，首先按照第一个未知量在列表中的降序排列，然后根据下一个变量在列表中的降序排列，在调用格式中使用 'plex' 参数项。

```
[>sort(p2, [x, y], 'plex')
```

第一个项中 x 的幂次是 3；第二项中 x 的幂次是 2；第三项中 x 的幂次是 0。

合并多项式

使用 collect 命令完成多项式项的合并操作。

```
[>collect(2*a*x*y + c*x^2*y - z*y^2 + a*z - 13*b*y + 3*y^2/x, y)
```

$$\left(-z+\frac{3}{x}\right)y^2+\left(2\,a\,x+c\,x^2-13\,b\right)y+a\,z$$

系数和次数

Maple 提供多个命令返回多项式的系数和次数的值。

命令	功能描述	示例
coeff	指定次数项的系数	[>coeff(1/2*x^3 - 2*x + 5, x^3)
lcoeff	首项系数	[>lcoeff(1/2*x^3 - 2*x + 5)
tcoeff	未项系数	[>tcoeff(1/2*x^3 - 2*x + 5)
coeffs	所有系数的序列，与项一一对应显示。 注意: 不返回零系数	[>coeffs(1/2*x^3 - 2*x + 5)
degree	（最高）次数	[>degree(1/2*x^3 - 2*x + 5)
ldegree	非零系数的最低次数项	[>ldegree(1/2*x^3 - 2*x)

多项式操作的其他命令

命令	功能描述
content	内容（多元多项式）
compoly	分解
discrim	判别式
gcd	（两个多项式的）最大公因式
gcdex	（两个多项式的）拓展欧几里德算法
CurveFitting[PolynomialInterpolation]	多项式插值（点的列表）

lcm	（两个多项式的）最小公倍式
norm	范数
EPROM	（两个多元多项式的）伪余式
primpart	（多元多项式的）本原部分
randpoly	随机多项式
PolynomialTools[IsSelfReciprocal]	判断是否是自反多项式
resultant	（两个多项式的）结式
roots	精确根（代数数域）
sqrfree	不含平方项的因式分解函数（多元多项式）

3.2 求方程与不等式

Maple 中主要有两个命令求解方程或方程组，求解一个或多个方程的精确解：

[> solve({方程 1,方程 2,…,方程 *n*}，{未知量 1,未知量 2,…,未知量 *m*})；

求解一个或多个方程的浮点数解：

[> fsolve({方程 1,方程 2,…,方程 *n*}，{未知量 1,未知量 2,…,未知量 *m*})。

[例3.1] 求方程$\frac{7x^2}{3}-x=12$的解。

解 在 Maple 中做如下运算：

```
[> restart:      #清除内存
[> eq:=(7*x^2)/3 - x = 12;      #定义方程 eq
```

$$eq := \frac{7}{3}x^2 - x = 12$$

```
[> solve( {eq} );      #用 solve 命令求解方程
```

$$\left\{x=\frac{3}{14}+\frac{3\sqrt{113}}{14}\right\},\left\{x=\frac{3}{14}-\frac{3\sqrt{113}}{14}\right\}$$

```
[> fsolve( {eq} );      #求浮点数解（近似解）
```

$$\{x=-2.0636026740\},\{x=2.4921741030\}$$

[例3.2] 求解方程组：$\begin{cases}x^2+y^2=2xy+4\\x+y=1\end{cases}$。

解：在 Maple 中做如下运算：

```
[> restart:      #清除内存
[> solve( {x^2+y^2=2*x*y+4, x+y=1},{x,y} );      #求解方程组（集合形式），{x,y}是未知量
```

$$\left\{x=\frac{3}{2},y=-\frac{1}{2}\right\},\left\{x=-\frac{1}{2},y=\frac{3}{2}\right\}$$

```
[> fsolve( {x^2+y^2=2*x*y+4, x+y=1}, {x,y} );      #求解方程组（集合形式）的数值解
```

$$\{x=1.500000000, y=-0.5000000000\}$$

因此，所求方程组的解为 x_1=-0.5, y_1=1.5 和 x_2=1.5, y_2=-0.5。

在求带参数方程或方程组解时，需要使用 parametric 参数项。

[例3.3] 求解方程组：$\begin{cases}x=2\\x=c\end{cases}$。

解：在 Maple 中做如下运算：

```
[> restart:
[> solve({x=2,x=c},{x});
[> solve({x=2,x=c},{x},'parametric');
```

$$\begin{cases}\%SolveTools_{Engine}(\{x-2,x-c\},\{x\}) & -2+c\neq 0\\ [\{x=2\}] & -2+c=0\end{cases}$$

```
[> value(%);
```

$$\begin{cases}[\,] & -2+c\neq 0\\ [\{x=2\}] & -2+c=0\end{cases}$$

可以发现 solve 不能求解参数方程，而使用 solve/parametric 命令运算结果是：当 c=2 时，方程组有解 $x=2$；当 $c\neq 2$ 时，方程组无解。

方程组消元

如要消去一方程组中某些变量，可使用 eliminate 命令来实现，其格式如下：

[> eliminate({方程 1,方程 2,…,方程 *n*},{消去变量 1,消去变量 2,…,消去变量 *m*});

[例3.4] 削去方程组：$\begin{cases}x^2+y^2+z^2=1\\x+y+z=0\end{cases}$ 中的变量 z。

解：在 Maple 中做如下运算：

```
[> restart:
[> eliminate({x^2+y^2+z^2=1,x+y+z=0},{z});
```

$$\left[\{z=-x-y\},\{2x^2+2y^2+2xy-1\}\right]$$

解得：削去方程组中的变量 *z* 后所得方程 $2x^2+2xy+2y^2=1$。

[例3.5] 求解多个方程或不等式。

```
[> solve([x*y^2 - y = 5, 0 < x])     #方程或不等式用集合或列表
```

$$\left\{x=\frac{y+5}{y^2},\ -5<y,\ y<0\right\},\ \left\{x=\frac{y+5}{y^2},\ 0<y\right\}$$

```
[> solve({x*y^2 - y = 5, x < 0});
```

$$\left\{x=\frac{y+5}{y^2},\ y<-5\right\}$$

[例3.6] 求解超越方程。

通常，solve 命令返回超越方程的单个解。

```
[> restart:
[> equation1 := sin(x) = cos(x);
```

$$\frac{\pi}{4}$$

要想得到所有的解，使用参数项 allsolutions.

```
[> solve(equation1, allsolutions = true);
```

$$\frac{1}{4}\pi+\pi\ _Z1\sim$$

Maple 返回的上述结果含有一个形式为_ZN~的变量，其中 N 是一个正整数，表示任意的整数。波浪线(~)表示这个一个含有假设条件的量。

RootOf 结构：

solve 命令有时会返回隐式的解，以 RoofOf 形式表示。例如求解高阶多项式时。

```
[> [solve(x^5 - 2*x^4 + 3*x^3 - 2)];
```

$$\left[1,\ RootOf\left(_Z^4-_Z^3+2\,_Z^2+2\,_Z+2,\ index=1\right),\right.$$
$$RootOf\left(_Z^4-_Z^3+2\,_Z^2+2\,_Z+2,\ index=2\right),$$
$$RootOf\left(_Z^4-_Z^3+2\,_Z^2+2\,_Z+2,\ index=3\right),$$
$$\left.RootOf\left(_Z^4-_Z^3+2\,_Z^2+2\,_Z+2,\ index=4\right)\right]$$

与其他符号表达式一样，可以使用 evalf 命令将 RootOf 结构转换为浮点数。

```
[> evalf(%)
```

$$[1.0000000000,\ 0.9840010519+1.5265908340\ I,$$
$$-0.4840010519+0.6099471405\ I,\ -0.4840010519$$
$$-0.6099471405\ I,\ 0.9840010519-1.5265908340\ I]$$

3.3 矩阵计算与线性代数

LinearAlgebra 函数包主要面向矩阵和向量数据结构。LinearAlgebra 函数包是 Maple 用于线性代数计算的函数包，共有 100 多个函数，所有的函数名均以大写字母开头。

LinearAlgebra 函数包提供的程序可用于构造和处理矩阵、向量，进行标准操作，查询结果

及求解线性代数中的问题。

输出矩阵时，10x10 或更小的矩阵及 10x1 或更小的向量在 Maple 工作表中显式输出，更大的矩阵或向量则以占位符的形式显示输出。要想查看矩阵或向量的元素或结构化信息，可用鼠标左键双击此占位符。

3.3.1 矩阵计算

前面《数据结构》中介绍了创建矩阵的几种方式。

加载 LinearAlgebra 线性代数函数包。

```
[> with(LinearAlgebra):
```

访问矩阵和向量中的元素项

选择矩阵中的元素项，方法是使用索引值。

```
[> M := <<-4.3, -6.7, 1.9> | <2.9, -1.2, 9.6> | <9.3, -8.0, -9.2>>;
```

$$\begin{bmatrix} -4.3000000000 & 2.9000000000 & 9.3000000000 \\ -6.7000000000 & -1.2000000000 & -8.0000000000 \\ 1.9000000000 & 9.6000000000 & -9.2000000000 \end{bmatrix}$$

```
[> M[1, 3];
```

$$9.3$$

选择整行，输入单个索引值；选择整列，输入整行，1..-1，然后是列索引值。

```
[> M[2];
```

$$\begin{bmatrix} -6.7 & -1.2 & -8.0 \end{bmatrix}$$

```
[> M[1 .. -1, 1];
```

$$\begin{bmatrix} -4.3 \\ -6.7 \\ 1.9 \end{bmatrix}$$

类似地，你可以访问子矩阵。以列表或范围的形式输入索引值。

```
[> M[2 .. 3, 1 .. 2];
```

$$\begin{bmatrix} -6.7 & -1.2 \\ 1.9 & 9.6 \end{bmatrix}$$

矩阵元素项操作

```
[> <<1 | 2>, <3 | 4>> . <<5 | 6>, <7 | 8>>;
```

$$\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

```
[> sin~(<<1.00 | 2.0>, <3.0 | 4.0>>);        #对所有元素项映射 sin 函数
```

$$\begin{bmatrix} 0.8414709848 & 0.9092974268 \\ 0.1411200081 & -0.7568024953 \end{bmatrix}$$

矩阵的加减乘除

```
[> M1 := LinearAlgebra:-RandomMatrix(3,3);
[> M2 := LinearAlgebra:-RandomMatrix(3,3);
[ > M1 + M2;
[ > M1 - M2;
[> M1.M2;
[> 12*M1; #标量相乘
[> M1^3;
[> 1/M1;
```

转置

```
[> Transpose(M1);
```

逆矩阵

```
[> M1^(-1);
```

或者使用

```
[> MatrixInverse(M1);
```

求行列式

```
[> LinearAlgebra:-Determinant(M1);
```

计算矩阵的特征值。

```
[> LinearAlgebra:-Eigenvalues(M1);
```

计算矩阵的特征向量。

```
[> LinearAlgebra:- Eigenvectors(M1);
```

求矩阵的行空间的一组基。

```
[> M3:= LinearAlgebra:-RandomMatrix(3,4);
[> LinearAlgebra[RowSpace](M3);
```

求矩阵的零度零空间的一组基。

```
[> LinearAlgebra[NullSpace](M3);
```

求矩阵的值域的一组基。

```
[> LinearAlgebra[ColumnSpace](M3);
```

求矩阵的秩。

```
[> LinearAlgebra[Rank](M3);
```

计算矩阵的零度。

```
[> LinearAlgebra[ColumnDimension] (M3);
```

3.3.2 矩阵 LU 分解

LUdecomp(A, P='p', L='l', U='u' , U1='u1', R='r', rank='ran', det='d');

其中，A 为矩阵(长方形)，P='p'—主元素因子，L='l'—单位下三角因子，U='u'—上三角因子，U1='u1'—修改的 U 因子，R='r'—行减少因子，rank='ran'—A 的秩，det='d'—U1 的行列式。当然，此命令的简写形式为：LUdecomp(A);

```
[> restart;
[> with(LinearAlgebra);
[> A := VandermondeMatrix([1, 3, 5, 7]);
```

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 9 & 27 \\ 1 & 5 & 25 & 125 \\ 1 & 7 & 49 & 343 \end{bmatrix}$$

```
[> LinearAlgebra:-LUDecomposition(A);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 3 & 3 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 2 & 8 & 26 \\ 0 & 0 & 8 & 72 \\ 0 & 0 & 0 & 48 \end{bmatrix}$$

```
[> LinearAlgebra[Rank](A);
```

4

```
[> LinearAlgebra[Determinant](A);
```

768

3.3.3 矩阵 QR 分解

在 QR 分解中，矩阵 A 被看作乘积 Q•R，此处，Q 为正交或归一化矩阵，R 为上三角阵。该分解是对一系列线性无关向量作 Gram-Schmidt 正交化，因此，Q 包含了正交的向量，R 的列记录了产生原向量的线性组合。命令的调用格式为：

QRDecomposition(A, fs, out, c, outopts, ...)

```
[> B := Matrix(3, 3, [1, 2, 3, 4, 5, 6, 7, 8, 10]);
[> LinearAlgebra[QRDecomposition](B);
```

$$\begin{bmatrix} \frac{\sqrt{66}}{66} & \frac{3\sqrt{11}}{11} & \frac{\sqrt{6}}{6} \\ \frac{2\sqrt{66}}{33} & \frac{\sqrt{11}}{11} & -\frac{\sqrt{6}}{3} \\ \frac{7\sqrt{66}}{66} & -\frac{\sqrt{11}}{11} & \frac{\sqrt{6}}{6} \end{bmatrix}, \begin{bmatrix} \sqrt{66} & \frac{13\sqrt{66}}{11} & \frac{97\sqrt{66}}{66} \\ 0 & \frac{3\sqrt{11}}{11} & \frac{5\sqrt{11}}{11} \\ 0 & 0 & \frac{\sqrt{6}}{6} \end{bmatrix}$$

3.3.4 大型矩阵的数值计算

Maple 可以自动实现大型数值矩阵的多核计算。使用 Maple 内置的线性代数数值库，可以非常有效地计算包含浮点数据（包括硬件浮点数据和任意精度的软件浮点数据）的大型矩阵和向量。

为了更好地使用这些算法，您有必要提供给 Maple 尽可能多的关于问题结构的信息。Maple 即使不需要这些附加信息也完全可以解决您的问题，某些情况下对您而言已经足够。但如果关心计算效率，下面的信息将比较重要。

数值算法仅适用于浮点数（包括复浮点数），以及整数矩阵。如果您知道您正在使用浮点数或整数，需要在函数中使用 Datatype = option 告诉 Maple 这些信息。否则，Maple 将在处理计算之前花费更多的时间检测其中数据项的类型。

```
[> with(LinearAlgebra):
```

```
[> N := 4000;
[> M1 := RandomMatrix(N, N, datatype = float[8]);
[> M2 := RandomMatrix(N, N, datatype = float[8]);
[> M1 . M2;
[> MatrixInverse(M1);
```

在构建矩阵时添加参数项 outputoptions=[datatype=float[8]]。可以对比下面两种方式调用 LinearSolve 命令的运行时间。

```
[> A := RandomMatrix(300, 300, generator = 0 .. 1.0000000000);
[> b := RandomVector(300, generator = 0 .. 1.0000000000);
[> t := time();
[> x := LinearSolve(A, b);
[> time() – t;
```

1.3900000000

```
[> A := RandomMatrix(300, 300, generator = 0 .. 1.0000000000, outputoptions = [datatype = float[8]]);
[> b := RandomVector(300, generator = 0 .. 1.0000000000, outputoptions = [datatype = float[8]]);
[> t := time();
[> x := LinearSolve(A, b);
[> time() – t;
```

0.9690000000

3.3.5 线性代数函数列表

```
[> with(LinearAlgebra);
```

[`&x`, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm, BilinearForm, CARE, CharacteristicMatrix, CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation, ColumnSpace, CompanionMatrix, CompressedSparseForm, ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct, DARE, DeleteColumn, DeleteRow, Determinant, Diagonal, DiagonalMatrix, Dimension, Dimensions, DotProduct, EigenConditionNumbers, Eigenvalues, Eigenvectors, Equal, ForwardSubstitute, FrobeniusForm,

FromCompressedSparseForm, FromSplitForm, GaussianElimination, GenerateEquations, GenerateMatrix, Generic, GetResultDataType, GetResultShape, GivensRotationMatrix, GramSchmidt, HankelMatrix, HermiteForm, HermitianTranspose, HessenbergForm, HilbertMatrix, HouseholderMatrix, IdentityMatrix, IntersectionBasis, IsDefinite, IsOrthogonal, IsSimilar, IsUnitary, JordanBlockMatrix, JordanForm, KroneckerProduct, LA_Main, LUDecomposition, LeastSquares, LinearSolve, LyapunovSolve, Map, Map2, MatrixAdd, MatrixExponential, MatrixFunction, MatrixInverse, MatrixMatrixMultiply, MatrixNorm, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply, MinimalPolynomial, Minor, Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm, ProjectionMatrix, QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row, RowDimension, RowOperation, RowSpace, ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues, SmithForm, SplitForm, StronglyConnectedBlocks, SubMatrix, SubVector, SumBasis, SylvesterMatrix, SylvesterSolve, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix, ZeroVector, Zip]

命令	功能简述
`&x`	向量的叉积
Add	矩阵或向量的对位相加
Adjoint	伴随矩阵
BackwardSubstitute	求解 A . X = B，其中 A 为上三角型行阶梯矩阵
BandMatrix	带状矩阵
Basis	返回向量空间的一组基
BezoutMatrix	构造两个多项式的 Bezout 矩阵
BidiagonalForm	将矩阵约化为双对角型
BilinearForm	向量的双线性形式
CARE	求解连续代数 Riccati 方程
CharacteristicMatrix	构造特征矩阵
CharacteristicPolynomial	构造矩阵的特征多项式
Column	返回矩阵的一个列向量序列
ColumnDimension	求矩阵的行数
ColumnOperation	对矩阵作出等列变换
ColumnSpace	返回矩阵列空间的一组基

CompanionMatrix	构造一个首一（或非首一）多项式或矩阵多项式的友矩阵（束）
CompressedSparseForm	计算压缩的稀疏行和列形式
ConditionNumber	计算矩阵关于某范数的条件数
ConstantMatrix	构造常数矩阵
ConstantVector	构造常数向量
Copy	构造矩阵或向量的一份复制
CreatePermutation	将一个 NAG 主元向量转换为一个置换向量或矩阵
CrossProduct	向量的叉积
DARE	求解离散代数 Riccati 方程
DeleteColumn	删除矩阵的行
DeleteRow	删除矩阵的列
Determinant	行列式
Diagonal	返回从矩阵中得到的向量序列
DiagonalMatrix	构造（分块）对角矩阵
Dimension	行数和列数
DotProduct	点积
EigenConditionNumbers	计算数值特征值制约问题的特征值或特征向量的条件数
Eigenvalues	计算矩阵的特征值
Eigenvectors	计算矩阵的特征向量
Equal	比较两个向量或矩阵是否相等
ForwardSubstitute	求解 A . X = B，其中 A 为下三角型行阶梯矩阵
FrobeniusForm	将一个方阵约化为 Frobenius 型（有理标准型）
FromCompressedSparseForm	将压缩的稀疏行和列形式转换为 Maple 原有形式
FromSplitForm	将分拆形式转为 Maple 原有形式
GaussianElimination	对矩阵作高斯消元
GenerateEquations	从系数矩阵中生成方程

GenerateMatrix	从方程中生成系数矩阵
GetResultDataType	返回矩阵或向量运算的结果数据类型
GetResultShape	返回矩阵或向量运算的结果形状
GivensRotationMatrix	构造 Givens 旋转的矩阵
GramSchmidt	计算一个正交向量集
HankelMatrix	构造一个 Hankel 矩阵
HermiteForm	计算一个矩阵的 Hermite 正规型
HermitianTranspose	共轭转置矩阵
HessenbergForm	将一个方阵约化为上 Hessenberg 型
HilbertMatrix	构造广义 Hilbert 矩阵
HouseholderMatrix	构造 Householder 反射矩阵
IdentityMatrix	构造一个单位矩阵
IntersectionBasis	返回向量空间交的一组基
IsDefinite	检验矩阵的正定性，负定性或不定性
IsOrthogonal	检验矩阵是否正交
IsSimilar	检验两个矩阵是否相似
IsUnitary	检验矩阵是否为酉矩阵
JordanBlockMatrix	构造约当块矩阵
JordanForm	将矩阵约化为约当型
KroneckerProduct	构造两个矩阵的 Kronecker 张量积
LeastSquares	方程的最小二乘解
LinearSolve	求解线性方程组 A . x = b
LUDecomposition	计算矩阵的 Cholesky，PLU 或 PLU1R 分解
LyapunovSolve	求解连续李雅普诺夫方程
Map	将一个程序映射到一个表达式上，对矩阵和向量在原位置上进行处理
Map2	将一个程序映射到一个表达式上，对矩阵和向量在原位置

	上进行处理
MatrixAdd	计算两个矩阵的线性组合
MatrixExponential	确定一个矩阵 A 的矩阵指数 exp(A)
MatrixFunction	确定方阵 A 的函数 F(A)
MatrixInverse	计算方阵的逆或矩阵的 Moore-Penrose 伪逆
MatrixMatrixMultiply	计算两个矩阵的乘积
MatrixNorm	计算矩阵的 p-范数
MatrixPower	矩阵的幂
MatrixScalarMultiply	计算矩阵与数的乘积
MatrixVectorMultiply	计算一个矩阵和一个列向量的乘积
MinimalPolynomial	构造矩阵的最小多项式
Minor	计算矩阵的子式
Multiply	矩阵相乘
Norm	计算矩阵或向量的 p-范数
Normalize	向量正规化
NullSpace	计算矩阵的零度零空间
OuterProductMatrix	两个向量的外积
Permanent	方阵的不变量
Pivot	矩阵元素的主元消去法
PopovForm	Popov 正规型
ProjectionMatrix	构造一个子空间上的正交投影的矩阵
QRDecomposition	QR 分解
RandomMatrix	构造随机矩阵
RandomVector	构造随机向量
Rank	计算矩阵的秩
RationalCanonicalForm	将一个方阵化简为 Frobenius 型(有理标准型)
ReducedRowEchelonForm	对矩阵作高斯－约当消元

Row	返回矩阵的一个行向量序列
RowDimension	求矩阵的行数
RowOperation	对矩阵作初等行变换
RowSpace	返回矩阵行空间的一组基
ScalarMatrix	构造一个单位矩阵的数量倍数
ScalarMultiply	矩阵与数的乘积
ScalarVector	构造一个单位向量的数量倍数
SchurForm	将方阵约化为 Schur 型
SingularValues	计算矩阵的奇异值
SmithForm	将矩阵约化为 Smith 正规型
StronglyConnectedBlocks	计算方阵的强连通块
SplitForm	计算稀疏表的拆分形式
StronglyConnectedBlocks	计算方阵的强连通块
SubMatrix	构造矩阵的子矩阵
SubVector	构造向量的子向量
SumBasis	返回向量空间直和的一组基
SylvesterMatrix	构造两个多项式的 Sylvester 矩阵
SylvesterSolve	求解 Sylvester 矩阵方程
ToeplitzMatrix	构造 Toeplitz 矩阵
Trace	计算方阵的迹
Transpose	转置矩阵
TridiagonalForm	将方阵约化为三对角型
UnitVector	构造单位向量
VandermondeMatrix	构造一个 Vandermonde 矩阵
VectorAdd	计算两个向量的线性组合
VectorAngle	计算两个向量的夹角
VectorMatrixMultiply	计算一个行向量和一个矩阵的乘积

VectorNorm	计算向量的 p-范数
VectorScalarMultiply	计算向量与数的乘积
ZeroMatrix	构造一个零矩阵
ZeroVector	构造一个零向量
Zip	将一个具有两个参数的程序作用到一对矩阵或向量上
LinearAlgebra[Generic] 子函数包	[Generic]子函数包提供作用在场，欧几里得域，积分域和环上的线性代数算法。命令列表和详细信息见帮助系统。
LinearAlgebra[Modular]子函数包	[Modular]子函数包提供一组工具用于完成在 Z/m 稠密线性代数计算，整数模 m。

3.4 微积分

Maple 内置大量的命令和工具处理微积分问题。下面的段落描述了 Maple 中的一些重要微积分计算命令，其中许多命令同样可以在任务模板中发现，或者可以通过右侧的关联菜单完成计算。

3.4.1 极限

当其中的自变量逼近某一个值时，计算表达式的的极限值：

表达式 面板中，点击极限$\lim_{x \to a} f$。

定义自变量、极限点、表达式，然后对其求值。按 Tab 键可以切换到下一个占位符。

[> limit(x/sin(x), x = 0)

[> $\lim_{x \to 0}\left(\frac{x}{\sin(x)}\right)$

limit 命令

默认情况下，Maple 搜索双向极限值（除非极限点是∞或-∞）。想要定义方向，可在 limit 命令中加入 left, right, real, 或者 complex 参数项。

极限	命令的语法	输出
$\lim_{x \to 0}\left(\frac{1}{x}\right)$	[> limit(1/x, x = 0)	undefined

$\lim_{x \to 0+} \left(\frac{1}{x} \right)$	[> limit(1/x, x = 0, 'right')	infinity
$\lim_{x \to 0-} \left(\frac{1}{x} \right)$	[> limit(1/x, x = 0, 'left')	-infinity

使用 limit 命令，你也可以计算多元极限。

[> limit(x^2/y, {x = 1, y = infinity})

0

数值计算极限：

• 使用 evalf(Limit(参数)) 调用格式。

重要：使用惰性命令 Limit，而不是 limit。

Limit 命令接受与 limit 命令相同的参数项。

[> evalf(Limit(sin(x)/(cos(x) + tan(x)), x = 1.225))

0.3020605357

Limit 命令并不计算极限，而是返回未求值的极限（惰性）。

[> Limit(sin(x)/(cos(x) + tan(x)), x = 1.225)

$$\lim_{x \to 1.22500} \frac{\sin(x)}{\cos(x) + \tan(x)}$$

3.4.2 微分

Maple 可以完成符号和数值微分计算。

对表达式求微分：

1. 在 表达式 面板中，点击微分项 $\frac{d}{dx} f$ 或者偏微分项 $\frac{\partial}{\partial x} f$

2. 定义表达式和自变量，然后求值。

例如，求 $x \sin(a x)$ 关于 x 的微分：

[> diff(x*sin(a*x), x);

$$\sin(a x) + x \cos(a x)\, a$$

想要计算高阶或偏微分，需要编辑插入的微分符号。例如，计算$x \sin(a x) + x^2$ 关于 x 的二阶微分：

[> diff(x*sin(a*x) + x^2, x, x)

$$2 \cos(a x)\, a - x \sin(a x)\, a^2 + 2$$

计算$x\sin(3y)+yx^5$的混合偏导数：

```
[> diff(x*sin(3*y) + y*x^5, y, x)
```

注意：想要插入其他的∂符号，你可以通过拷贝和粘帖已有的符号，或者输入字母 d 然后按 ESC 符号补全。

diff 命令

Maple 使用 diff 命令对表达式求微分。通常的用法是 diff(expr,var)，其中 var 是要求微分的变量。例如：

```
[> diff(x*sin(a*x) + x^2, x);
```

可以通过定义一组微分变量计算高阶微分。Maple 递归地调用 diff 命令。

```
[> diff(x*sin(3*y) + y*sqrt(x), x, y)
```

想要计算偏微分，使用相同的语法。Maple 会假设为偏微分计算。

如果要对一个变量多次求导，可以使用 diff(f, x$n)，它实际上是一种缩写的形式，n 代表变量 x 重复的次数。这个语法也可以用于计算符号 nth 阶微分。

例如：

```
[> diff(cos(t), t $ n)
```

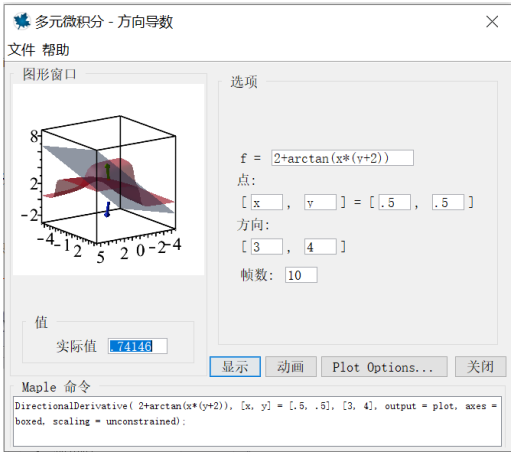
$$\cos\left(t+\frac{1}{2}n\pi\right)$$

3.4.3 方向导数

使用方向导数向导，计算并可视化方向导数。向导计算了方向导数的浮点数值。

使用方向导数向导：

从工具菜单，选择助教→微积分-多变量→方向导数。Maple 打开方向导数向导。



想要计算方向导数的符号值，使用 Student[MultivariateCalculus][DirectionalDerivative] 命令。第一列数定义求导数的点，第二列数定义求导的方向。

例如，在 [1, 2]点上，x^2+y^2 在方向 [2, 4]的梯度点。在正交方向[-2, 1]的方向导数是零。

```
[> with(Student[MultivariateCalculus]);
[> DirectionalDerivative(x^2 + y^2, [x, y] = [1, 2], [1, 2]);
[> DirectionalDerivative(x^2 + y^2, [x, y] = [1, 2], [-2, 1]);
```

3.4.4 级数

使用 taylor 命令，生成函数在一点上的泰勒级数展开。

```
[> taylor(sin(4*x)*cos(x), x = 0)
```

$$4x-\frac{38}{3}x^3+\frac{421}{30}x^5+O\left(x^6\right)$$

注意: 如果泰勒级数不存在，可以使用 series 命令发现广义级数展开。

例如余弦积分函数在 0 点上没有泰勒级数展开。

```
[> taylor(Ci(x), x = 0)
```

Error, does not have a taylor expansion, try series()

想要生成函数在某一点上的截断级数展开，可以使用 series 命令。

```
[> series(Ci(x), x = 0)
```

$$\gamma+\ln(x)-\frac{1}{4}x^2+\frac{1}{96}x^4+O\left(x^6\right)$$

默认情况下，Maple 的级数计算阶数为 6。如果想使用其他阶数，需要定义一个非负整数参数项。

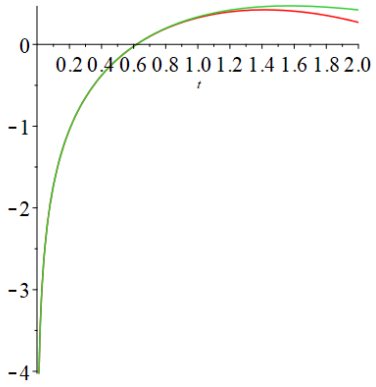
```
[> expansion := series(Ci(t), t = 0, 4)
```

$$expansion := \gamma + \ln(t) - \frac{1}{4}t^2 + O(t^4)$$

使用 Order 环境变量，可以定义所有计算的阶数。

一些命令，例如 plot，不接受 series 类型的参数项。想要使用展开，必须使用 convert/polynom 命令将它转换为多项式。

```
[> plot({Ci(t), convert(expansion, polynom)}, t = 1/100 .. 2)
```



3.4.6 多变量和向量微积分

VectorCalculus 函数包提供命令完成多变量和向量微积分的操作，例 VectorCalculus vectors（含其他坐标系属性的向量），以及 vector fields（含其他坐标系和向量场属性，例如，Curl，Flux，和 Torsion）。

```
with(VectorCalculus);
BasisFormat(false);
SetCoordinates('cartesian[x, y, z]');
VectorField1 := VectorField(<-y, x, z>);
```

$$\begin{bmatrix} -y \\ x \\ z \end{bmatrix}$$

求 VectorField1 的旋度。

```
Curl(VectorField1);
```

$$\begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

求 VectorField1 通过半径为 r 的通量。

Flux(VectorField1, Sphere(<0, 0, 0>, r))

$$\frac{4}{3} r^3 \pi$$

计算空间曲线的挠度。曲线必须是一个参数函数。

(simplify(Torsion(<t, t^2, t^3>, t)) assuming t::real);

$$\frac{3}{9 t^4 + 9 t^2 + 1}$$

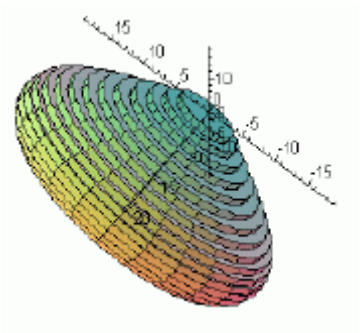
3.4.7 Student 微积分函数包和助教

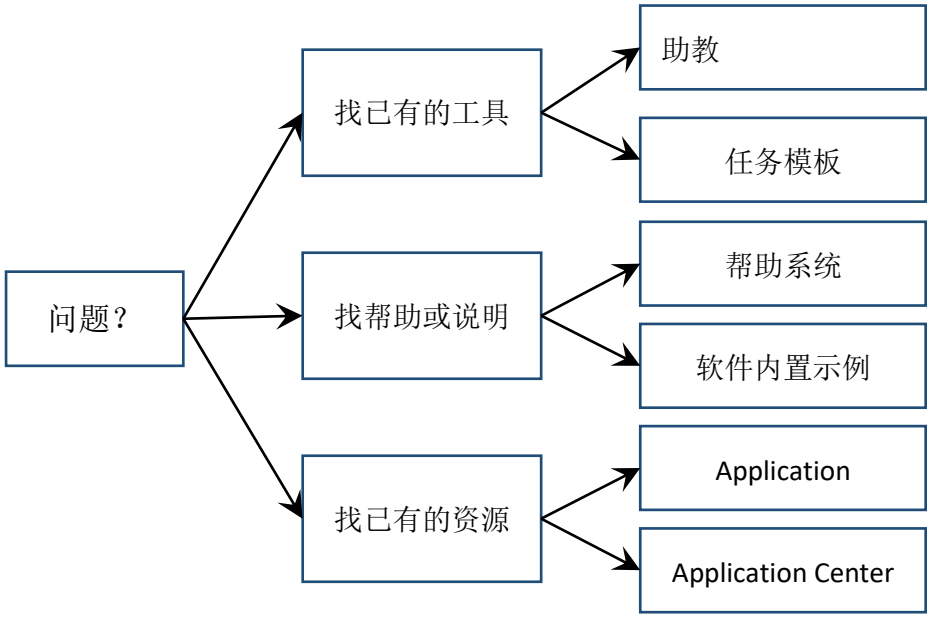
Student 函数包内置多个子函数包，主要用途是帮助老师课堂教学、学生学习本科高等数学、概率统计和线性代数的知识，Student 函数包具有一些独特的特点，包括可视化数学概念、逐步显示解题过程和原理、引导学生的数学思考和解决数学问题等。Student 微积分子函数包包括 Calculus1, MultivariateCalculus（多元微积分）和 VectorCalculus（向量微积分）。

3.4.8 微积分问题求解示例

问题：

应用场景 A: 某公司正在为其新的矿泉水设计一个瓶子。瓶子必须装有 18 盎司的水，并且高度是固定的。 该设计包括波浪形的曲面。已知曲线的振幅和方程式，但是必须找到半径。需要使用“旋转量”。	

应用场景 B： 你想教你的学生旋转体的概念。具体而言，你想绘制并计算通过绕轴或与轴平行的线旋转生成的实体的体积。	



图：解决问题的流程图

1）找已有的工具：助教

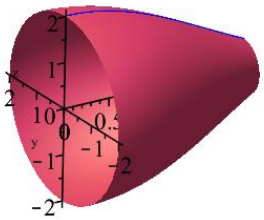
首先到工具菜单下的助教，查找解决旋转体问题的向导。

访问旋转体向导的途径：

1) 选择菜单工具 → 助教 → 微积分 – 单变量 → 旋转体的体积。

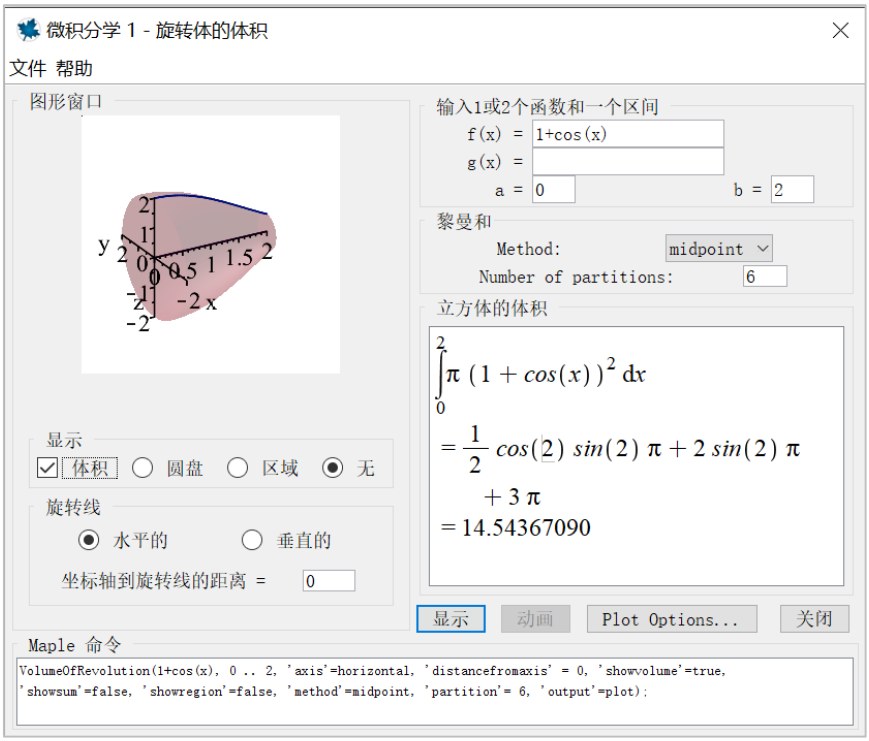
2) 点击“旋转体的体积”后，当前光标处出现如下的 Maple 命令。

```
[> Student[Calculus1][VolumeOfRevolutionTutor]();
```



弹出“旋转体的体积”窗口，如下图。在这个向导中可以输入函数和区间，提供工具查看

和操作对应的图形，以及查看相应的 Maple 命令和参数项。



2) 找已有的工具：任务模板

1) 选择菜单工具 → 任务 → 浏览，将打开浏览任务对话框，在左侧窗格中显示任务列表。任务按主题排序，以帮助你快速找到所需的任务。

2) 展开 Calculus - Integral→ Applications → Solids of Revolution 文件夹。

3) 从显示的列表中，选择 Volume。 “旋转体积”任务显示在“浏览任务”对话框的右窗格中。

4) 选中“插入到新的工作表中”复选框。

单击“插入默认内容”。在插入任务之前，Maple 检查任务变量是否已在工作表中分配值。如果分配了任何任务中的变量，则打开“任务变量”对话框，允许你修改名称。Maple 对插入的任务中的所有变量实例使用已编辑的变量名称。内容将插入到当前文件。如下图。

Volume of Revolution

Calculate the volume of revolution for a solid of revolution when a function is rotated about the horizontal or vertical axis.

Enter the function as an expression and specify the range:

```
> sin(x) cos(x) + 1, 0..π/2
```

$$\sin(x)\cos(x)+1,\ 0..\frac{1}{2}\,\mathrm{Pi} \qquad (1)$$

Calculate the volume of revolution:

```
> Student[Calculus1][VolumeOfRevolution]((1))
```

$$\mathrm{Pi}+\frac{9}{16}\,\mathrm{Pi}^2 \qquad (2)$$

Display the floating-point value using the evalf command:

```
> evalf((2))
```

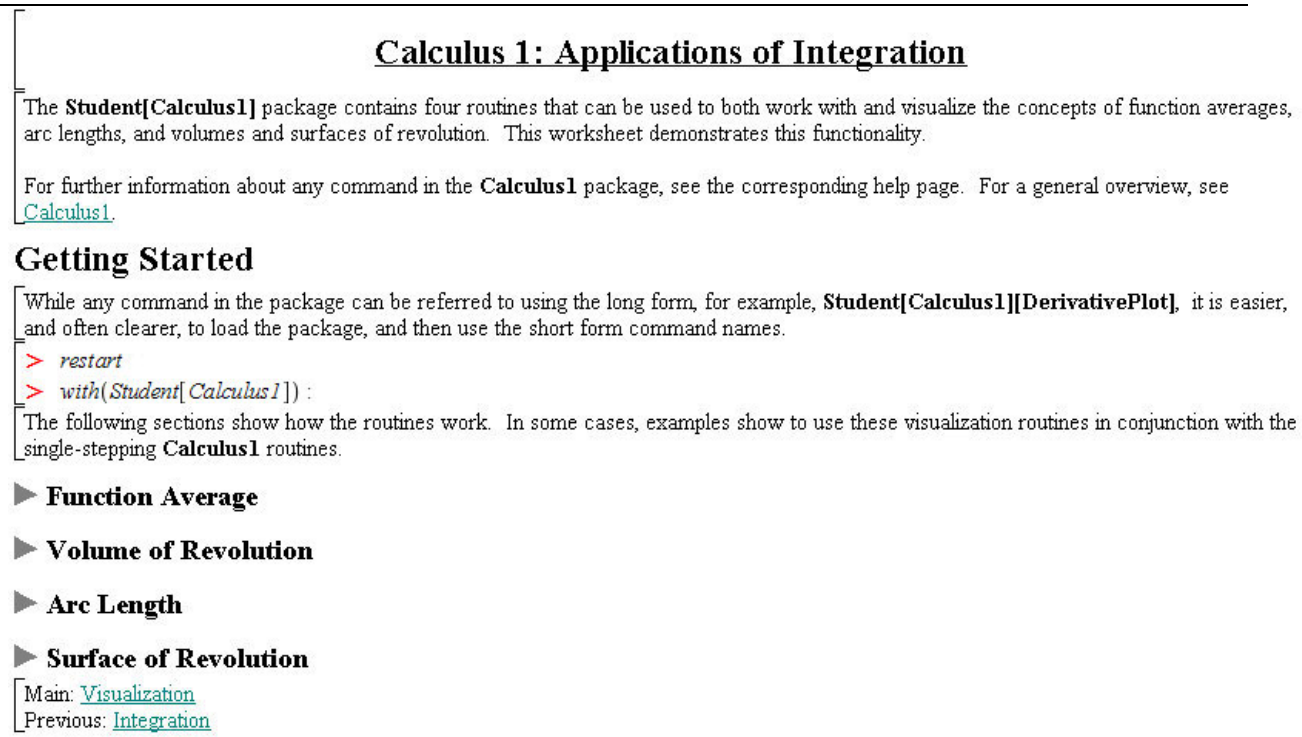
$$8.693245131 \qquad (3)$$

3）找帮助或说明：帮助页面和软件内置示例

帮助系统提供命令的语法信息。

进入帮助页面：

1) 从帮助菜单，选择 Maple 帮助。
2) 在搜索框中，输入 volume of revolution，然后点击搜索。搜索结果包括命令帮助页、字典定义和相关的向导帮助页。
3) 在 Student [Calculus1] [VolumeOfRevolution]帮助页面中查看调用格式、参数和说明。
4) 拷贝帮助页中的例子到你的工作表中。
5) 关闭帮助导航器。
6) 在文件中从编辑菜单选择粘贴，或者使用快捷键 CTRL + V.
7) 执行例子并检查结果。



Calculus 1: Applications of Integration

The **Student[Calculus1]** package contains four routines that can be used to both work with and visualize the concepts of function averages, arc lengths, and volumes and surfaces of revolution. This worksheet demonstrates this functionality.

For further information about any command in the **Calculus1** package, see the corresponding help page. For a general overview, see Calculus1.

Getting Started

While any command in the package can be referred to using the long form, for example, **Student[Calculus1][DerivativePlot]**, it is easier, and often clearer, to load the package, and then use the short form command names.

> restart
> with(Student[Calculus1]) :

The following sections show how the routines work. In some cases, examples show to use these visualization routines in conjunction with the single-stepping **Calculus1** routines.

▶ **Function Average**

▶ **Volume of Revolution**

▶ **Arc Length**

▶ **Surface of Revolution**

Main: Visualization
Previous: Integration

4）找已有的其他资源：Application Center

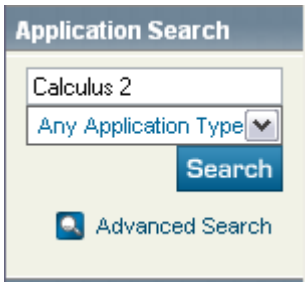
Maple Application Center 是一个由用户分享的应用程序中心，你可以免费下载，其中包含与数学、教育、科学、工程、计算机科学、统计和数据分析、财务、通信、图形等相关的免费应用案例。

获取免费的应用案例 volume of revolution:

1) 进入 Maplesoft 官方网站：http://www.maplesoft.com 或者 http://www.maplesoft.com.cn

2) 从主页菜单支持与资源进入 Application Center.

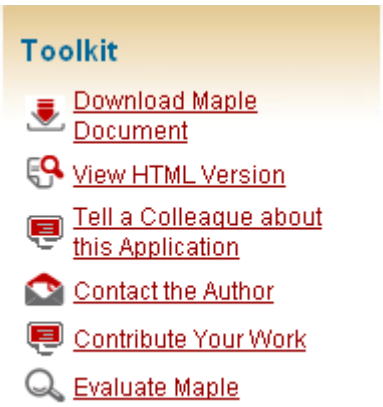
3) 在 Application Search 区域，输入关键词 Calculus 2.



4) 点击搜索。

5) 从搜索结果选择 Calculus II: Complete Set of Lessons.

6) 点击 Download Maple Document 链接。



7) 下载.zip 文件。

执行工作表并检查结果。

3.5 优化

优化的目标是从一组可能的答案中发现问题的最佳解。答案通过使用一个或多个问题变量的实际值目标函数进行对比。可能的集合由约束条件决定，约束条件通常是关于问题变量的不等式或方程（组）。数学意义上，目的是发现目标函数的最大值或最小值，同时满足约束条件的点，这个点称为极值。优化问题通常定义如下。

f(x)的最大值或最小值

约束条件：$g_i(x) \le 0$，$i = 1..p$

$h_i(x) = 0$，$i = 1..m$

优化模型由$f(x)$，$g_i(x)$和$h_i(x)$的结构分类。如果所有的函数是 x 线性函数，这个模型是一个线性规划。如果$f(x)$是 x 的二次函数，以及$g_i(x)$和$h_i(x)$是 x 的线性函数，这个模型是一个二次规划。如果$f(x)$是一个平方和函数，这个模型是一个非线性规划（最小二乘问题）。对于其他任意结构，模型称为非线性回归（NLP）。Maple 中的优化函数包提供了一系列算法分别求解这些类型的问题。

传统上，优化研究集中在局部搜索算法。局部搜索的目的是发现$f(x)$在可行区域内的局部极值。从一个初始点出发，局部搜索算法迭代搜索当前点范围中的一个点，提高目标函数值，同时维持或逼近可行性、使用当前点上关于$f(x)$，$g_i(x)$和$h_i(x)$的迭代信息。理想情况下，搜

索会终止于一个可行的局部极值。优化算法的不同决定它们如何衡量逼近可行，以及它们如何搜索。

当$f(x)$在可行区域内有唯一的局部极值时，局部搜索是有效的，这是因为搜索发现的局部解是问题的全局解。如果$f(x)$，$g_i(x)$都是凸函数，并且所有的$h_i(x)$是仿射函数时，极值是唯一的。

当$f(x)$在可行区域内有许多局部极值，局部搜索难以完成。在这种情况下，局部极值的发现依赖于起始点，但可能并不是全局解。当任一个$f(x)$，$g_i(x)$是非凸函数，或者任一个$h_i(x)$是非线性函数时，存在多个局部最小值。非凸性经常存在于现实问题中，可能是求解优化问题最大的障碍。全局优化是最新的和最成功方法，能够克服这个障碍。

Maple 提供两个优化函数包，其中 Optimization 函数包提供局部优化搜索算法，GlobalOptimization 工具箱提供全局优化搜索算法，此外 Maple 还提供一些开源优化算法，例如多目标优化。

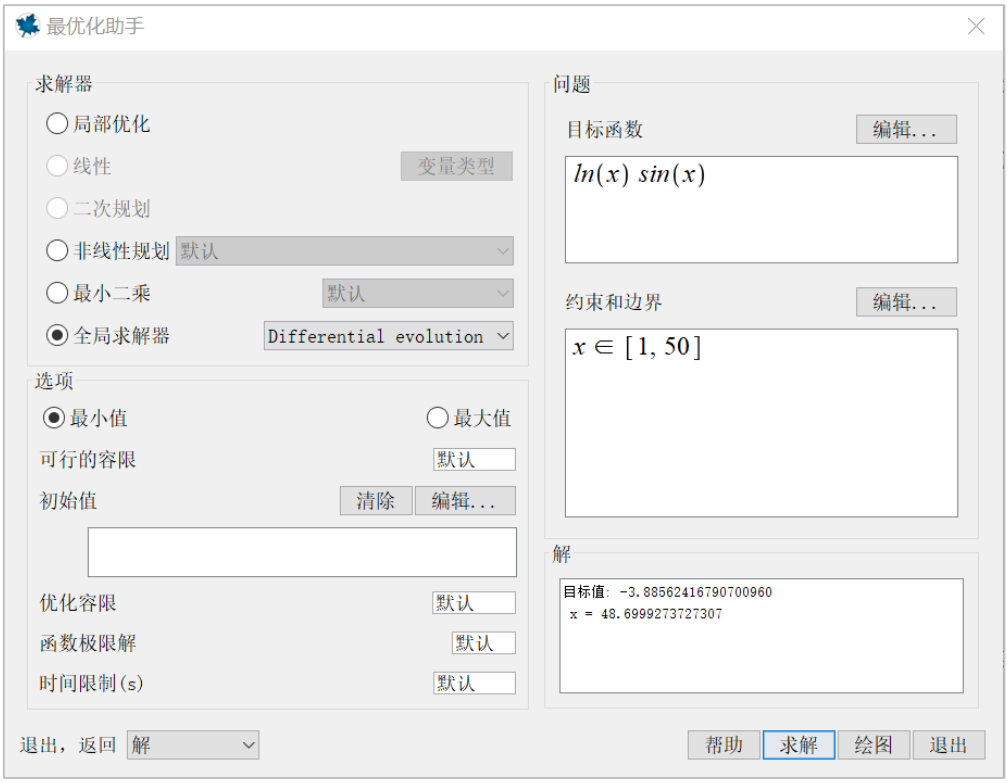
Optimization 函数包提供了一系列强大的优化算法数值求解优化问题，例如求受约束目标函数的最大值或最小值。通过使用优化函数包，您可以求解线性规划 (LPs)、二次规划(QPs)、非线性规划(NLPs)、以及线性和非线性最小二乘问题。优化函数包接受广泛的输入格式，包括代数形式、矩阵形式、Maple 程序。

加载优化函数包：

```
[> restart:
[> with(Optimization);
```

[*ImportMPS, Interactive, LPSolve, LSSolve, Maximize, Minimize, NLPSolve, QPSolve*]

或者调用图形用户界面：



3.5.1 线性规划

如前所述，Optimization 函数包可以处理广泛的输入形式。这里将介绍代数和矩阵输入两种方式。

[例3.7] 求函数 x + y 的最大值

求函数 x + y 的最大值

约束条件：$\begin{bmatrix} x + 2 \cdot y \geq 2 \\ x - y \leq 2 \\ 4 \cdot x - y \geq 0 \\ x \leq 4 \\ y \leq 6.5 \end{bmatrix}$

解：首先，定义目标函数：

ObjectiveFunctionLP1 := x + y;

$$x + y$$

然后定义约束的方程：

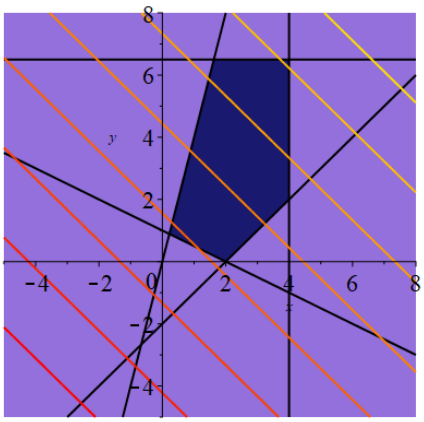
ConstraintEquationsLP1 := {0 <= 4*x - y, 2 <= x + 2*y, x <= 4, y <= 6.50000, x - y <= 2};

$$\{0 \leq 4\,x - y,\ 2 \leq x + 2\,y,\ x \leq 4,\ y \leq 6.50000,\ x - y \leq 2\}$$

对于一个两维线性规划，您可以轻松地用图形表示可行区域，如下图。黑线代表约束，红

线代表目标函数 x + y 的轮廓。

```
p1:=plots[inequal](ConstraintEquationsLP1, x = -5 .. 8, y = -5..8, thickness = 2, optionsexcluded = (color = "MediumPurple"), optionsfeasible = (color = "MidnightBlue"));
p2:=plots[contourplot](ObjectiveFunctionLP1, x = -5 .. 8, y = -5..8, thickness = 2);
plots[display](p1, p2);
```



使用 Maximize 命令，得到目标函数和约束方程的最大值。输出结果中的第一项是 10.5，是最优目标值，同时输出结果第二项指出一个点，(4, 6.5)，它是优化问题的坐标点。坐标点(4, 6.5)位于可行区域的右上角顶点。

```
Maximize(ObjectiveFunctionLP1, ConstraintEquationsLP1);
```

[10.50000, [x = 4.00000, y = 6.50000]]

类似的，Minimize 命令可用于求最小解。

```
Minimize(ObjectiveFunctionLP1, ConstraintEquationsLP1);
```

[1.11111, [x = 0.22222, y = 0.88889]]

或者，也可以使用 LPSolve 命令求目标函数的最大和最小值。LPSolve 相比 Maximize 和 Minimize 更加灵活，它可以接受矩阵形式的问题。

```
LPSolve(ObjectiveFunctionLP1, ConstraintEquationsLP1);
LPSolve(ObjectiveFunctionLP1, ConstraintEquationsLP1, maximize);
```

[1.11111111111111, [x = 0.222222222222222, y = 0.888888888888889]]

[10.5000000000000, [x = 4., y = 6.50000000000000]]

或者也可以使用交互式优化助手求解问题。优化助手（如下图所示）允许你通过点击方式输入、编辑、和求解（或再求解）问题，无需输入命令。调入优化助手的途径有：通过交互式

命令或工具菜单（工具→助教）。

在优化助手中，选择选项下的最小值或最大值，然后按求解。

MinValue := Interactive(ObjectiveFunctionLP1, ConstraintEquationsLP1);

MinValue := [1.11111, [*x* = 0.22222, *y* = 0.88889]]

MaxValue := Interactive(ObjectiveFunctionLP1, ConstraintEquationsLP1);

MaxValue := [1.11110, [*x* = 0.22222, *y* = 0.88889]]



注意：优化助手仅仅接受代数形式的输入。

[例3.8] “饮食问题”是一个线性规划优化问题，于 1940 年 Georg Danzig 提出。在给定营养值和食品单位价格的前提下，饮食问题试图发现能够满足我们日常营养需要的最便宜的饮食。

让我们用变量 Cost 定义 1000 个不同食品的成本，这里$Cost_i$是食品 *i* 单位总量的价格。

Cost := LinearAlgebra[RandomVector](1000, generator = 0 .. 1.0);

类似的，我们定义变量 *Nutrients*，一个 600x1000 的矩阵，这里$Nutrients_{i,j}$是单位食品 *j* 的营养值 *i* 的总和。

Nutrients := LinearAlgebra[RandomMatrix](600, 1000, generator = 0 .. 1.0);

最后，我们定义一个向量变量 *DailyRequirements*， 指定我们日常需要的 600 个营养物，这里$DailyRequirements_i$是对营养物 i 的日常需要。

DailyRequirements := LinearAlgebra[RandomVector](600, generator = 2 .. 20.0);

向量 x 是我们每天选择购买的食品数量。

最小值 $Cost^T \cdot x$

约束条件：$\begin{bmatrix} Nutrients \cdot x \geq DailyRequirements \\ x \geq 0 \end{bmatrix}$

解：求解线性问题如下：

LPSolve(Cost, [Nutrients, DailyRequirements])求解最小值，$Cost^T \cdot x$ 满足 $Nutrients \cdot x \leq DailyRequirements$。

MinCost := LPSolve(Cost, [-Nutrients, -DailyRequirements], assume = nonnegative);

$$MinCost := \begin{bmatrix} 0.34016, & \begin{bmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ \vdots \end{bmatrix} \\ & \text{1000 element Vector[column]} \end{bmatrix}$$

结果中有几个正值？

select(x -> 0 < x, convert(sol[2], list));

nops(%);

[9.39873, 32.03490, 6.65624, 26.88435, 13.32010]

5

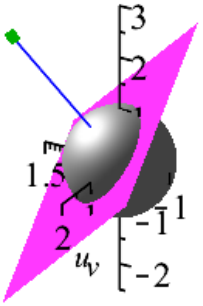
3.5.2 非线性规划

[例3.9] 单位球体和平面$x + y + z = \frac{1}{2}$交叉形成一个圆圈，求圆圈上距离点 (1, 2, 3) 最近的点。

解：通过绘制单位球体和定义交叉平面的方程的图形，可视化这个问题。

```
[> p1 := plottools[sphere]([0, 0, 0], 1, color = "Grey");
```

```
[> p2 := plot3d(1/2 - u - v, u = -1 .. 1.50000, v = -1 .. 1.50000, color = "Magenta");
[> p3 := plottools[line]([0, 0, 0], [1, 2, 3], color = blue, thickness = 1);
[> p4 := plots[pointplot3d]([[1, 2, 3]], color = "Green", symbolsize = 30);
[> plots[display]([p1, p2, p3, p4], scaling = constrained, style = patchnogrid, axes = normal, orientation = [120, 64], lightmodel = light3);
```



目标函数是点 (x, y, z)和点 (1, 2, 3) 之间距离的平方。

```
[> ObjectiveFunctionNLP1 := (x - 1)^2 + (y - 2)^2 + (z - 3)^2;
```

$$ObjectiveFunctionNLP1 := (x-1)^2 + (y-2)^2 + (z-3)^2$$

点 (x, y, z) 限制在同时位于单位球体和给定平面上。因此约束条件是：

```
[> ConstraintsEquationsNLP1 := {x + y + z = 1/2, x^2 + y^2 + z^2 = 1};
```

$$ConstraintsEquationsNLP1 := \left\{x + y + z = \frac{1}{2}, x^2 + y^2 + z^2 = 1\right\}$$

我们使用 NLPSolve 命令最小化包含约束条件的目标函数。NLPSolve 命令可以解决常规的问题。只要目标函数和约束函数的导数是连续的，就可以使用 NLPSolve。

```
[> sol := NLPSolve(ObjectiveFunctionNLP1, ConstraintsEquationsNLP1);
```

$$sol := [10.2919871984546809, [x = -0.510336533719664, y = 0.166666666666666, z = 0.843669867052997]]$$

因此，最近距离是 10.29，最近的点是(-0.51, 0.17, 0.84)。

3.5.3 最小二乘优化问题

这里举例使用最小二乘法实现数据的曲线拟合。

```
restart;
with(Optimization);
```

如果我们有下面的数据点，(x[i], y[i])：

```
data := [[1, 1.50000], [2, 3.50000], [2.50000, 1.90000], [3.10000, 4.50000], [4.30000, 1.90000], [4.70000, 2.40000], [5.80000, 3], [6, 5.70000], [6.60000, 3.40000], [6.70000, 0.50000], [7.10000, 4], [7.40000, 1.70000]];
```

我们希望使用 5 次多项式拟合曲线。假定 5 次多项式形式是：

$p(x) = a\ x^5 + b\ x^4 + c\ x^3 + d\ x^2 + e\ x + f$.

现在我们需要求系数的值，满足$\frac{1}{2}\sum_{x=1}^{12}\left(p\left(x_i\right) - y_i\right)^2$是最小值。

LSSolve 接受输入一列残差$p\left(x_i\right) - y$。

```
p := a*x^5 + b*x^4 + c*x^3 + d*x^2 + e*x + f;
residues := map(d -> eval(p, x = d[1]) - d[2], data);
```

现在我们可以使用 LSSolve 命令，参数项是 residues。

```
sol := LSSolve(residues);
```

$$sol := [9.79941650829796096,\ [a = 0.00506817808955152,\ b = -0.148717633940652,\ c = 1.53722867786023,\ d = -7.04487080570615,\ e = 14.2425902266783,\ f = -7.12139344171469]]$$

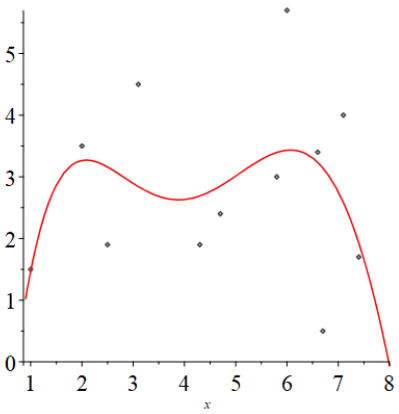
多项式变为：

```
poly := eval(p, sol[2]);
```

$$poly := 0.00506817808955152\ x^5 - 0.148717633940652\ x^4 + 1.53722867786023\ x^3 - 7.04487080570615\ x^2 + 14.2425902266783\ x - 7.12139344171469$$

对数据和曲线绘图：

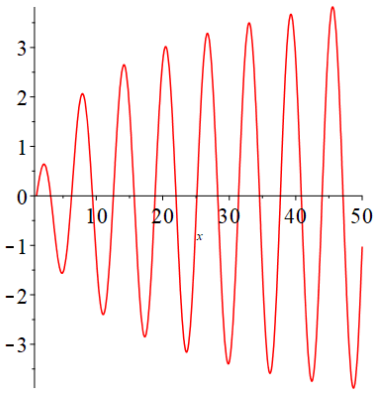
```
with(plots);
p1 := pointplot(data);
p2 := plot(poly, x = 0.90000 .. 8);
display(p1, p2);
```



3.5.4 全局优化

对于许多优化问题，简单的方法并不能足以发现最优解。它们仅能发现局部极值，通常是最靠近用户给定的搜索起始点的极值点。

```
interface(warnlevel = 0);
with(plots);
plot(ln(x)*sin(x), x = 1 .. 50);
```



您也许可以通过观察图形发现给定区域内的全局最小值。但如果您不能正确地求它的近似值，使用常规的优化算法并不能发现全局最小值。

```
Optimization[Minimize](ln(x)*sin(x), x = 1 .. 50);
```

[-3.39620, [x = 29.85500]]

根据 Minimize 命令的计算结果，最小值发生在 x = 30。但是你可以通过上面的图形发现它并不是全局最小值。

通过使用全局优化工具箱中有等效功能的命令，可以确保发现指定区间内的全局最小值。全局优化工具箱使用了全局搜索方法，不同于使用求导发现局部极值，它们系统地搜索整

个可行区域寻找一个全局极值。对于可能有许多局部极值的优化模型，全局搜索方法寻找一个全局极值。这些方法并不依赖于初始点。

尽管全局搜索不依赖于起始点，但它需要所有决策变量的有限边界，限定搜索空间。同时相比局部搜索方法，随着变量数的增加，全局搜索方法的计算工作量会急剧增加。这种情况通常并不明显，除非变量的数目达到上百个。

```
with(GlobalOptimization);
GlobalSolve(ln(x)*sin(x), x = 1 .. 50);
```

[-3.88562416790700960, [x = 48.6999273727307]]

为了查看使用了何种方法发现全局最小值，改变 infolevel 变量，然后重新执行命令。设置 infolevel = 3，命令显示输入模型的大小和类型，求解器运行模式和参数，以及详细的运行时间信息。

```
infolevel[GlobalOptimization] := 3;
GlobalSolve(ln(x)*sin(x), x = 1 .. 50);
```

通常，infolevel 设置为默认值 0。

```
infolevel[GlobalOptimization] := 0;
```

下面是一个例子，使用线性方法不能发现全局最小值，但全局求解器可以发现最优值。

考虑一个非线性方程组：

```
eq1 := sqrt(y^4 + x^2) + exp(-y^2 + x) + 5*sin(-4*x*y + 2*x) - 12*cos(x*y);
eq2 := 5*ln(x^2 + 1) + exp(-y + x) + 5*sin(6*x*y);
```

最小二乘误差函数有多个极值。

```
plot3d(eq1^2 + eq2^2, x = -2 .. 2, y = -1 .. 3, grid = [30, 30], lightmodel = light4, axes = boxed);
```

为了求最小二乘误差的全局最小值，定义要优化的目标函数和约束。

```
objf := eq1^2 + eq2^2;
cons := {eq1, eq2}
```

首先，使用 Maple 内置的 Optimization 函数包求优化解。由于方程组的复杂性，线性优化求解器并不能发现可行的解。

```
localsoln := Optimization[Minimize](objf, {eq1 = 0, eq2 = 0}, x = -1 .. 2, y = -2 .. 1);
```

Error, (in Optimization:-NLPSolve) no improved point could be found

然而，全局优化算法可以发现全局最小值。

```
sol := GlobalSolve(objf, {eq1 = 0, eq2 = 0}, x = -1 .. 2, y = -2 .. 1);
sol[1]; sol[2];
```

$$6.31089\ 10^{-30}$$

$$[x = -0.55897, y = -1.58235]$$

代入约束函数，结果显示最小二乘逼近是相当精确的解。也就是说，发生在该最小点上的最小二乘逼近的误差非常小。

```
eval(cons, sol[2]);
```

$$\{-0.00018, -0.00014\}$$

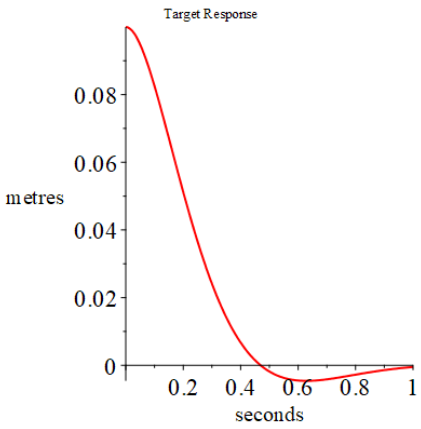
[例3.10] 汽车悬架系统的调校

考虑悬架系统设计中的问题，显示路面颠簸时的响应。问题变量是弹簧常数 k 和阻尼常数 b。每个车轮上车重给定 m，以及一个典型的颠簸的期望幅度，求 k 和 b 的值，创建与期望响应匹配的系统响应。

要最小化的目标函数是期望和实际响应之间的平方误差，是关于 k 和 b 在离散时间集上的函数。在通过求解系统的微分方程得到实际响应后，使用优化算法求 k 和 b 的值，最小化误差值。

解： 假设目标响应是一个衰减指数函数。当汽车碰到一个幅度为 0.1 米的颠簸时，其振荡幅度呈指数衰减。

```
restart;
Target := t -> 0.14000*exp((-1)*4.90000*t)*cos(5*t - 0.77540);
plot(Target(t), t = 0 .. 1, title = "Target Response", labels = ["seconds", "metres"], thickness = 2);
```



计算实际响应：

为了得到系统对颠簸的实际响应，求解微分方程组，初始条件是 x(0) = 0.1。

自然状态下质量-弹簧-阻尼系统的微分方程是：

```
System_Equation := m*diff(x(t), t, t) + b*diff(x(t), t) + k*x(t) = 0
```

$$m\left(\frac{d^2}{dt^2}x(t)\right)+b\left(\frac{d}{dt}x(t)\right)+k\,x(t)=0$$

求解微分方程，定义响应为 k, b, 和 t 的函数，质量为 m=450 kg。

```
m := 450;
```

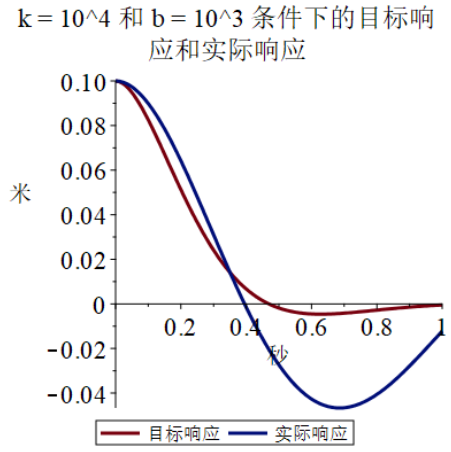
```
sol := dsolve({System_Equation, x(0) = 0.10000, D(x)(0) = 0});
```

系统的实际响应是关于 k, b, 和 t 的函数。

```
Actual := unapply(rhs(sol), k, b, t);
```

k=10^4 N/m 和 b=10^3 Ns/m 条件下的目标响应和实际响应。

```
plot([Target(t), Actual(10^4, 10^3, t)], t = 0 .. 1, thickness = 3, labels = ["秒", "米"], title = "k = 10^4 和 b = 10^3 条件下的目标响应和实际响应", legend = ["目标响应", "实际响应"])
```



显然 k 和 b 的值没有很好地匹配目标。提高匹配性的一个方法是代入 k 和 b 不同的组合，直到发现一个可接受的结果。下面的小节描述了一个更加系统的方法：使用优化。

测量目标和实际响应之间的误差

理想情况下，误差是目标和实际响应的平方差的积分。但是在这个应用中由于响应函数的复杂性，求积分值非常困难。作为近似，在一些关键的时间点上取函数样本，函数对这些时间点上的平方差相加。

在时间 t = 1/2, 1, 3/2, 2 上的输出样本：

```
time_sample := 1/2, 1, 3/2, 2;
```

目标函数是实际和目标响应在这 4 个时间点上的平方差的和。尽管误差函数仅有两个变量，但是可以看出非常复杂。

Error := add((Actual(k, b, time_sample[i]) - Target(time_sample[i]))^2, i = 1 .. 4);

画出误差函数在 k-b 平面上的图形。函数不仅仅是非凸的，而且在可能含有全局最小值的区域比较平坦。在计算前还不知道下面显示的图形区域是否含有期望的答案。

（注意：根据微分方程理论，通常取消响应函数和误差函数的虚部项。但是，Maple 的绘图函数在计算之前并不能检验。为了正确地绘图，对误差函数的虚部绘图，等同于函数绘图。）

plot3d(Re(Error), b = 100 .. 5000, k = 1000 .. 25000, axes = boxed, shading = zhue, transparency = 0.35000, style = patchnogrid, orientation = [75, 75]);

使用全局优化工具箱求误差函数的最小值。

with(GlobalOptimization);

[GetLastSolution, GlobalSolve, Interactive]

工具箱发现全局最小值。注意到误差值非常小，接近于 0，正如期望的一样。

sol := GlobalSolve(Re(Error), b = 100 .. 10000, k = 100 .. 40000);

$$sol := [1.67931\ 10^{-15}, [b = 4410.26174, k = 22055.21546]]$$

画出全局最小值领域内的误差函数。即使在这个小区域内，图形也不能明显地显示全局最小值的位置。

plot3d(Re(Error), b = 4300..4600, k = 21000..23000, axes = boxed, shading = zhue, transparency = 0.35000, style = patchnogrid, orientation = [75, 75]);

测试结果

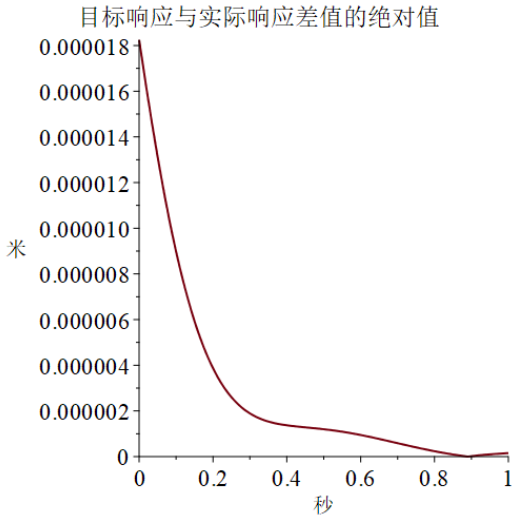
在同一图形上显示实际响应和目标响应，k 和 b 使用全局优化得到的值。它们可以很好地匹配。

assign(sol[2]);

plot([Target(t), Actual(k, b, t) + 0.05000, 0.05000], t = 0 .. 1, labels = ["秒", "米"], title = "目标与实际响应之间的差值)", thickness = 2, color = [red, green, black]);

从目标响应和实际响应之间差的图形上可以发现，即使是最大差值也可以忽略。

plot(abs(Target(t) - Actual(k, b, t)), t = 0 .. 1, title = "目标响应与实际响应差值的绝对值", labels = ["秒", "米"], thickness = 2);



[例3.11] 香水瓶子设计的优化

一家香水公司希望设计一个新的香水瓶子，要求容量固定的前提下降低运费。运费是与包装盒占有的体积成正比。因此设计任务是最小化占有空间，同时保持瓶子的容量和美观性。

瓶子的设计是椭圆体、平底。这里有 4 个问题变量：椭圆的 3 个离心率参数，椭圆体被切除形成平底的高度位置。

瓶子模型

```
restart;
interface(warnlevel = 0);
with(plots); with(plottools); with(GlobalOptimization);
```

瓶子形状的模型是一个包含参数 a, b, c 的椭圆体，中心在原点。上面部分是椭圆体，但椭圆体地段是平面 z = -h，这里 h 是模型的第 4 个参数。长度单位是 cm，体积是 mL。

```
bottleShape := x^2/a^2 + y^2/b^2 + z^2/c^2 = 1;
```

$$bottleShape := \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$$

对应的图形是一个平底瓶。目的是使用优化方法改进这个形状。

```
implicitplot3d(eval(bottleShape, {a = 2, b = 1.5, c = 1}), x = -2 .. 2, y = -1.5 .. 1.5, z = -6/10 .. 1,
scaling = constrained, orientation = [75, 80], style = patchnogrid, lightmodel = light1);
```

目标函数：包装体积

包含瓶子的盒子是一个平行六面体，宽度是 2a，长度是 2b，高度是 h + c + 1，这里额外的长度是为喷嘴准备的空间。目标是最小化$4\,a\,b\,(c+h+1)$。

```
packageVolume := 4*a*b*(h + c + 1)
```

约束条件 1：瓶子体积

瓶子必须能够至少装 40 mL 的香水。瓶子的体积是椭圆体的横截面，是关于 z 的函数，在区间 z = -h .. c 上的的积分。

```
z_crossSection := Pi*a*b*(1 - z^2/c^2)
perfumeVolume := Int(z_crossSection, z = -h .. c);
perfumeVolume := value(perfumeVolume);
```

通过求它在 h = c 上的值检查公式。变为一个求椭圆体的公式，符合期望。

```
eval(perfumeVolume, {h = c})
constraint1 := 40 <= perfumeVolume
```

约束条件 2：底的宽度

底必须至少有 2 cm 的直径，以保持瓶子的稳定性。作为 h 的函数，在平面 z = -h 上的横截面最小直径是$2\sqrt{b^2\left(1-\frac{h^2}{c^2}\right)}$。约束 h 要求直径至少是 2 cm。

```
hSquareMax := solve(4*b^2*(1 - h^2/c^2) = 2^2, h^2)
```

得到下面的约束：

```
constraint2 := h^2 <= hSquareMax
```

约束条件 3：美学比例

客户通常喜欢圆滑瓶。尽管是最经济的设计，但公司不会销售球形香水瓶。限制瓶子的比例满足$2\,b \le a$。

```
constraint3 := 2*b <= a
```

优化

```
constraints := {constraint1, constraint2, constraint3}
```

$$constraints := \left\{40 \le -\frac{1}{3}\,\frac{\pi\,a\,b\left(c^3+h^3\right)}{c^2}+\pi\,a\,b\,(h+c),\ h^2 \le \frac{\left(-1+b^2\right)c^2}{b^2},\ 2\,b \le a\right\}$$

使用全局优化工具箱发现最优解。得到包装体积的最小值 77.69 mL，显然大于瓶子的体积 40 mL，满足期望。

```
sol := GlobalSolve(packageVolume, constraints, a = 1 .. 10, b = 1 .. 10, c = 1 .. 10, h = 1 .. 10)
```

sol := [77.68893, [a = 2.15296, b = 1.07648, c = 5.38623, h = 1.99399]]

检查得到的结果是否在可接受的精度内满足所有的约束条件。为了更精确，使用 GlobalSolve 调用格式中的 feasibilitytolerance 和 penaltymultiplier 参数项。

```
subs(sol[2], constraints);
evalf[6](%);
```

{2.15296 <= 2.15296, 3.97599 <= 3.97599, 40.00000 <= 40.00000}

优化后的椭圆体：

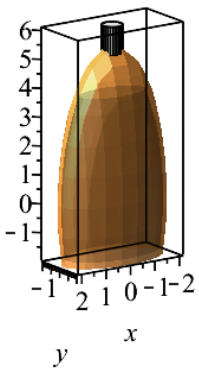
```
bottleShape := eval(bottleShape, sol[2]);
```

$$bottleShape := 0.21574\,x^2 + 0.86296\,y^2 + 0.03447\,z^2 = 1$$

```
A := eval(a, sol[2]);    B := eval(b, sol[2]);    C := eval(c, sol[2]);    H := eval(h, sol[2]);
```

优化后的香水瓶的轮廓图：

```
glass := implicitplot3d(bottleShape, x = -A .. A, y = -B .. B, z = -H .. C, grid = [8, 8, 8], style = patchnogrid, lightmodel = light4, transparency = 0.10000, color = gold);
liquid := implicitplot3d(lhs(bottleShape) = 0.85000, x = -A .. A, y = -B .. B, z = -0.90000*H .. 0.70000*C, style = patchnogrid, lightmodel = light4, transparency = 0.80000, color = black);
cap := cylinder([0, 0, 0.95000*C], A/6, color = grey);
display(glass, liquid, cap, scaling = constrained, axes = box);
```



3.6 概率论与数理统计

Statistics 函数包提供工具完成统计计算和数据分析，支持广泛的统计计算功能，包括定量分析和图形数据分析、模拟、曲线拟合。

除了标准的数据分析工具，Statistics 函数包提供大量的符号和数值工具计算随机变量。该

函数包提供超过 35 个概率分布，并可扩展新的分布函数。

3.6.1 概率分布和随机变量

Statistics 函数包支持：

- 连续分布，通过概率密度函数定义实线。Maple 支持许多连续分布，包括 Normal, Student-t, Laplace, logistic 分布。
- 离散分布，仅在离散点上有非零概率。离散分布通过 probability function 定义。Maple 支持大量的离散分布，包括 Bernoulli, geometric 和 Poisson 分布。

你可以通过在 RandomVariable 命令中调用一个分布，定义随机变量。

```
with(Statistics);
X := RandomVariable(Poisson(lambda));
```

X 的概率分布函数。

```
PDF(X, t)
```

$$\sum_{k=0}^{\infty} \frac{\lambda^k \, e^{-\lambda} \, \mathrm{Dirac}(-t+k)}{k!}$$

添加自定义分布

想要添加一个新的分布，在调用 Distribution 命令时定义一个概率分布。

$$U := Distribution\left(PDF = \left(t \to \begin{cases} 0 & t < 0 \\ \frac{1}{3} & t < 3 \\ 0 & otherwise \end{cases}\right)\right):$$

或者

```
U := Distribution(PDF = (t -> piecewise(t < 0, 0, t < 3, 1/3, 0)))
```

为了创建一个一维形式的分段连续函数，使用 piecewise 命令，例如 t -> piecewise(t < 0, 0, t < 3, 1/3, 0)。

使用该分布定义一个新的随机变量。

```
Z := RandomVariable(U);
PDF(Z, t);
```

计算随机变量的均值。

```
Mean(Z);
```

3.6.2 统计计算

除了支持基础计算，例如均值、中位数、标准差、百分位数，Statistics 函数包还包含大量的计算命令，例如四分位距和失效率。

[例3.12] 四分位距

计算 Rayleigh 分布（Scale 参数为 3）的四分位距的平均绝对范围。

```
with(Statistics);
InterquartileRange(Rayleigh(3));
```

$$\sqrt{36}\,\sqrt{\ln(2)} - \sqrt{-18\ln\left(\frac{3}{4}\right)}$$

计算数值结果：

指定 'numeric' 参数项

```
InterquartileRange(Rayleigh(3), 'numeric')
```

2.71974

[例3.13] 失效率

计算 Cauchy 分布的失效率。

```
HazardRate(Cauchy(a, b), t)
```

$$\frac{1}{\pi b\left(1+\frac{(t-a)^2}{b^2}\right)\left(\frac{1}{2}-\frac{\arctan\left(\frac{t-a}{b}\right)}{\pi}\right)}$$

可以指定点 t 的值。

```
HazardRate(Cauchy(a, b), 1/2)
```

也可以要求 Maple 返回数值结果。

```
HazardRate(Cauchy(10, 1), 1/2, 'numeric')
```

0.00361

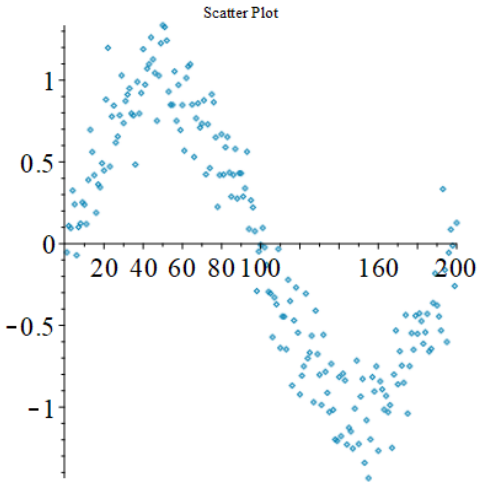
3.6.3 统计图形

可以使用 Statistics 函数包中的可视化命令生成统计图形，包括：

- 条形图
- 频率图
- 直方图
- 饼图
- 散点图

例如，创建一个散点图，对象是基于$\sin\left(\frac{2\pi x}{200}\right)$的分布点，同时附加一个由正态分布样本数据点定义的较小值。

```
N := 200;
U := Sample(Normal(0, 1), N);
X := <seq(x, x = 1 .. N)>;
Y := <seq(sin(2*Pi*x/N) + U[x]/5, x = 1 .. N)>;
ScatterPlot(X, Y, 'title' = "Scatter Plot");
```

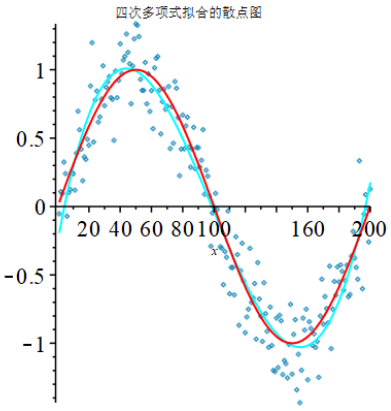


拟合数据点，加入 fit 方程参数项。

使用 plots[display]命令，创建一个图形，包括：

- 数据点的散点图
- 使用一个四次多项式拟合数据点：$f(x) = a x^4 + b x^3 + c x^2 + d x + e$
- 函数 $\sin\left(\frac{2\pi x}{N}\right)$

```
P := ScatterPlot(X, Y, fit = [a*x^4 + b*x^3 + c*x^2 + d*x + e, x], thickness = 2)
Q := plot(sin(2*Pi*x/N), x = 1 .. N, thickness = 2, color = red)
plots[display](P, Q, 'title' = "四次多项式拟合的散点图")
```



3.4.5 积分

Maple 可以完成符号和数值积分。

计算一个表达式的不定积分：

1. 从表达式面板，点击不定积分项 $\int f\,dx$ 。

2. 定义被积函数和积分变量，然后求值。

例如，求$x\sin(a\,x)$ 关于 x 的积分：

[> $\int x\sin(a\,x)\,dx$

$$\frac{\sin(a\,x)-x\cos(a\,x)\,a}{a^2}$$

再次提醒，你可以使用符号补全功能输入符号，包括$\int$和 d。

输入符号名（或者部分名称），例如 int 或者 d，然后按命令补全快捷键 ESC。也可以使用关联菜单计算不定积分。

计算表达式的定积分：

1. 从表达式面板，点击定积分项$\int_a^b f\,dx$。

2. 定义积分区间的端点、被积函数表达式、积分变量，然后求值。

例如，求$e^{-a\,t}\ln(t)$ 在区间 $(0,\ \infty)$上的积分：

[> $\int_0^{\infty} e^{-a\,t}\ln(t)\,dt$

$$\lim_{t\to\infty}\left(-\frac{e^{-a\,t}\ln(t)+\mathrm{Ei}(1,a\,t)+\gamma+\ln(a)}{a}\right)$$

Maple 将参数 a 解析为复数。可以使用 assuming 命令假设 a 是一个正的实数。

[> $\int_0^{\infty} e^{-at}\ln(t)\,dt \text{ assuming } a>0$

$$-\frac{\gamma+\ln(a)}{a}$$

想要计算迭代积分、线积分和面积分，使用 Multivariate 和 Vector Calculus 文件夹下的任务模板 (工具→任务→浏览) 。

int 命令

$\int f\,dx$ 和 $\int_a^b f\,dx$ 使用 int 命令。如果要直接使用 int 命令，需要定义如下的参数项。

```
[> int(x*sin(a*x), x);
```

$$\frac{\sin(a\,x)-x\cos(a\,x)\,a}{a^2}$$

对于定积分，设置积分变量等同于定义积分区间。

```
[> int(x*sin(a*x), x = 0 .. Pi/a);
```

$$\frac{\pi}{a^2}$$

数值积分

使用 evalf(Int(arguments)) 调用格式。

重要：使用惰性命令 Int，而不是 int。

除了使用 int 命令可用的参数项外，你可以使用其他一些参数，例如 method，指定数值积分的方法。

```
[> evalf(Int(1/GAMMA(x), x = 0 .. 2, 'method' = _Dexp));
```

提示: 如果想在 2-D 数学格式下输入下划线字符，输入 _.

关于数值积分的更多信息，包括迭代积分和设置积分算法，请参阅帮助系统的 evalf/Int 帮助页。

3.6.4 Student Statistics 学生学习函数包

Student [Statistics]函数包帮助教师课堂辅助教学，以及大学生学习统计课程中的概念。Student [Statistics] 函数包是 Statistics 的一个子函数库，可以完成创建和探索随机分布，执行假设检验等任务。

3.7 内置工程及科学常数数据库

热力学、热物理数据库。

```
[> with(ThermophysicalData);
[> Property(enthalpy, water, temperature = 390*Unit(K), pressure = Unit(atm));
```

$$2.710124643\ 10^{6}\ \frac{\mathrm{J}}{\mathrm{kg}}$$

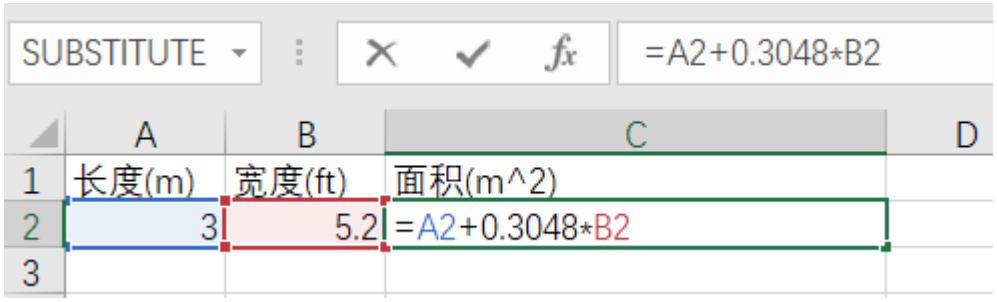
科学常数库。

```
[> with(ScientificConstants);
[> Na := Element('Na', atomicweight);
[> GetValue(Na)*GetUnit(Na);
```

$$3.817543727\ 10^{-26}\ \mathrm{kg}$$

3.8 含单位和公差的科学计算

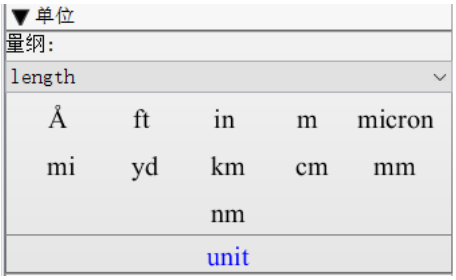
不使用单位计算可能面临风险。使用常规表格计算，往往会出现下图所示，在计算过程中添加单位转换系数，存在隐患。



3.8.1 输入单位

在 Maple 中有三种方式，可以在数值后添加单位：

1.使用快捷键 Ctrl+Shift+U	$area1 := 5.5\ [\![\mathrm{m}^2]\!]$
2. Unit 函数	[> area2 := 4.5*Unit(cm^2)

3. 使用窗口左侧的单位面板	

以下具体显示如何使用快捷键 Ctrl+Shift+U 来完成单位操作。

首先输入 x:=3.5	> $x := 3.5$
同时按住 Ctrl+Shift+U	> $x := 3.5[\![\text{unit}]\!]$
输入单位 m	> $x := 3.5[\![\text{m}]\!]$
按键盘上向右箭头	> $x := 3.5\text{m}$
按回车键	> $x := 3.5\text{m}:$

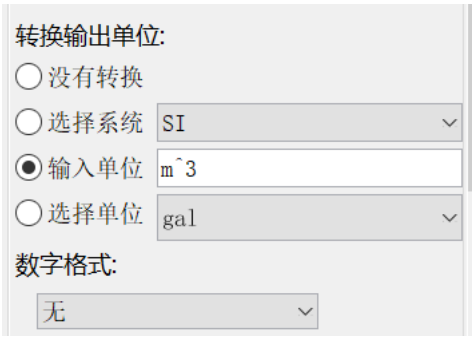
改变结果的单位显示：

可以通过右侧的关联菜单，对计算结果的单位进行转换，以及数字格式设置。

> $area := 3.5\ \text{m} \cdot 5.4\ [\![\text{ft}^2]\!]$

$$area := 18.9000000000\ \text{m ft}^2$$

鼠标点击输出结果，然后从关联菜单选择或者直接输入要转换的单位。



3.8.2 国际规范

Maple 中的单位名称，格式是按照 NIST 的标准来定义的。

$w := 4.5\ [\![\text{N m}^{-1}]\!]:$

NIST - 美国国家标准与技术研究院（National Institute of Standards and Technology，NIST）直属美国商务部。

3.8.3 单位计算

[> with(Units)

Automatically loading the Units[Simple] subpackage

[> force := 4.5*Unit(N);

[> area := 3.4*Unit(cm^2);

$$stress := \frac{force}{area} = 13235.29412\ \text{Pa}$$

以上是按 CTRL=组合键后再同一行显示结果。有命令行输入符[>时，按回车键将另起一行显示结果。

计算后自动转换单位。同一量纲下的不同单位，在经过计算后会自动转换。

$$3.5\text{ft} + 2.5[\![\text{m}]\!] = 3.57\ \text{m}$$

$$force := 3.4\text{N}:$$
$$area := 5\ \text{m}^2:$$
$$stress := \frac{force}{area} = 0.6800000000\ \text{Pa}$$

不允许不同量纲的单位计算。举例：

$$1.2\ \text{m}\ \text{s}^{-1} + 2.3\ [\![\text{ft}]\!]$$

Error, (in Units:-Simple:-+) the following expressions imply incompatible dimensions: {1.2*Units:-Unit(m/s)+2.3*Units:-Unit(ft)}

可以自动辅助检查输入公式和单位。

Maple 支持多种单位系统。

举例，下式使用了多种表示功率的单位，Maple 可以轻松计算：

$$1\text{kW} + 2.5\text{hp} + 50\cdot 10^{9}[\![\text{erg}\ \text{s}^{-1}]\!] = 447.5315240\ \frac{\text{Btu}}{\text{min}}$$

单位转换。

[> 3.0*Unit(kg*m/(s))

$$3.0000000000\ \frac{\text{kg m}}{\text{s}}$$

[> convert(3.0*Units:-Unit(kg*m/s), units, lb*ft/(h))

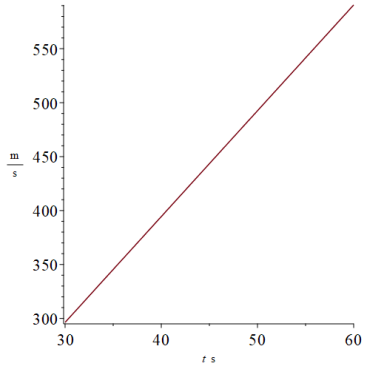
$$78116.5495900000\ \frac{\text{lb ft}}{\text{h}}$$

3.8.4 绘图中使用单位

```
[> u := 2*Unit((m)/(s));
[> a := 9.8100000000*Unit(('m')/'s'^2);
```

$$a := 9.81 \frac{\mathrm{m}}{\mathrm{s}^2}$$

```
[> v := t -> u + a*t;      #定义函数 v(t)
[> plot(v(t), t = 30*Unit(s) .. Unit(min));      #绘制函数 v(t)的图形
```



3.8.5 在方程、积分、优化等计算时使用单位

例3.1 计算燃烧温度

液态丁烷与 100%理想空气在 298.15K 初始温度下开始燃烧，燃烧反应公式为：

$$\#\ C_4H_{10}(l) + 6.5\,O_2(g) + 24.44\,N_2(g) \rightarrow 4\,CO_2(g) + 5\,H_2O(g) + 24.44\,N_2(g)$$

接下来，我们要计算它的绝热火焰燃烧温度。

```
with(ThermophysicalData);
```

[*Chemicals, CoolProp, PHTChart, Property, PsychrometricChart, TemperatureEntropyChart*]

查询丁烷的生成热：

```
h_f_C4H10 := Chemicals:-Property("HeatOfFormation", "C4H10(l),n-buta", useunits);
```

$$-150.6640000 \frac{\mathrm{kJ}}{\mathrm{mol}}$$

查询温度“T”时，各燃烧产物的焓值：

```
h_N2 := Chemicals:-Property("Hmolar", "N2(g)", "temperature" = T);
h_O2 := Chemicals:-Property("Hmolar", "O2(g)", "temperature" = T);
h_H2O := Chemicals:-Property("Hmolar", "H2O(g)", "temperature" = T);
```

h_CO2 := Chemicals:-Property("Hmolar", "CO2(g)", "temperature" = T);

丁烷的焓值：

H_reactants := Unit('mol')*h_f_C4H10

$$-150.6640000\ \text{kJ}$$

燃烧产物的总焓值：

H_products := 4*Unit(mol)*h_CO2 + 5*Unit(mol)*h_H2O + 24.44*Unit(mol)*h_N2;

H_products := (4*Chemicals:-Property("Hmolar", "CO2(g)", "temperature" = T) + 5*Chemicals:-Property("Hmolar", "H2O(g)", "temperature" = T) + 24.44*Chemicals:-Property("Hmolar", "N2(g)", "temperature" = T))*Unit(mol)

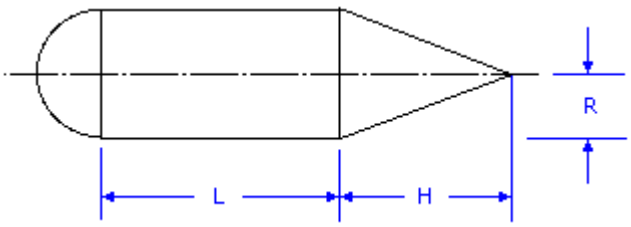
联立方程，绝热环境下，燃烧前后焓值相等。

fsolve(H_reactants = H_products, T = 2000*Unit(K))

$$2379.853026\ \text{K}$$

例3.2 燃油舱设计

设计优化一款燃油舱如下图，左边是半球形，中间是圆柱形，右边是圆锥形。



在保持容积为 3*m^3 计算 L,H,R 的值为多少时，油箱的表面积最小？

restart:

表面积方程为：

[> obj := 4*1/2*Pi*R^2 + 2*Pi*R*L + Pi*R*sqrt(H^2 + R^2);

边界条件：容积保持为 3 立方米

[> cons1 := 1/2*4/3*Pi*R^3 + Pi*R^2*L + 1/3*Pi*R^2*H = 3*Unit('m'^3)

所有尺寸都为正数：

[> cons2 := 0 <= R, 0 <= L, 0 <= H

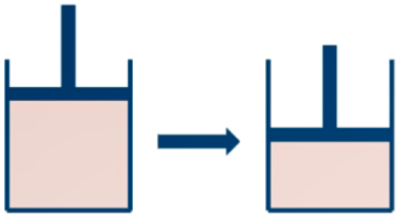
优化求解各尺寸的值：

dimensions := Optimization:-Minimize(obj, {cons1, cons2}, initialpoint = {H = Unit(m), L = Unit(m), R = Unit(m)})

$$[10.25\ \mathrm{m}^2,\ [H = 0.79\ \mathrm{m},\ L = 0.39\ \mathrm{m},\ R = 0.88\ \mathrm{m}]]$$

例3.3 计算压缩气体所做的功

将甲烷气体在 350K 温度下，等温压缩，从$1.0\ m^3kg^{-1}$压缩到$0.5\ m^3kg^{-1}$计算此过程所需的功。



```
[> with(Units);
```

Automatically loading the Units[Simple] subpackage

首先建立 P 与 V 的查表关系。

```
[> P := ThermophysicalData:-Property("pressure", "methane", "temperature" = 350*Unit(K), "density" = 1/V);
```

输入一个比体积，测试查遍关系是否建立完好：

```
[> eval(P, V = 0.1000000000*Unit('m'^3/('kg')))
```

$$1.785330702\ \mathrm{MPa}$$

带入数值积分，求解压缩时所做的功。

$$\int_{1.0\mathrm{m}^3\,\mathrm{kg}^{-1}}^{0.5\left[\!\left[\mathrm{m}^3\,\mathrm{kg}^{-1}\right]\!\right]} P\,\mathrm{d}V$$

$$-125.43\ \frac{\mathrm{kJ}}{\mathrm{kg}}$$

3.9 公差计算

变量可以带有单位和公差，贯穿整个计算过程。

例如，假设水压计读数为 0.08 m，测量误差为±0.002 m。 因此，可能的压力范围是

```
restart;
with(Tolerances);
h := (0.0800000000 &+- 0.002)*Unit(m);
```

$$h := 0.08 \pm 0.002\text{m}$$

g := 9.8100000000*Unit(('m')/'s'^2);

$$g := 9.8100000000 \frac{\text{m}}{\text{s}^2}$$

rho__water := ThermophysicalData:-Property(density, water, pressure = 100*Unit(kPa), temperature = 25*Unit(degC))

$$\rho_{water} := 997.0470390 \frac{\text{kg}}{\text{m}^3}$$

Delta := rho__water*g*h;

$$\Delta := 782.4825162000 \pm 19.5620630000\ \text{Pa}$$

第四章 图形和动画

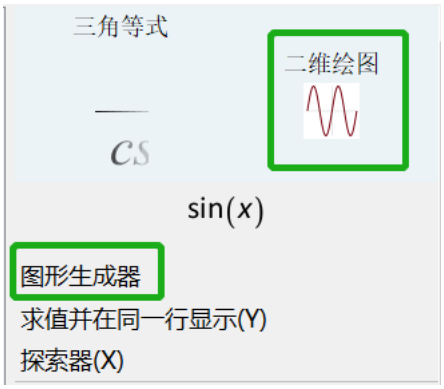
Maple 提供各种各样的可视化工具。您可使用关联菜单创建 2-D 和 3-D 图形以及动画。Maple 同样提供了大量的命令和编程工具实现图形的创建和客户化设置。这些命令可以交互式使用或者包含在 Maple 程序脚本中，创建自定义的特定图形和高级应用。

Maple 可以生成各种形式的图形，帮助您可视化问题并进一步理解概念。

- Maple 提供 2D、3D 图形和动画工具，超过 200 种图形类型，包括隐式、等高图、复数、极、向量场、密度、保角变换、常微分方程、偏微分方程、统计图、时域和频域响应图、根轨迹图、和根轮廓图。
- 可以识别各种坐标系。
- 所有的绘图区域都处于活动状态（非静态），因此您可以在绘图区域之间来回拖动表达式。
- Maple 提供了许多绘图选项，例如颜色、线型、点样式、轴样式、网格线、图例、刻度标签、阴影选项、照明模式、光泽度、透明度、阴影、投影等，您可以完全控制图形。

4.1 使用关联菜单绘图

在“数学”状态下输入表达式，从右侧的菜单选择“图形生成器”。该方式适合常用的图形类型和选项控制，请自行点击体验。



4.2 二维图形和三维图形命令

Maple 中绘制二维和三维图形的命令是 plot 和 plot3d 命令。

plot(plotexpression, x=a..b, ...)

plot3d(plotexpression, x=a..b, y=a..b, ...)

- plotexpression - 要绘图的公式
- x=a..b – 轴名称和水平轴范围
- y=a..b – 轴名称和垂直轴范围

注意：绘图命令的默认范围是-10 到 10，如果公式含有三角函数，那么默认范围是-2*Pi 到 2*Pi.

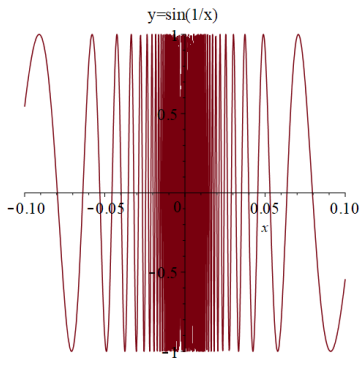
3.2.1 单变量表达式绘图

plot 命令的第一个参数项是表达式，第二个参数项是独立变量的范围。

```
plot(x^2, x=0..5);
```

在同一幅图上绘制两个表达式的图形。

```
plot(sin(1/x), x = -0.1000000000 .. 0.1000000000, title = "y=sin(1/x)", axes = normal) #其中 title 和 axes 是可选参数项
```



3.2.2 函数绘图

```
f:=x->x^2:  #定义函数 f(x)
plot(f(x),x=0..5);  #绘图
```

3.2.3 两个变量函数的二维图形

对包含两个变量的函数绘图，其中一个变量赋值为常数。

```
f:=(x,y)->x^2-2*y: #定义函数 f(x,y)
plot([f(x,1),f(x,5)],x=0..5);     #绘制 f(x,1)和 f(x,5)的图形
```

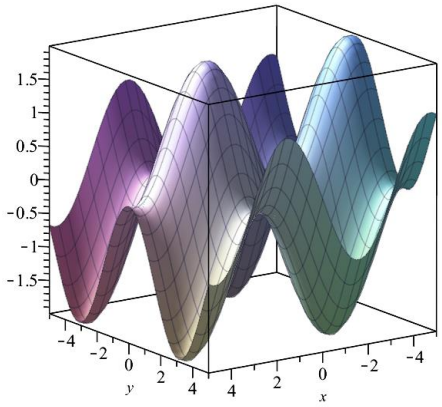
3.2.4 两个变量函数的三维图

包含两个变量的函数的三维图形。

```
f:=(x,y)->sin(x)+cos(y);
```

$$f := (x, y) \mapsto \sin(x) + \cos(y)$$

```
plot3d(f(x,y),x=-5..5,y=-5..5);
```



4.3 plots 函数包

plots 函数包提供许多用于绘图的命令，包括：animate, contourplot, densityplot, fieldplot, odeplot, matrixplot, spacecurve, textplot, tubeplot 等。

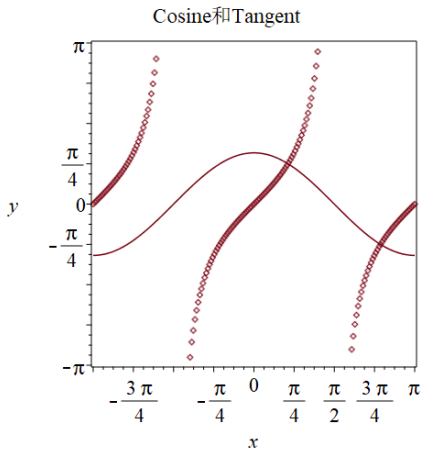
```
with(plots):   #加载 plots 函数包
```

3.3.1 多个图形和动画的合并显示

合并显示两个表达式的二维图形。

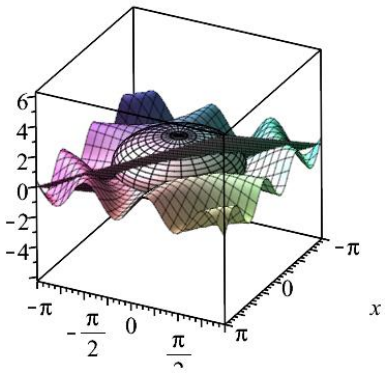
```
F := plot(cos(x), x = -Pi .. Pi, y = -Pi .. Pi, style = line);
G := plot(tan(x), x = -Pi .. Pi, y = -Pi .. Pi, style = point);
```

```
display({F, G}, axes = boxed, scaling = constrained, title = "Cosine 和 Tangent");
```



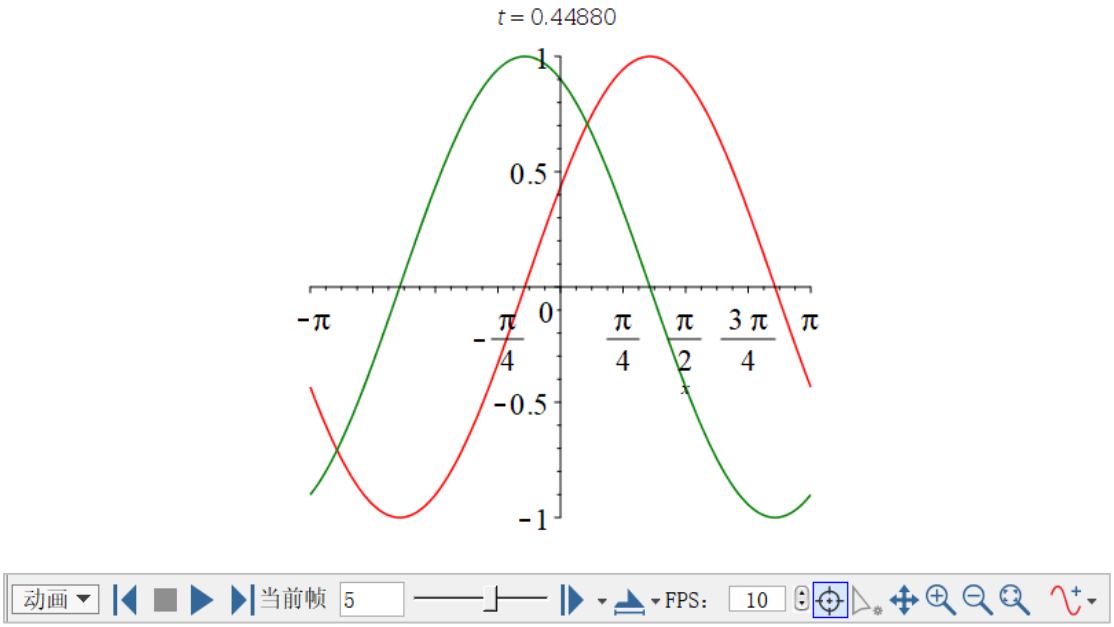
合并显示两个表达式的三维图形。

```
F:=plot3d(sin(x*y), x=-Pi..Pi, y=-Pi..Pi):
G:=plot3d(x + y, x=-Pi..Pi, y=-Pi..Pi):
H:=plot3d([2*sin(t)*cos(s), 2*cos(t)*cos(s), 2*sin(s)], s=0..Pi, t=-Pi..Pi):
display({F, G, H});
```



合并显示多个动画。

```
P := animate(plot, [sin(x+t), x=-Pi..Pi, color="Red"], t=-Pi..Pi):
Q := animate(plot, [cos(x+t), x=-Pi..Pi, color="Green"], t=-Pi..Pi):
display([P, Q]);
```



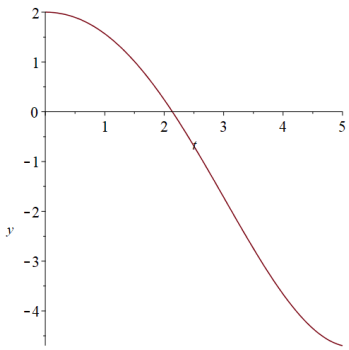
3.2.2 微分方程解的绘图

绘制微分方程解的图形：

```
de:=3*diff(y(t),t,t) = diff(y(t),t) - 1.2*y(t), D(y)(0)=0,y(0)=2;
```

$$de := 3\left(\frac{d^2}{dt^2}y(t)\right) = \frac{d}{dt}y(t) - 1.2\,y(t), D(y)(0) = 0, y(0) = 2$$

```
res:=dsolve({de},{y(t)},numeric);
plots:-odeplot(res,t=0..5);
```



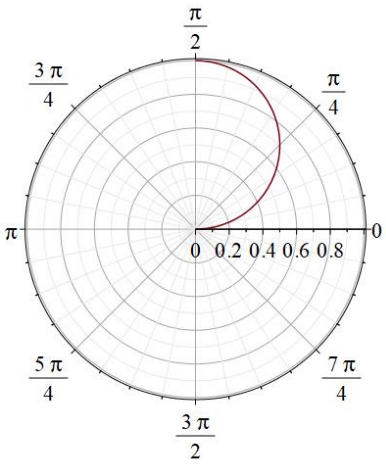
3.2.3 对数作图

对数作图主要有三种情形：logplot(线性-对数)、loglogplot(对数-对数)、semilogplot (对数-线性)。

```
with(plots):
logplot(x^2 - x + 4, x = 1 .. 12);
loglogplot(x^2 - x + 4, x = 1 .. 12);
semilogplot(x^2 - x + 4, x = 1 .. 12);
loglogplot([cos(2*t)^2 + 3, sin(t^2)^2 + 1, t = 0 .. 3]);
```

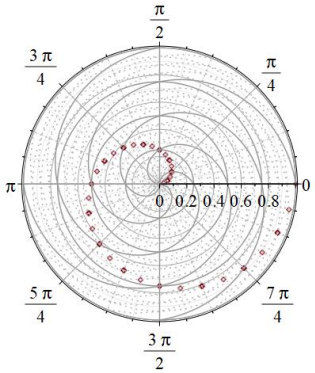
3.2.3 极坐标图

```
plots:-polarplot(sin(theta),theta=0..Pi/2);
```



3.3.4 隐函数的极坐标图形

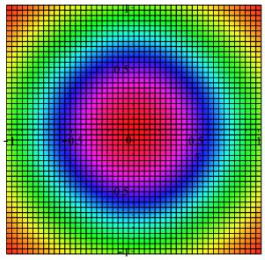
```
with(plots):
implicitplot(2*Pi*r - t = 0, r = 0 .. 1, t = 0 .. 2*Pi, coords = polar, axes = boxed, outlines, grid = [5, 5], gridrefine = 2, style = point, axiscoordinates = polar);
```



密度图和空间曲线

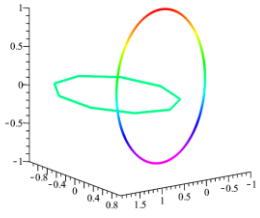
密度图：

```
> densityplot(exp(-x^2 - y^2), x = -1 .. 1, y = -1 .. 1, colorstyle = HUE);
```



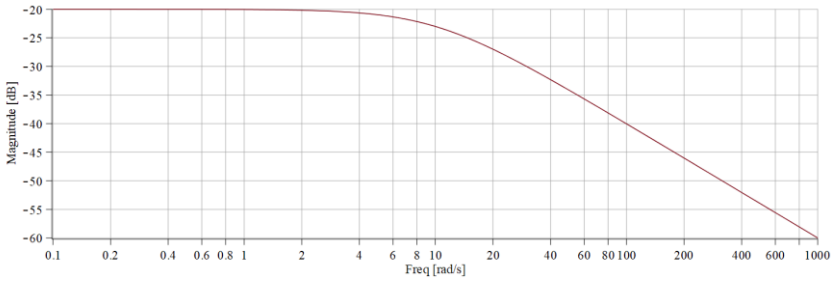
3D 空间曲线图：

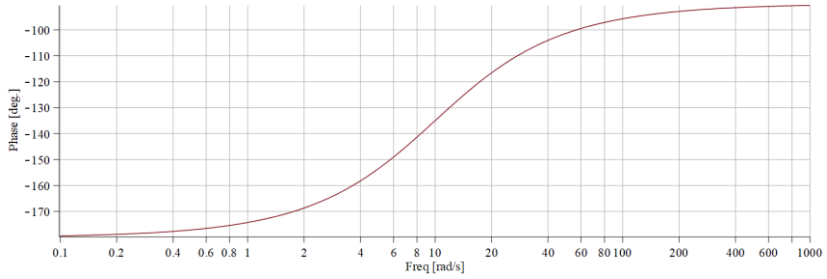
```
> spacecurve({[cos(t) + 1, sin(t), 0, numpoints = 10], [sin(t), 0, cos(t), t = 0 .. 2*Pi]}, t = -Pi .. Pi,
axes = framed, thickness = 5, shading = zhue);
```



3.3.5 波特图

```
with( DynamicSystems ):
sys := TransferFunction( 1/(s-10) ):
BodePlot(sys);
```



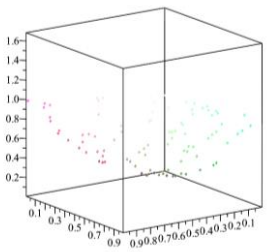


3.3.6 数据表的绘图

```
data:=LinearAlgebra:-RandomMatrix(5);
plots:-matrixplot(data);
```

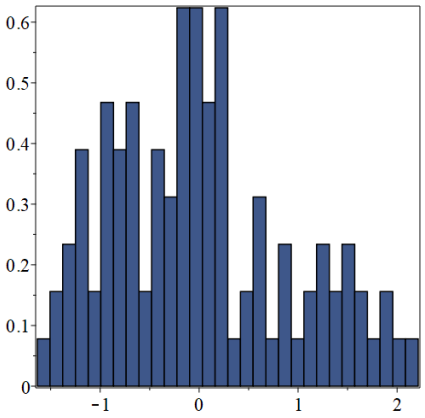
3.3.7 点云图

```
X:=LinearAlgebra:-RandomVector(100,generator=0.0..1.0);
Y:=LinearAlgebra:-RandomVector(100,generator=0.0..1.0);
Z:=Vector(100,i->X[i]^2+Y[i]^2);
points:=[seq([X[i],Y[i],Z[i]],i=1..100)]:
plots:-pointplot3d(points);
```



3.2.8 柱状图

```
N := Statistics:-RandomVariable(Normal(0, 1));   #加载统计 Statistics 函数包中的命令
A := Statistics:-Sample(N, 100);
Statistics:-Histogram(A);
```



3.2.9 双 y 轴图形

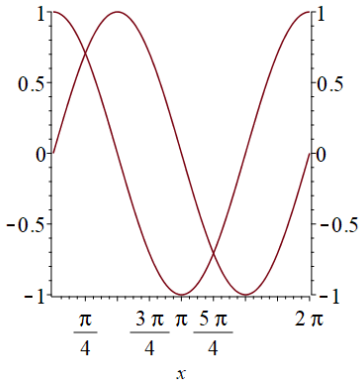
命令和调用格式

```
dualaxisplot(expr1, expr2, xrange, opts)

dualaxisplot(p1, p2, mopts)

with(plots):

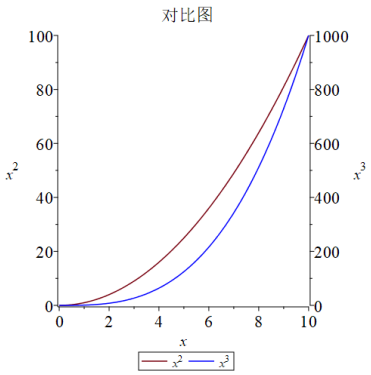
dualaxisplot(sin(x), cos(x), x=0..2*Pi);
```




```
plot1 := plot(x^2, x=0..10, labels=[x, x^2], legend=x^2):

plot2 := plot(x^3, x=0..10, color=blue, labels=[x, x^3], legend=x^3):

dualaxisplot(plot1, plot2, title="对比图");   #左右两个 y 轴的刻度范围不同
```



4.4 提取图形的数据并保存为 Excel 文件

getdata 命令可从 2D 或 3D 图形结构中获取图形的数据点，然后可将矩阵或数组形式的数据点输出为 excel 文件。

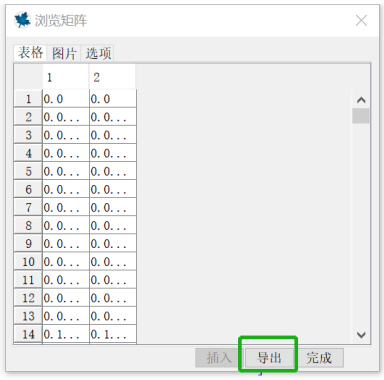
```
[>with(plottools):    #加载函数包
[>p1 := plot(sin(x), x=0..Pi/2):      #2D 图形
[>plot1 := plot3d(x^2 + y^2, x = -1 .. 1, y = -1 .. 1):      #3D 图形
[>getdata(p1);       #获取图形对象 p1 的绘图点数据
[>getdata(plot1):     #获取图形对象 p1 的绘图点数据
```

```
"curve", [0. ..1.5707963236, 0. ..1.0000000000],
```

0.	0.
0.0082586266	0.0082585328
0.0154444135	0.0154437995
0.0235255824	0.0235234124
0.0316603185	0.0316550296
0.0397563917	0.0397459195
0.0472624879	0.0472448944
0.0550346157	0.0550068383
0.0630725614	0.0630307510
0.0710847293	0.0710248788
⋮	⋮

200 × 2 Matrix

双击输出的矩阵，弹出“浏览矩阵”窗口，点击“导出”按钮，输出数据文件。



也可以用命令输出数据文件。

```
[>with(plottools):
[>data1 := getdata(p1)[3];
```

将图形数据导出为 Excel 文件。

```
[>with(ExcelTools);
[>Export(data1, "D:\\data.xls");
```

将图形数据导出为 csv 件。

```
[>ExportMatrix("D:\\data2.csv", data1, delimiter = ",");
```

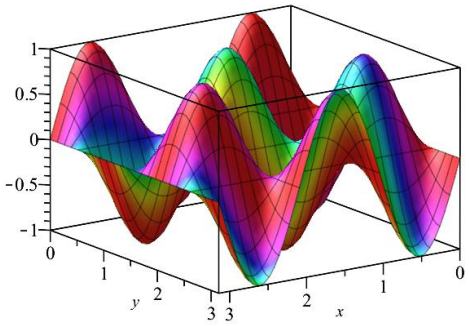
4.5 动画

Maple 的动画命令是 plots 函数库的 animate 和 animate3d。其命令格式分别如下：

```
[>animate (F, x, t);
[>animate3d (F,x,y,t);
```

其中，F—要绘图的函数，x, y—横轴、纵轴的变化范围，t—结构参数的变化范围

```
[>with(plots)
[>animate(cos(3*t)*sin(3*x), x = 0 .. 2*Pi, t = 0 .. 2*Pi, frames = 100, color = red, scaling = constrained);
```



执行上述语句后，用鼠标点击图形，Maple 窗口上侧的工具栏将显示下图所示动画控制工具条，我们可以通过上面的按钮控制动画的播放。



第五章 数据处理

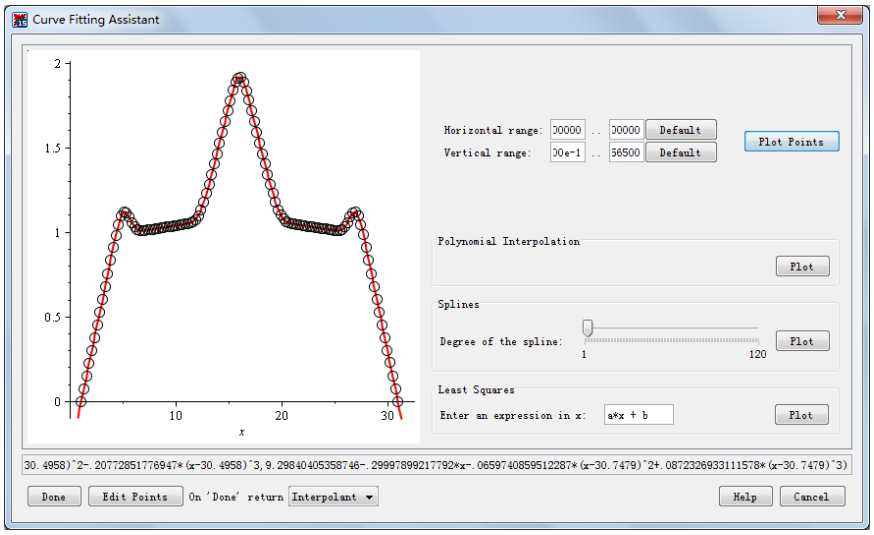
很多情况下，我们难以甚至不能生成工程系统的数学模型，但可能可以从实际系统生成的实验数据中完成系统辨识。Maple 提供了许多数据处理工具，例如线性和非线性模型拟合函数，定义数学模型的近似形式（通常知道形式，但实际的参数值未知），然后使用 Maple 的优化工具生成最佳的参数值，让模型的行为接近于实际系统。

5.1 导入数据

我们可以使用 Maple 中的多种交互式工具或命令从外部文件中输入数据，以及输出数据到文件中，数据类型支持许多格式，例如 Excel，TXT，ASCII，CSV，图片，声音，矩阵，分隔符文件等。

5.1.1 使用图形化用户界面导入数据文件

可以使用交互式工具输入和拟合数据 – 图形用户界面。



Maple 的图形用户界面 Curve Fitting Assistant 提供曲线拟合分析，帮助用户快速输入和分析数据。注意：Curve Fitting Assistant 仅提供有限的功能，Maple 中的命令能提供远远超过该助手的工具和功能。

在上图中我们通过鼠标操作输入了 Excel 数据文件，或者我们也可以直接选中和拷贝 Excel

中的数据到 Maple 工作区，然后使用鼠标右键菜单，选择 Curve Fitting Assistant 菜单进入上面的界面。Curve Fitting Assistant 提供了三种数据拟合，分别是多项式插值、样条曲线（需要定义插值的次数）、最小二次方，并绘制出对应的图形。界面的下方是拟合后的数学函数，你可以选择返回数学模型或图形。

5.1.2 使用命令导入 Excel 文件中的数据

可通过以下几种方式导入 Excel 文件中的数据。

1）使用 ExcelTools 函数包中的 Import 命令

```
with(ExcelTools):      #加载 ExcelTools 函数包
data1 := Import("C:\\Program Files\\Maple 2017\\data\\portal\\ExcelData.xls")   #导入 Excel 数据文件
```

$$data1 := \begin{bmatrix} 30 \times 2\ Matrix \\ Data\ Type:\ anything \\ Storage:\ rectangular \\ Order:\ Fortran_order \end{bmatrix}$$

```
whattype(data1)      #查看数据类型
```

Matrix

导入的 Excel 数据被赋值给变量 data1，从输出的结果看，data1 是一个 30x2 的矩阵。

2）使用 Import 命令

注意：这里的 Import 命令与上面的 ExcelTools 函数包中的 Import 命令是不同的。Import 命令是一个可以导入广泛数据类型的命令，包括图像文件、文本文件、数据表文件、几何体文件等，当导入的文件格式是矩形时，返回一个 DataFrame 对象。DataFrame 是一个两维数据容器，类似于矩阵，区别是 DataFrame 可以包含异类数据，并可以使用符号名称关联行和列。

```
data2 := Import("C:\\Program Files\\Maple 2017\\data\\portal\\ExcelData.xls")   #导入 Excel 数据文件
```

$$data2 := \begin{bmatrix} & 1 & 2 \\ 1 & 0. & 0.918632608623005 \\ 2 & 0.100000000000000 & 0.668029131068941 \\ 3 & 0.200000000000000 & 0.953058689602197 \\ 4 & 0.300000000000000 & 0.759362374941990 \\ 5 & 0.400000000000000 & 0.620458639476208 \\ 6 & 0.500000000000000 & 1.11081409356008 \\ 7 & 0.600000000000000 & 0.366527091766061 \\ 8 & 0.700000000000000 & 0.862156999926789 \\ \dots & \dots & \dots \end{bmatrix}$$

```
whattype(data2)      #查看数据类型
```

DataFrame

导入的 Excel 数据被赋值给变量 data2，从输出的结果看，data2 是一个 DataFrame 矩阵。

可以将 DataFrame 转化为矩阵数据文件。

```
data3 := convert( data2, Matrix )
```

$$data3 := \begin{bmatrix} 30 \times 2 \text{ Matrix} \\ \text{Data Type: } float_8 \\ \text{Storage: rectangular} \\ \text{Order: Fortran_order} \end{bmatrix}$$

5.2 拟合数据

Maple 中的 Statistics 和 CurveFitting 函数包提供多个命令实现数据拟合，包括线性、指数、多项式、最小二乘法、样条函数。

常用的数据拟合命令有：

Statistics 函数包	CurveFitting 函数包
ExponentialFit	ArrayInterpolation
Fit	BSpline
LinearFit	BSplineCurve
LogarithmicFit	Interactive
Lowess	LeastSquares
NonlinearFit	Lowess
OneWayANOVA	PolynomialInterpolation
PolynomialFit	RationalInterpolation

PowerFit PredictiveLeastSquares RepeatedMedianEstimator	Spline ThieleInterpolation

关于以上命令的调用格式和详细信息见 Maple 帮助系统。

5.3.1 Fit 命令

下面以 Statistics 函数包中的 Fit 命令为例，说明如何使用 Fit 拟合数据。Statistics:-Fit 命令使用最小二乘误差方法拟合模型，可实现线性和非线性模型的拟合。

调用格式：

Fit(f, X, Y, v, options)

Fit(f, XY, v, options)

其中：

f – 模型函数，表达式

X – 自变量，向量或者矩阵数据

Y – 因变量，向量或矩阵数据

XY – 自变量和因变量，矩阵数据

v – 自变量和因变量的名称，变量名或列表

options – 参数项，例如可以定义输出结果的内容等

例子：

```
with(Statistics):    #加载统计函数包
```

定义一个向量类型的数据组，赋值给自变量 X：

```
X := Vector([1, 2, 3, 4, 5, 6], datatype = float);      #定义一个向量
```

定义一个向量类型的数据组，赋值给因变量 Y：

```
Y := Vector([1, 2, 3, 4, 5, 6], datatype = float);      #定义一个向量
```

使用 Fit 命令，得到拟合后的模型，并显示模型的摘要信息。

```
Fit(b*t+a, X, Y, t);      #拟合命令
```

$$0.926666666666669 + 0.940000000000000\,t$$

其中 b*t+a 是拟合模型，X 是自变量，Y 是因变量，t 是拟合模型中的自变量。

拟合为二阶多项式。

```
Fit(c*t^2+b*t+a, X, Y, t);      #拟合命令
```

$$1.96000000000000 + 0.164999999999999\,t + 0.110714285714286\,t^2$$

如果想从回归模型中返回特定的值，参数项 'output' 非常有用。可用的 output 参数项有：

AtkinsonTstatistic	leverages	rsquared
confidenceintervals	parametervalues	rsquaredadjusted
CookDstatistic	parametervector	standarderrors
degreesoffreedom	residuals	tprobability
externallystandardizedresiduals	residualmeansquare	tvalue
internallystandardizedresiduals	residualstandarddeviation	variancecovariancematrix
leastsquaresfunction	residualsumofsquares	

输出标准误差：

```
Fit(c*t^2+b*t+a, X, Y, t, output=standarderrors);      #拟合命令
```

$$\begin{bmatrix} 1.17199057366598 & 0.766748258006800 & 0.107226158093964 \end{bmatrix}$$

输出回归的回归的可决系数(r-squared)和回归的修正可决系数(adjusted r-squared)的值：

```
Fit(c*t^2+b*t+a, X, Y, t, output=[rsquared,rsquaredadjusted])      #拟合命令
```

$$[0.925169145624351, 0.875281909373919]$$

通常以汇总表的形式显示结果是一个有用的途径。

```
Fit(c*t^2+b*t+a, X, Y, t, summarize=true):      #拟合命令
```

```
Summary:
----------------
Model: 1.9600000+.16500000*t+.11071429*t^2
----------------
Coefficients:
Estimate   Std. Error   t-value   P(>|t|)
a      1.9600      1.1720       1.6724    0.1930
b      0.1650      0.7667       0.2152    0.8434
c      0.1107      0.1072       1.0325    0.3778
----------------
R-squared: 0.9252, Adjusted R-squared: 0.8753
```

也可用表格的形式显示汇总信息，可以了解关于残差的更多信息。

```
Fit(c*t^2+b*t+a, X, Y, t, summarize=embed):        #拟合命令
```

Summary

Model: $1.9600000 + 0.16500000\,t + 0.11071429\,t^2$

系数	Estimate	Standard Error	t-value	P(>|t|)
a	1.96000	1.17199	1.67237	0.193045
b	0.165000	0.766748	0.215194	0.843415
c	0.110714	0.107226	1.03253	0.377769

R-squared: 0.925169

Adjusted R-squared: 0.875282

▶ Residuals

Fit 命令中的数据也可以是矩阵格式的文件，不需要预先对数据文件进行 X 和 Y 的分离处理。调用格式是 Fit(f, XY, v, options)。

示例：变量 x, y, z, w 的关系如下，未知的参数是 a, b, c。已知一个 6 组数据点的矩阵，矩阵中的列分别对应 x, y, z 和 w 的值

$$w = x^a + \frac{bx^2}{y} + cyz$$

定义一个矩阵类型的数据变量 ExperimentalData。

```
ExperimentalData := <1, 1, 1, 2, 2, 2 | 1, 2, 3, 1, 2, 3 | 1, 2, 3, 4, 5, 6 | 0.531, 0.341, 0.163, 0.641, 0.713, -0.040>;        #创建一个矩阵
```

$$ExperimentalData := \begin{bmatrix} 1 & 1 & 1 & 0.531 \\ 1 & 2 & 2 & 0.341 \\ 1 & 3 & 3 & 0.163 \\ 2 & 1 & 4 & 0.641 \\ 2 & 2 & 5 & 0.713 \\ 2 & 3 & 6 & -0.040 \end{bmatrix}$$

为了得到正确的拟合结果，我们首先做一些假设，猜测这个数学模型可能是一个 2 次方程，b 近似于 1，对于 c，我们不知道是正值还是负值，我们假设是 c=0。

```
infolevel[Statistics] := 2:      #获得 Maple 计算的更多信息
NonlinearFit(x^a + b*x^2/y + c*y*z, ExperimentalData, [x, y, z], initialvalues=[a = 2, b = 1, c = 0], output=[leastsquaresfunction, residuals]);      #非线性拟合
```

In NonlinearFit (algebraic form)

$$\left[x^{1.14701973996968} - \frac{0.298041864889394\,x^2}{y} - 0.0982511893429762\,yz, [0.0727069457676300, 0.116974310183398, -0.146607992383251, -0.0116127470057686, -0.0770361532848388, 0.0886489085642805]\right]$$

从输出的结果信息可以看到，Maple 选择了非线性拟合算法。

我们改变模型表达式，如下：

$$w = ax + \frac{bx^2}{y} + cyz$$

```
Fit(a*x + b*x^2/y + c*y*z, ExperimentalData, [x, y, z], initialvalues=[a = 2, b = 1, c = 0],
output=[leastsquaresfunction, residuals], summarize=embed);
```

In Fit
In LinearFit (container form)
final value of residual sum of squares: .0537598869493245

$$\left[0.823072918385878\,x - \frac{0.167910114211606\,x^2}{y} - 0.0758022678386438\,yz,\ [-0.0483605363356285, -0.0949087899254999,\right.$$
$$\left.0.0781175302268541, -0.0302963085707583, 0.160697070037893, -0.0978248634499976]\right]$$

Model: $0.82307292\,x - \frac{0.16791011\,x^2}{y} - 0.075802268\,yz$

Coefficients	Estimate	Standard Error	t-value	P(>|t|)
a	0.823073	0.189761	4.33742	**0.0226122**
b	−0.167910	0.0940047	−1.78619	0.172045
c	−0.0758023	0.0182477	−4.15408	**0.0253587**

R-squared: 0.960049

Adjusted R-squared: 0.920099

Residuals

这一次，Maple 自动选择了一个线性拟合算法，因为表达式是线性的。

5.3.2 拟合噪声信号的模型

为了进一步熟悉和理解数据拟合的应用，我们再看一些示例。

这个例子将使用随机数生成程序添加噪声到信号中，然后拟合模型。

```
restart:      #清除内存
with(Statistics):      #加载 Statistics 函数包
```

使用均值为 1 标准偏差为 0.5 的随机数生成器，定义一个噪声数据。

```
Noise := Sample(RandomVariable(Normal(1, .5))):      #使用随机数生成样本数据
```

创建数量为 200 的噪声样本数据，数据类型为向量。

```
PureNoiseVector := Noise(200)      #生成一个 200 样本向量
```

$$PureNoiseVector := \begin{bmatrix} 1 .. 200\ Vector_{row} \\ Data\ Type: float_8 \\ Storage: rectangular \\ Order: Fortran_order \end{bmatrix}$$

我们可以从向量中提取数据项。向量的索引从 1 到 200。想要获取第 i 个数据项，使用选择运算符：PureNoiseVector[i].

```
PureNoiseVector[1]      #取向量 PureNoiseVector 第一行的值
```

0.972074978004195

接下来，我们定义一个信号数据，然后添加噪声到信号数据中：

```
Signal := x -> x*sin(x):      #定义一个函数
```

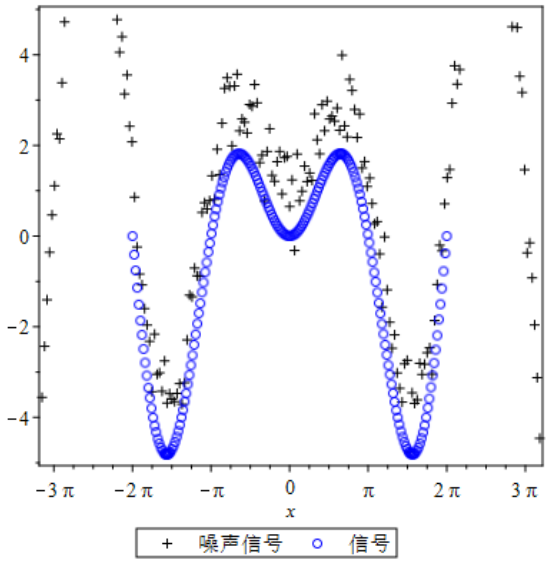
使用序列命令 seq 生成一个数据集。

```
noisyData := [seq([.1*(i-100), signal(.1*(i-100))+PureNoiseVector[i]], i = 1..200)]: #使用 seq 命令定义一个列表数据
```

noisyData 是一组坐标值的列表，其中 x 是 0.1*(i-100)，y 是 signal(0.1*(i-100))+PureNoiseVector[i]]，i 从 1 到 200.

在同一幅图上画出 signal 和 noisyData 的二维图形。

```
plot([noisyData, signal(x)], style=point, symbol=[cross, circle], symbolsize=15, color=["Black", "Blue"], legend=["噪声信号", "信号"], view=[default,-5..5], axes=boxed)      #画出 noisyData 和 signal 的图形
```



拟合数据对应的模型：使用统计函数包中的 Fit 命令拟合噪声数据对应的函数模型。

变量 nosieData 包含两列数据，分别提取创建两个列表 Xdata 和 Ydata。

```
Xdata := [seq(noisyData[i, 1], i = 1 .. numelems(noisyData))]:     #创建一个列表数据
Ydata := [seq(noisyData[i, 2], i = 1 .. numelems(noisyData))]:     #创建一个列表数据
```

注意：通常我们会处理数据对象中的元素项。这里，我们使用 i=1..numelems(noisyData)，其中命令 numelems 是获得元素项的数量。

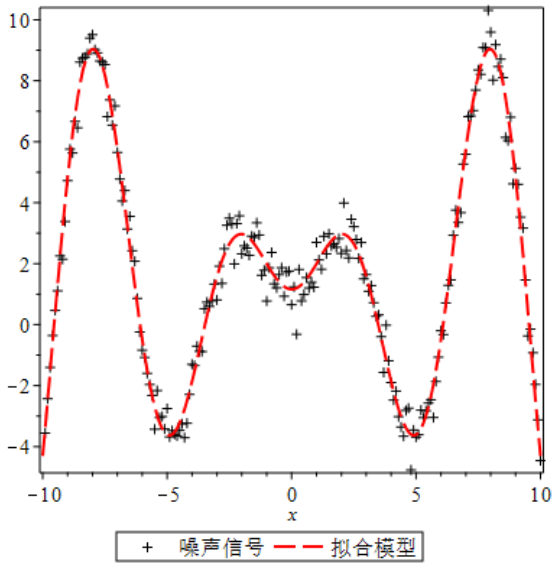
函数模型是 $f=c\cdot x\cdot \sin(a\cdot x)+b$ 。

```
f := Fit(c*x*sin(a*x)+b, Xdata, Ydata, x)      #拟合模型
```

```
f := .996083910736681*x*sin(1.00054261797457*x)+1.15827595547478
```

在同一幅图中画出模型 f 和噪声数据 noisyData 的图形。

```
plot([noisyData, f(x)], x = -10 .. 10, style=[point, line], symbol=cross, symbolsize=15,
legend=["噪声信号", "拟合模型"], axes=boxed, color=["Black","Red"], linestyle=dash,
thickness=2)       #画出 noisyData 和 f(x)的图形
```



5.3.3 最小二乘拟合

有许多方法可以实现数据的曲线或曲面拟合。优化函数包提供命令用于求优化问题的数值解，求解目标函数满足约束条件时的最小或最大值。

Optimization 优化函数包可以求解线性规划、二次规划、非线性规划、以及线性和非线性最小二乘问题。

```
restart;      #重启内存
with(Optimization);     #加载优化函数包
```

[ImportMPS, Interactive, LPSolve, LSSolve, Maximize, Minimize, NLPSolve, QPSolve]

使用最小二乘方法实现曲线拟合。例如，有下面的数据点(x_i, y_i)：

```
data := [[1, 1.5], [2, 3.5], [2.5, 1.9], [3.1, 4.5], [4.3, 1.9], [4.7, 2.4], [5.8, 3], [6, 5.7], [6.6, 3.4], [6.7, .5], [7.1, 4], [7.4, 1.7]];     #定义一个列表类型的数据
```

使用 5 次多项式拟合数据：

$$p(x) = ax^5 + bx^4 + cx^3 + dx^2 + ex + f$$

需要求解得到 a, b, c, d, e, 和 f．求如下表达式的最小值：

$$\frac{1}{2}\sum_{x=1}^{12}\left(p(x_i) - y_i\right)^2$$

LSSolve 命令将最小二乘残差$p(x_i) - y_i$作为输入。

```
p := a*x^5+b*x^4+c*x^3+d*x^2+e*x+f;        #定义模型
residues := map( d-> eval(p, x = d[1])-d[2], data);     #求残差的和
```

$$residues := [a+b+c+d+e+f-1.5,\ 32a+16b+8c+4d+2e+f-3.5,\ 97.65625a+39.0625b+15.625c+6.25d+2.5e+f-1.9,\ 286.29151a+92.3521b+29.791c+9.61d+3.1e+f-4.5,\ 1470.08443a+341.8801b+79.507c+18.49d+4.3e+f-1.9,\ 2293.45007a+487.9681b+103.823c+22.09d+4.7e+f-2.4,\ 6563.56768a+1131.6496b+195.112c+33.64d+5.8e+f-3,\ 7776a+1296b+216c+36d+6e+f-5.7,\ 12523.32576a+1897.4736b+287.496c+43.56d+6.6e+f-3.4,\ 13501.25107a+2015.1121b+300.763c+44.89d+6.7e+f-0.5,\ 18042.29351a+2541.1681b+357.911c+50.41d+7.1e+f-4,\ 22190.06624a+2998.6576b+405.224c+54.76d+7.4e+f-1.7]$$

其中，map 命令是映射命令，将函数 d-> eval(p, x = d[1])-d[2]映射到数据集 data 上，作用于其中的所有元素项。eval(p, x = d[1])-d[2] 是求值，d[1]是取 d，也就是 data 的值，依次是[1, 1.5]，[2, 3.5]…

对 residues 使用 LSSolve 命令。

```
sol := LSSolve(residues);      #求解最小二乘问题
```

sol := [9.79941650829796096, [a = 0.506817808955152e-2, b = -.148717633940652, c = 1.53722867786023, d = -7.04487080570615, e = 14.2425902266783, f = -7.12139344171469]]

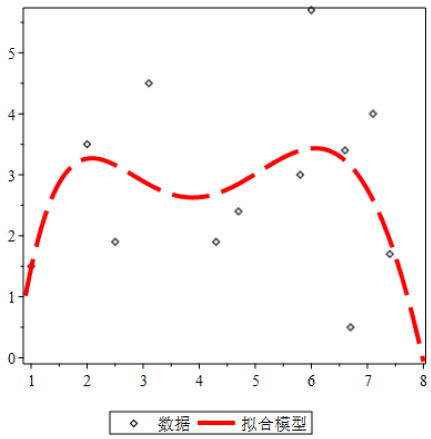
代入系数值，多项式变为：

```
poly := eval(p, sol[2]);     #代入参数值 sol[2]到表达式 p 中
```

poly := 0.506817808955152e - 2*x^5 - .148717633940652*x^4 + 1.53722867786023*x^3 - 7.04487080570615*x^2 + 14.2425902266783*x - 7.12139344171469

画出数据和曲线的图形。

```
p1 := Statistics:-ScatterPlot(data, symbolsize=15, color="Black", legend="数据"): #画出散点图
p2 := plot(poly, x = .9 .. 8, thickness=4, color="Red", linestyle=dash, legend="拟合模型"):    #画出多项式 poly 的图形
plots:-display(p1, p2, axes=boxed);      #在同一幅图上显示多个图形
```



5.3.4 三维数据的多项式拟合

在这个例子中，我们将完成：

1）使用 Maple 中的工具生成一个三维数据，包含 X, Y, Z 点的值，Z 点中加入一些随机噪声；

2）定义一个二元多项式，阶数可以修改；

3）使用最小二乘方法拟合多项式模型；

4）画出原数据和拟合模型的曲面图形。

首先，生成数据。

```
restart;      #重启内存
X := Vector(10, i -> i):        #定义一个向量
Y := Vector(10, i -> i):        #定义一个向量
```

定义 Z 点，注意添加了随机噪声。

```
Z := Matrix(10, 10, (i, j) -> i^2+j*i+3*j^2.3+ 6*10^(-11)*rand(), datatype = float[8]):
```

```
#定义一个 10X10 的矩阵，其中 rand()是生成随机数的命令
```

接下来，我们定义个二元多项式作为要拟合的数学模型。

```
f := n -> sum(sum(a[i, j]*x^i*y^j, i=0..n), j=0..n)        #sun 是求和命令，->是函数定义运算符
```

$$f := n \to \sum_{j=0}^{n} \sum_{i=0}^{n} a_{i,j}\, x^i\, y^j$$

```
n := 3        #定义多项式的阶数
```

列出要拟合的未知变量。

```
vars := sort([seq(seq(a[i, j], i = 0..n), j = 0..n)])[]        #sort 是对一组值或多项式进行排序
```

$$vars := a_{0,0}, a_{0,1}, a_{0,2}, a_{0,3}, a_{1,0}, a_{1,1}, a_{1,2}, a_{1,3}, a_{2,0}, a_{2,1}, a_{2,2}, a_{2,3}, a_{3,0}, a_{3,1}, a_{3,2}, a_{3,3}$$

生成二元多项式。

```
poly := unapply(f(n), x, y, vars)     #unapply 命令的结果是一个函数运算符，x,y,vars 是自变量列表，f(n)是函数
```

$$poly := (x, y, a_0_0, a_0_1, a_0_2, a_0_3, a_1_0, a_1_1, a_1_2, a_1_3, a_2_0, a_2_1, a_2_2, a_2_3, a_3_0, a_3_1, a_3_2, a_3_3) \to a_3_3\, x^3 y^3 + a_2_3\, x^2 y^3 + a_3_2\, x^3 y^2 + a_1_3\, x y^3 + a_2_2\, x^2 y^2 + a_3_1\, x^3 y + a_0_3\, y^3 + a_1_2\, x y^2 + a_2_1\, x^2 y + a_3_0\, x^3 + a_0_2\, y^2 + a_1_1\, x y + a_2_0\, x^2 + a_0_1\, y + a_1_0\, x + a_0_0$$

接下来，计算出最佳的拟合参数。

残差是：

```
residuals := seq(seq(poly(X[i], Y[j], vars)-Z[i, j], i=1..10), j=1..10):     #通过 seq 中的循环功能实现差值的和
```

对残差求最小二乘最小值。

```
results := Optimization[LSSolve]([residuals])    #使用优化函数包中的最小二乘命令
```

```
results := [13591.9957858173693, [a[0, 0] = 41.6887760283600, a[0, 1] = -42.8447036481329, a[0, 2] = 15.7116959691764, a[0, 3] = -.566228245257700, a[1, 0] = 1.07371116140913, a[1, 1] = 24.9091742930187, a[1, 2] = -7.13493562688761, a[1, 3] = .484977722795804, a[2, 0] = .678577157737150, a[2, 1] = -4.57185889860260, a[2, 2] = 1.37195732911947, a[2, 3] = -0.933904284937924e-1, a[3, 0] = 0.148702330626055e-2, a[3, 1] = .267679197709019, a[3, 2] = -0.788087966565691e-1, a[3, 3] = 0.530462719799963e-2]]
```

```
fitVars := map(rhs, results[2])[]      #rhs 是取右边的值，result[2]是获取上面输出结果的第二项，map 是将 rhs 命令映射到 result[2]上
```

```
fitVars := 41.6887760283600, -42.8447036481329, 15.7116959691764, -.566228245257700,
```

1.07371116140913, 24.9091742930187, -7.13493562688761, .484977722795804, .678577157737150, -4.57185889860260, 1.37195732911947, -0.933904284937924e-1, 0.148702330626055e-2, .267679197709019, -0.788087966565691e-1, 0.530462719799963e-2

得到拟合后的模型表达式。

```
model := poly(x, y, fitVars)    #生成坐标值列表
```

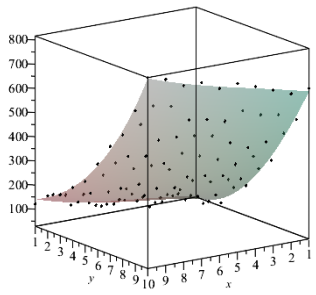
```
model := 0.530462719799963e - 2*x^3*y^3-0.933904284937924e - 1*x^2*y^3 - 0.788087966565691e - *x^3*y^2+.484977722795804*x*y^3 + 1.37195732911947*x^2*y^2
+ .267679197709019*x^3*y-.566228245257700*y^3-7.13493562688761*x*y^2-4.57185889860260*x^2*y +0.148702330626055e-2*x^3+15.7116959691764*y^2+24.9091742930187*x*y+.678577157737150*x^2-42.8447036481329*y+1.07371116140913*x+41.6887760283600
```

最后，用实验数据拟合曲面模型。

```
L := [seq(seq([X[i], Y[j], Z[i, j]], i = 1 .. 10), j = 1 .. 10)]:    #生成坐标值列表
p1 := plots[pointplot3d](L, color = black):    #画原数据点的图形
```

画出拟合模型的图形。

```
p2 := plot3d(model, x = min(X) .. max(X), y = min(Y) .. max(Y), style = patchnogrid, transparency = .5)    #画出拟合模型的图形
plots[display](p1, p2, axes = boxed)    #在同一幅图上显示和对比两个图形
```



5.4 提取图形的绘图数据并输出 EXCEL 文件

在 Maple 中，我们可以绘制二维或者三维图形，一些情况下，我们希望获得这些图形的绘图数据，并输出为 Excel 格式的数据文件。

绘图，并将图形结构赋值给变量 plot1。

```
restart:
plot1 := plot3d(x^2+y^2, x = -1 .. 1, y = -1 .. 1);
```

加载 plottool 函数包，提取绘图数据。

```
with(plottools):
data1 := getdata(plot1)[3];
```

$$data1 := \begin{bmatrix} 1..49 \times 1..49 \ Array \\ Data\ Type: float_8 \\ Storage: rectangular \\ Order: Fortran_order \end{bmatrix}$$

双击输出的结果矩阵，可以输出 EXCEL 数据文件。

第六章 微分方程求解

Maple 中提供多个命令和函数包求解常微分方程和偏微分方程的数值解和解析解，可求解如下的问题：

常微分方程（ODEs）：dsolve 命令用于求解线性和非线性 ODEs，初始值问题 (IVP), 以及边界值问题 (BVP),，可以通过参数项选择求解析解或数值解。微分方程分析器助手提供一个交互式用户界面方便用户求解常微分方程以及显示结果的图形。更多细节信息见帮助系统中的 dsolve, dsolve/numeric, 和 ODE Analyzer。

偏微分方程 (PDEs)：pdsolve 命令用于求 PDEs 和含边界值问题的 PDEs 的符号解或数值解。使用 Maple 的 PDE 工具可以完成对 PDE 系统的结构分析和指数降阶处理。更多细节信息见帮助系统中的 pdsolve 和 pdsolve/numeric。

微分代数方程（DAEs）：dsolve/numeric 命令是符号-数值混合求解器，使用符号预处理和降阶技术，让 Maple 能够求解高指数的 DAE 问题。Maple 内置 rkf45, ck45, rosenbrock, bvp, rkf45_dae, ck45_dae, rosenbrock_dae, dverk78, lsode, gear, taylorseries, mebdfi, 和 classical 等数值求解算法，使用这些算法可以获得数值解。

6.1 定义微分方程

在 Maple 中，用 diff(y(x),x) 表示 $\frac{dy}{dx}$，用 diff(y(x),x,x)或 diff(y(x),x$2)表示 $\frac{d^2y}{dx^2}$。Diff(f, [x1$n]) 表示 $\frac{d^n}{dx_1^n}f$；diff(f, x1$n, [x2$n, x3], …, xj, [xk$m]) 表示 $\frac{d^r}{dx_k^m dx_j \cdots dx_3 dx_2^n dx_1^n}f$。偏微分符号与常微分符号一样。

(1) diff()

```
diff(x(t),t);     #一阶微分
```

$$\frac{d}{dt}x(t)$$

```
diff(x(t),t,t);     #二阶微分
```

$$\frac{d^2}{dt^2}x(t)$$

(2) 点符号

在**数学模式**下，输入微分方程时可以使用点符号表示微分（相对于时间 t）。

```
ẏ    #一阶微分
```

$$\frac{\mathrm{d}}{\mathrm{d}t}\,y(t)$$

输入步骤：①确认当前的输入模式是数学 文本 数学（可以使用 F5 键在文本和数学模式之间切换）；②输入 y；③按组合键 Shift + Ctrl + “；④输入 . (英文句号) 表示一阶导数或者.. (两个英文句号) 表示二阶导数；⑤输入完成后按回车键。

输入上述微分表达式时，可以按下快捷键 Ctrl+F2，或者从菜单帮助->快速参考，在弹出的 Quick Reference Card 中可以查到输入方法。

(3) 定义初始条件

D 符号用于表示求在某一点上的导数值。

```
D(y)(0);    #初始条件
```

$$\mathrm{D}(y)(0)$$

6.2 使用 dslove 命令求解析解和数值解

求解常微分方程最简单的方法是利用求解函数 dsolve。调用格式是：

dsolve(eqn, y(x), 参数项);　　#求解常微分方程 eqn，y 是 x 的函数

dsolve({ODE, ICs}, y(x), 参数项);　　#求解含初始条件边界条件的常微分方程 eqn

dsolve({sysODE, Ics}, {funcs}, 参数项);

其中, ODE 表示常微分方程, y(x)表示单变量的任意变量函数, Ics 表示初始条件, {sysODE}表示 ODE 方程组的集合, {funcs}表示变量函数的集合，参数项表示依赖于要求解的问题类型。

需要注意的是，无论在方程中还是作为第二个参数，未知函数必须用函数的形式给出（即必须加括号，并在其中明确自变量），这一规定是必须的，否则 Maple 将无法区分方程中的函数、自变量和参变量，这一点和我们平时的书写习惯不一致。

Dsolve 默认求解常微分方程的精确解（解析解），所求的解可以是通解，也可以是满足初始条件或边界条件的特解。通解中的_C1，_C2 等是用下划线起始的内部变量，表示积分常数。

示例：使用 dsole 命令求解单个常微分方程

6.2.1 定义微分方程

```
de := diff(y(t),t,t) + diff(y(t),t) + y(t) = 3;     #定义微分方程
```

$$de := \frac{d^2}{dt^2} y(t) + \frac{d}{dt} y(t) + y(t) = 3$$

```
ic := D(y)(0)=3, y(0)=4;     #定义初始条件
```

$$ic := D(y)(0) = 3, y(0) = 4$$

6.2.2 求解析解

```
sol := dsolve({de,ic});     #dsolve 命令求解
```

$$sol := y(t) = \frac{7}{3} e^{-\frac{1}{2}t} \sin\left(\frac{1}{2}\sqrt{3}\, t\right)\sqrt{3} + e^{-\frac{1}{2}t} \cos\left(\frac{1}{2}\sqrt{3}\, t\right) + 3$$

```
y_func := unapply(rhs(sol),t);    # rhs 是取结果的等式右侧部分，unapply 将结果转变为函数 y_func(t)
```

$$y_func := t \to \frac{7}{3} e^{-\frac{1}{2}t} \sin\left(\frac{1}{2}\sqrt{3}\, t\right)\sqrt{3} + e^{-\frac{1}{2}t} \cos\left(\frac{1}{2}\sqrt{3}\, t\right) + 3$$

```
y_func(7.0);;    # t=7.0 的解
```

$$0.07046056131 \sin(3.500000000\sqrt{3})\sqrt{3} + 0.03019738342 \cos(3.500000000\sqrt{3}) + 3$$

以上结果是精确解，可以求指定位数的近似解。

```
Evalf[10](%);    #求 10 位近似解
```

3.002709897

6.2.3 求数值解

许多微分方程都无法求得精确解，此时只好退而求其次，求其数值解。若方程式包含初始条件，则可以在 dsolve()命令之后加上一个选项 method=numeric，告诉 Maple 求数值解。

调用格式：dsolve({eqn, ini_conds}, y(x), number)

dsolve 返回的结果是一个程序体(procedure)，我们只要给予该程序体特定的值，此程序体将会返回相关解的信息。对于初始值问题 IVP，dsolve 默认使用 Runge-Kutta Fehlberg 方法，得

到一个 5 阶精确解。

```
Sol:=dsolve({de,ic},numeric);      #添加参数项 numeric 强制 dsolve 求数值解
```

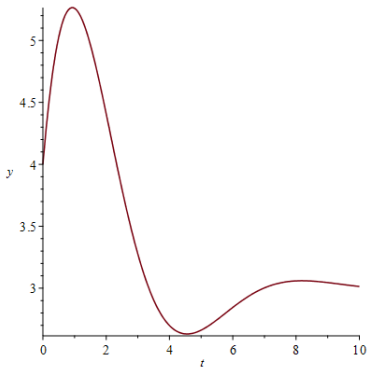
sol:=proc(x_rkf45) … end proc

```
sol(3);      #求 t=3 时的值
```

$$\left[t = 3., y(t) = 3.27537298676751, \frac{d}{dt} y(t) = -0.906034403201660\right]$$

画出解的图形。

```
Plots:-odeplot(sol,0..10);      #使用 plot 绘图函数包中的 odeplot 命令画出解在 0 到 10 范围内的图形
```



使用 Laplace 变换方法求数值解。

```
Sol := dsolve({de, ic}, y(x), method=laplace);      #使用 method=laplace 求数值解
```

$$sol := y(t) = 3 + \frac{1}{3} e^{-\frac{1}{2}t}\left(7\sqrt{3}\sin\left(\frac{1}{2}\sqrt{3}\,t\right) + 3\cos\left(\frac{1}{2}\sqrt{3}\,t\right)\right)$$

检测上面的解是否满足原有常微分方程和初始条件。

```
Odetest(sol, [de, ic]);      #判定解的正确性
```

[0, 0, 0]

6.2.4 求级数解

```
series_sol := dsolve({de,ic}, y(t), series);      #使用参数项 series 求级数解
```

$$series_sol := y(t) = 4 + 3t - 2t^2 + \frac{1}{6}t^3 + \frac{1}{8}t^4 - \frac{1}{30}t^5 + O(t^6)$$

注意，dsolve()命令在求解幂级数解解，默认的阶数是 6。如果要改变阶数，可通过全局变量 Order 来改变。如果阶数设定为 8 阶，则得到：

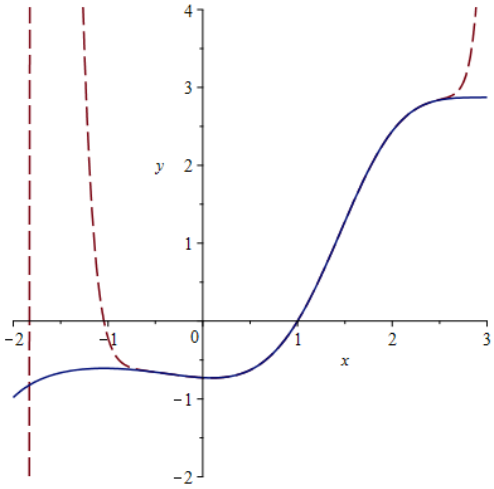
```
Order := 8:      #设置 8 阶
```

```
series_sol := dsolve({de,ic}, y(t), series);        #求 8 阶级数解
```

$$series_sol := y(t) = 4 + 3t - 2t^2 + \frac{1}{6}t^3 + \frac{1}{8}t^4 - \frac{1}{30}t^5 + \frac{1}{720}t^6 + \frac{1}{1680}t^7 + O(t^8)$$

如果我们设定幂级数解的阶数为 24，并将其相应的多项式解与方程的数值解进行比较。

```
Restart:        #清除内存
de := diff(y(t),t,t) + diff(y(t),t) + y(t) = 3;        #定义微分方程
ic := D(y)(0)=3, y(0)=4;        #定义初始条件
nsol := dsolve({de,ic}, y(t), numeric):        #求数值解
Order := 24:        #级数解的阶数是 24
dsolve({de,ic}, y(t), series):        #求微分方程的级数解
p24 := convert(rhs(%), polynom):        #将级数转化为多项式
plot([p24, 'rhs'('nsol'(t)[2])], t=-20..20, y=-10..10, linestyle=[3,1]);        #其中虚线是数值解
```



可以看到级数解的阶数增大时，级数解接近数值解的区域也就更加宽广。

6.3 微分方程的判定

使用 DETools 函数包中的 odeadvisor 命令判定微分方程所属可解类型，例如变量可分离方程、齐次方程、线性方程、全微分方程。详细介绍见?odeadvisor。

示例：验证方程$\frac{d}{dx}=\phi(x)\quad g\left(\frac{y}{x}\right)+\frac{y}{x}$在变量变为$y = xu$时可分离方程。

```
Restart:    #重启内存
```

```
with(DETools):    #加载 DETools 微分方程工具函数包
eqn := diff(y(x),x)=phi(x)*g(y(x)/x)+y(x)/x;    #定义微分方程
```

$$eqn := \frac{d}{dx}y(x) = \phi(x)\, g\left(\frac{y(x)}{x}\right) + \frac{y(x)}{x}$$

```
value(subs(y(x)=x*u(x),%));   # value 对替换后的表达式求值
```

$$\frac{d}{dx}\left(x\ u(x)\right) = \phi(x)\ \ g(u(x)) + u(x)$$

```
odeadvisor(%);      #判定类型
```

[_separable]

结果表示变换后的方程是变量可分离的。

在 Maple 中，微分方程的解是很容易验证的，只需要将解代入到原方程并化简就可以了。例如，对于一阶微分方程$xy'=y\ln(xy)-y$使用 dsolve 直接求得解析解。

```
Restart:      #清除内存
ode := x*diff(y(x), x)= y(x)*ln(x*y(x))-y(x);      #定义微分方程
```

$$ode := x\left(\frac{d}{dx}\ y(x)\right) = y(x)\ \ \ln(x\ \ y(x)) - y(x)$$

```
dsolve(ode, y(x));      #解微分方程
```

$$y(x) = \frac{e^{-\frac{x}{CI}}}{x}$$

```
odetest(%, ode);      #判定解
```

0

odetest 命令是检验常微分方程求解器所得的显式和隐式解。如果解为有效的，则返回值为 0；否则返回剩余代数表达式。

```
Subs(%, ode);      #将 y(x)的解代入原方程
simplify(%) assuming _C1::real,   x::real;      #化简
evalb(%);    #判定解
```

True

可以提高 infolevel[dsolve]等级获取计算过程的更多信息。

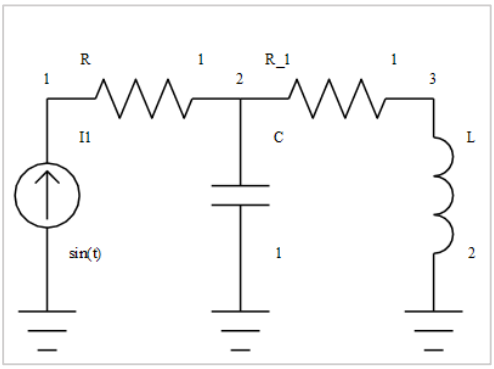
```
Infolevel[dsolve]:=3:    #设置求解过程信息等级
ode := x*diff(y(x), x)= y(x)*ln(x*y(x))-y(x);      #解微分方程
```

Methods for first order ODEs:
--- Trying classification methods ---
trying a quadrature
trying 1st order linear
trying Bernoulli
trying separable
trying inverse linear
trying homogeneous types:
trying homogeneous G
<- homogeneous successful

$$y(x) = \frac{e^{-\frac{x}{CI}}}{x}$$

6.4 求解一阶常微分方程 ODE

[例1] 求解电路的微分方程。



电路方程通过下面的微分方程表示

deqs := diff(i__L(t), t) = -1/2*i__L(t) + 1/2*v__C(t), diff(v__C(t), t) = -i__L(t) + sin(t), i__L(0) = 1, v__C(0) = 2

$$deqs := \frac{d}{dt} i_L(t) = -\frac{1}{2} i_L(t) + \frac{1}{2} v_C(t), \frac{d}{dt} v_C(t) = -i_L(t) + \sin(t), i_L(0) = 1, v_C(0) = 2$$

求符号解：

sol1 := dsolve({deqs})

$$sol1 := \left\{ i_L(t) = \frac{11}{14} e^{-\frac{1}{4}t} \sin\left(\frac{1}{4}\sqrt{7}\,t\right)\sqrt{7} + \frac{3}{2} e^{-\frac{1}{4}t} \cos\left(\frac{1}{4}\sqrt{7}\,t\right) - \frac{1}{2}\cos(t) - \frac{1}{2}\sin(t),\ v_C(t) = -\frac{5}{14} e^{-\frac{1}{4}t} \sin\left(\frac{1}{4}\sqrt{7}\,t\right)\sqrt{7} + \frac{7}{2} e^{-\frac{1}{4}t} \cos\left(\frac{1}{4}\sqrt{7}\,t\right) + \frac{1}{2}\sin(t) - \frac{3}{2}\cos(t) \right\}$$

求数值解：

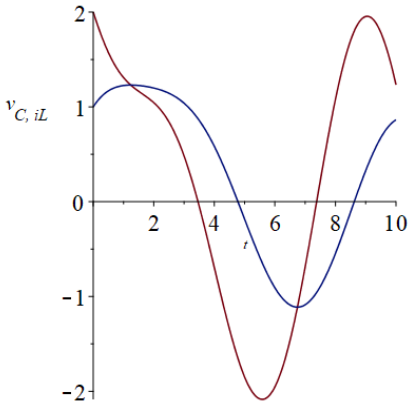
sol2 := dsolve({deqs}, numeric)

sol2:=proc(x_rkf45) ... end proc

sol2(3)

[t = 3.0000000000, i__L(t) = 1.0389096668, v__C(t) = 0.4824825878]

plots:-odeplot(sol2, [[t, v__C(t)], [t, i__L(t)]], t = 0 .. 10);



[例2] 求解方程 $\frac{dy}{dx} = \frac{3\ x^2 + 4\ x + 2}{2\ (y-1)}$

```
restart:     #清除内存
eqn := diff(y(x),x) = (3*x^2+4*x+2)/(2*(y(x)-1));       #定义微分方程
sol := dsolve(eqn, y(x));     #解微分方程
rhs(sol[1]);     #取解的右边部分
```

$$1 - \sqrt{1 + x^3 + 2\ x^2 + 2\ x + _C1}$$

sol 是方程的通解，我们定义的微分方程是一阶的，所以在通解中存在一个常数_C1。

如果给定一个初始条件 y(0)=-1，我们可以得到特解。

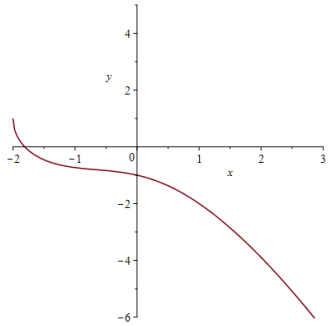
```
Sol0 := dsolve({eqn, y(0)=-1}, y(x));     #求初始条件下的特解
```

$$sol0 := y(x) = 1 - \sqrt{4 + x^3 + 2\ x^2 + 2\ x}$$

可以用 plot 命令画出特解的图形。调用格式是：

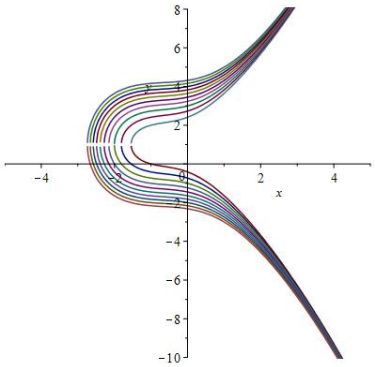
plot(sol, x=a..b, y=c..d)　画出特解 sol 在 x 从 a 到 b，y 从 c 到 d 范围内的图形。

```
Plot(rhs(sol0), x=-2..3, y=-6..5);       #rhs 命令表示取特解 sol0 右侧的表达式
```



如果把通解中的常数代入不同的值并绘图，将得到微分方程的积分曲线族的图形。

```
Sol2 := seq(subs(_C1=c, rhs(sol[1])), c=1..10):       #生成列表形式的解（c=1..10）
sol3 := seq(subs(_C1=c, rhs(sol[2])), c=1..10):       #生成列表形式的解（c=1..10）
plot([sol2,sol3], x=-5..5, y=-10..8);     #绘图
```

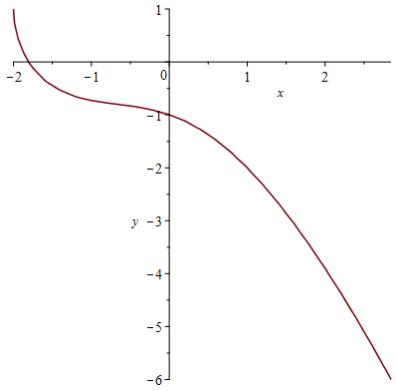


Maple 的 dsolve 命令通常会返回微分方程的显式解。但如果显式解不易阅读或者比较复杂，可以使用 implicit 参数项告诉 Maple 用隐式解来表示。

[例3] 求解方程$\frac{dy}{dx} = \frac{3\ x^2 + 4\ x + 2}{2\ (y-1)}$的隐式解。

```
Restart:     #清除内存
```

```
eqn := diff(y(x),x)= (3*x^2+4*x+2)/(2*(y(x)-1));        #定义微分方程
dsolve(eqn, y(x), implicit);        #求隐式通解
sol0 := dsolve({eqn, y(0)=-1}, y(x), implicit);        #求隐式通特解
plots:-implicitplot(sol0, x=-2..3, y=-6..1);        #绘图
```



6.5 求解常微分方程组 ODEs

这里，我们将求解两个微分代数方程的问题，第一个问题仅调用 dsolve/numeric 命令，使用默认的 DAE 求解器；第二个问题指定 Mebdfi 方法。

```
De:=diff(y(t),t,t) + diff(y(t),t) + y(t)=sin(t),
      diff(x(t),t,t) + diff(y(t)-x(t),t)+x(t)=4;        #定义两个微分方程
```

$$de := \frac{d^2}{dt^2}y(t) + \frac{d}{dt}y(t) + y(t) = \sin(t),\ \frac{d^2}{dt^2}x(t) + \frac{d}{dt}y(t) - \left(\frac{d}{dt}x(t)\right) + x(t) = 4$$

```
ic := D(y)(0)=1,D(x)(0)=4,y(0)=5,x(0)=6;        #定义初始条件
```

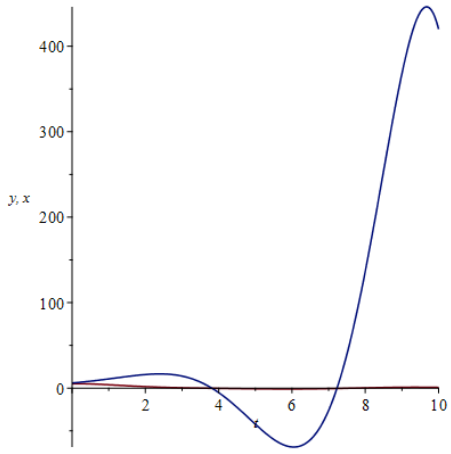
$$ic := D(y)(0) = 1, D(x)(0) = 4, y(0) = 5, x(0) = 6$$

```
sol:=dsolve({de,ic},numeric);        #求数值解
```

sol:=proc(x_rkf45) … end proc

$$ic := D(y)(0) = 1, D(x)(0) = 4, y(0) = 5, x(0) = 6$$

```
plots:-odeplot(sol,[[t,y(t)],[t,x(t)]],t=0..10);        #求数值解
```



第二个问题使用指定的数值算法求解 DAE：物体相互作用的范例。

首先我们调用 PDEtools[declare]，用撇号(')表示微分。

```
Restart:      #清除内存
with(plots):      #加载绘图函数包
PDEtools[declare](prime = t);      #微分符号使用'表示
```

这里，解是限制在平面 a(t) + b(t) + c(t) = 1 上。

```
Sys := {a(t)+b(t)+c(t) = 1, a(0) = 1, b(0) = 0, diff(a(t), t) = -(1/10)*a(t)+10000*b(t)*c(t),
diff(b(t), t) = (1/10)*a(t)-10000*b(t)*c(t)-10000000*b(t)^2}      #定义微分方程和初始条件，{}表示集合形式
```

$$sys := \left\{a(t) + b(t) + c(t) = 1,\ a(0) = 1,\ b(0) = 0,\ a' = -\frac{1}{10}\,a(t) + 10000\,b(t)\,c(t),\ b' = \frac{1}{10}\,a(t) - 10000\,b(t)\,c(t) - 10000000\,b(t)^2\right\}$$

默认的数值求解器在期望的范围内失败。

```
Dsolve(sys, range = 0 .. 40000, numeric)      #求数值解
```

Warning, cannot evaluate the solution further right of 6.7974236, maxfun limit exceeded (see ?dsolve,maxfun for details)

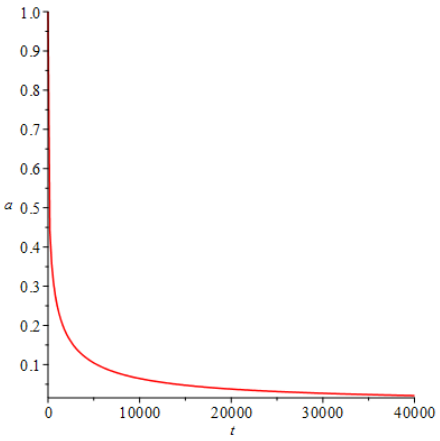
这是三个变量的问题，不能使用 Maple 默认的数值 ODE 求解器求解，但可以通过指定的 DAE 求解器求解。

现在，我们可以使用 Modified Extended Backward-Differentiation Implicit method (mebdfi) 的 DAE 求解器。

```
DAESolution := dsolve(sys, numeric, method = mebdfi);      #求数值解
```

DAESolution := proc(x_mebdfi) … end proc

```
odeplot(DAESolution, [t, a(t)], 0 .. 40000);       #绘图
```



6.6 求解偏微分方程 PDE

6.6.1 一个简单的例子

```
eq := diff(u(x, t), t) + c*diff(u(x, t), x) = -lambda*u(x, t);     #定义偏微分方程
```

$$eq := \frac{\partial}{\partial t} u(x, t) + c \left(\frac{\partial}{\partial x} u(x, t) \right) = -\lambda u(x, t)$$

```
sol := pdsolve(eq);
```

$$sol := u(x, t) = _F1\left(\frac{t c - x}{c} \right) e^{-\frac{\lambda x}{c}}$$

```
bc := u(x, 0) = phi(x);

sol := pdsolve([eq, bc]);
```

$$sol := u(x, t) = \phi(-c t + x)\, e^{-\lambda t}$$

6.6.2 求解热传导方程

数值解：

```
pde := diff(u(x, t), t) = diff(u(x, t), x, x);
```

$$pde := \frac{\partial}{\partial t} u(x, t) = \frac{\partial^2}{\partial x^2} u(x, t)$$

初始条件：

```
ibc := [(D[1](u))(0, t)+a*u(0, t) = h(t), u(x, 0) = 0];
```

$$ibc := \left[\mathrm{D}_1(u)(0,t) + a\,u(0,t) = h(t),\ u(x,0) = 0\right]$$

可以得到数值解，但是我们需要定义 a 和 h 的值，以及提供第二个边界条件：

```
ibc1:=[D[1](u)(0,t)+2*u(0,t)=cos(t),u(x,0)=0,u(0,t)=1];
```

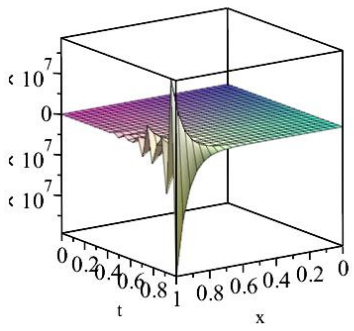
$$ibc1 := \left[\mathrm{D}_1(u)(0,t) + 2\,u(0,t) = \cos(t),\ u(x,0) = 0,\ u(0,t) = 1\right]$$

```
pds:=pdsolve(pde,ibc1,numeric,time=t,range=0..1);
```

pds:=module() ... end module

可以做其他计算，例如生成解的三维图形：

```
pds:-plot3d(t=0..1,x=0..1,axes=boxed);
```



解析解：

我们可以使用 pdsolve （无需参数项）命令计算偏微分方程的解析解：

```
gsol:=pdsolve(pde);
```

$$gsol := u(x,t) = _F1(x)\,_F2(t) \ \textbf{where}\ \left[\left\{\frac{\mathrm{d}}{\mathrm{d}t}\,_F2(t) = _c_1\,_F2(t),\ \frac{\mathrm{d}^2}{\mathrm{d}x^2}\,_F1(x) = _c_1\,_F1(x)\right\}\right]$$

结果可以拆分为方程（eq）和条件（conds）：

```
eq:=op(gsol)[1];

conds:=op(op(gsol)[2]);
```

$$eq := u(x,t) = _F1(x)\,_F2(t)$$

$$conds := \left\{\frac{\mathrm{d}}{\mathrm{d}t}\,_F2(t) = _c_1\,_F2(t),\ \frac{\mathrm{d}^2}{\mathrm{d}x^2}\,_F1(x) = _c_1\,_F1(x)\right\}$$

可以用 dsolve 命令求解常微分方程组，然后使用结果对 eq 求值：

```
conditions:=dsolve(conds);
solution:=eval(eq,conditions);
```

$$conditions := \left\{ _F1(x) = _C1\, e^{\sqrt{-c_1}\, x} + _C2\, e^{-\sqrt{-c_1}\, x}, _F2(t) = _C3\, e^{-c_1 t} \right\}$$

$$solution := u(x, t) = \left(_C1\, e^{\sqrt{-c_1}\, x} + _C2\, e^{-\sqrt{-c_1}\, x} \right) _C3\, e^{-c_1 t}$$

我们可以判定结果是否满足偏微分方程：

```
pdetest(solution,pde);
```

0

然后我们可以调整常数 _c1, _C1, _C2 和 _C3 满足期望的条件。

6.6.3 求解波动方程

Maple 求解经典力学难题的能力是非常著名的，它的数值和符号偏微分方程求解器是其中的重要工具。

例子：在不同的边界条件下，求波动方程的数值解、解析解、和图形解。

```
restart;
with(plots);
```

下面的 Maple 代码定义了一个名为 PX 的程序，生成函数的周期展开。

```
PX := proc(h::{algebraic,procedure},g::{range,name=range})
    local L, D, var;
    if type(g,'range') then L := lhs(g); D := rhs(g) - L;
    if not type(h,'procedure') then var := indets(h,'name');
    if nops(var) <> 1 then error "need to specify a variable"; end if;
    var := op(var); end if;
    else L := lhs(rhs(g));D := rhs(rhs(g)) - L;var := lhs(g); end if;
    if type(h,'procedure') then proc(x::algebraic) h(x - floor((x-L)/D)*D); end;
    else proc(x::algebraic) eval(h, var = x - floor((x-L)/D)*D); end;
    end if;
end:
```

数值解和图形解

一个空间变量的波动方程是：

```
pde := diff(u(x, t), t, t) = diff(u(x, t), x, x);
```

$$pde := \frac{\partial^2}{\partial t^2} u(x, t) = \frac{\partial^2}{\partial x^2} u(x, t)$$

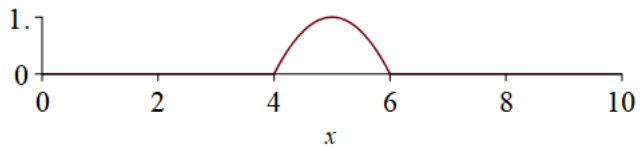
假设初始形状由下面的函数给出：

```
f := x -> piecewise(x < 4, 0, x < 6, (x - 4)*(6 - x), 0);
```

$$f := x \mapsto \begin{cases} 0 & x < 4 \\ (x-4)\cdot(6-x) & x < 6 \\ 0 & otherwise \end{cases}$$

对应的图形如下：

```
plot(f(x), x = 0 .. 10, scaling = constrained, tickmarks = [6, 2])
```



假设波动方程的初始条件和边界条件如下：

```
ibc := [u(0, t) = 0, u(10, t) = 0, u(x, 0) = f(x), D[2](u)(x, 0) = 0];
```

$$ibc := \left[u(0, t) = 0,\ u(10, t) = 0,\ u(x, 0) = \begin{cases} 0 & x < 4 \\ (-4 + x)\ (6 - x) & x < 6 \\ 0 & otherwise \end{cases},\ \mathrm{D}_2(u)(x, 0) = 0 \right]$$

Maple 中的命令 pdsolve 将求解单变量演化方程（双曲和抛物）的数值解（有限差分）。

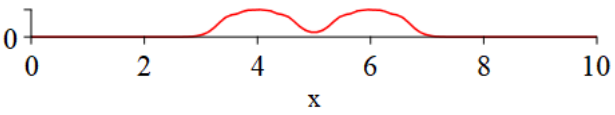
```
SOL := pdsolve(pde, ibc, numeric, time = t, range = 0 .. 10, spacestep = 0.1)
```

SOL:=module() ... end module

这个命令创建了一个模块，可以看到模块的输出函数是 plot, plot3d, animate,和 value。

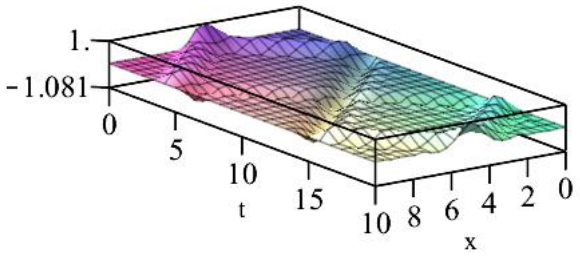
时间 t = 1 的图形：

```
SOL:-plot(t = 1, numpoints = 100, scaling = constrained, tickmarks = [6, 2], title = "图 2")
```



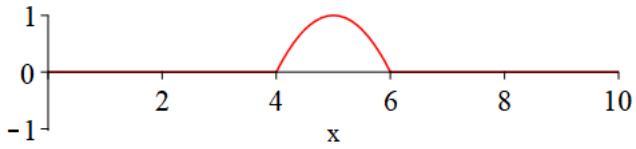
解的表面图：

SOL:-plot3d(t = 0 .. 20, x = 0 .. 10, axes = box, scaling = constrained, tickmarks = [6, 5, 2], title = "图 3");



动画：

SOL:-animate(t = 20, frames = 60, scaling = constrained, tickmarks = [6, 3], title = "time = %f");



给出计算值 U(x, t)的函数：

U := rhs(SOL:-value(output = listprocedure)[3]);

U:=proc() ... end proc

例如，U(2.5, 3)的解：

U(2.5, 3);

0.364667328524451

D'Alembert 解析解:

对于这个问题，D'Alembert 解的形式是：

$$u(x,\ t) = \frac{f(x+t) + f(x-t)}{2}$$

因此 u(2.5, 3)的精确值是：

(f(2.5 + 3) + f(2.5 - 3))/2;

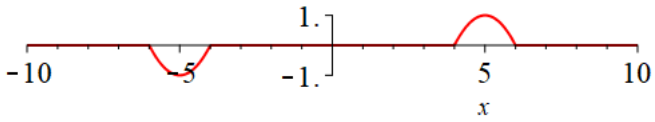
0.3750000000

在 f(x)区间[-10, 10]上的奇次展开如下：

FOE := simplify(piecewise(x < 0, -f(-x), x < 10, f(x)));

$$FOE := \begin{cases} 0 & x \le -6 \\ (4+x)\ (6+x) & x \le -4 \\ 0 & x < 4 \\ -(-4+x)\ (-6+x) & x < 6 \\ 0 & 6 \le x \end{cases}$$

plot(FOE, x = -10 .. 10, color = red, thickness = 2, scaling = constrained, ytickmarks = 2, title = "图 4")

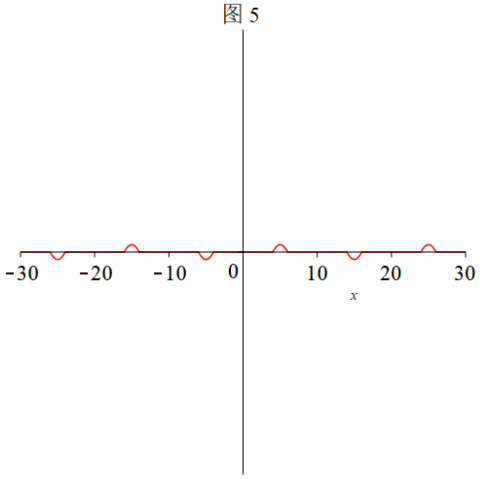


f(x)的奇次周期展开如下：

F := PX(FOE, x = -10 .. 10);

我们注意到输出是一个过程程序，忽略 Maple 输出的细节，我们再次使用图形来验证工作，得到[图 5]。

plot(F, -30 .. 30, color = red, scaling = constrained, tickmarks = [6, [0]], labels = [x, ``], labelfont = [TIMES, ITALIC, 12], view = [-30 .. 30, -30 .. 30], title = "图 5")

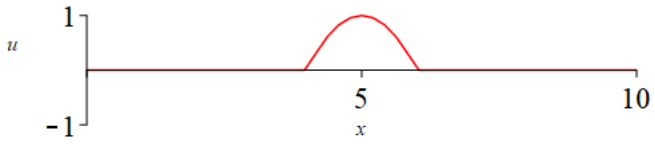


最后，通过下式组合解 u(x, t) = 1/2*F(x + t) + F(x - t)

UDAL := (F(x + t) + F(x - t))/2；

U 的输出是一个分段函数的复杂组合，但是下面的动画显示了在指定区间内波的相应运动。

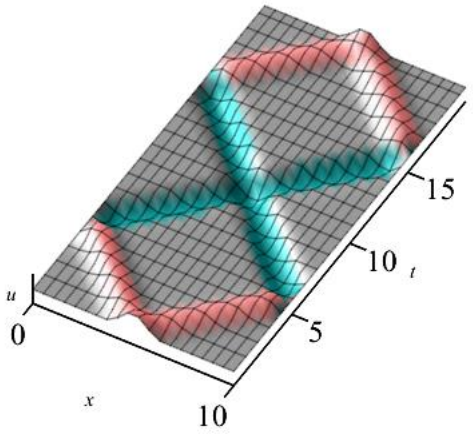
animate(UDAL, x = 0 .. 10, t = 0 .. 20, frames = 61, color = red, scaling = constrained, tickmarks = [3, 2], labels = [x, `u`], labelfont = [TIMES, ITALIC, 12]);



【图 6】是解的表面，与动画提供的信息一致。

plot3d(UDAL, x = 0 .. 10, t = 0 .. 20, axes = frame, color = [(1 + signum(UDAL))/2, 0.5, 0.5], labels = [x, t, u], tickmarks = [2, [5, 10, 15], 0], grid = [20, 40], orientation = [-60, 45], scaling = constrained, labelfont = [TIMES, ITALIC, 12], title = "图 6");

图 6



传播和 Klein-Gordon 方程:

数值解:

波动和 Klein-Gordon 传播方程分别表示为：

wave := pde;

$$wave := \frac{\partial^2}{\partial t^2} u(x, t) = \frac{\partial^2}{\partial x^2} u(x, t)$$

KG := diff(u(x, t), t, t) = diff(u(x, t), x, x) - u(x, t)

$$KG := \frac{\partial^2}{\partial t^2} u(x, t) = \frac{\partial^2}{\partial x^2} u(x, t) - u(x, t)$$

两者之间的区别是 Klein-Gordon 方程中有 u(x, t)项。

为了说明两个方程解之间的查表，考虑下面的初始扰动：

L := 40;

f := piecewise(x < L/2 - 1, 0, x < L/2 + 1, (x + (-L/2 + 1))*(L/2 + 1 - x), 0);

$$f := \begin{cases} 0 & x < 19 \\ (-19 + x)\ (21 - x) & x < 21 \\ 0 & otherwise \end{cases}$$

传播长度设定为 20，端点固定。初始条件和边界条件的方程是：

ibc := [u(0, t) = 0, u(L, t) = 0, u(x, 0) = f, D[2](u)(x, 0) = 0]

$$ibc := \left[u(0,\ t) = 0,\ u(40,\ t) = 0,\ u(x, 0) = \begin{cases} 0 & x < 19 \\ (-19 + x)\ (21 - x) & x < 21 \\ 0 & otherwise \end{cases},\ \mathrm{D}_2(u)(x, 0) = 0 \right]$$

方程的数值解分别是：

SW := pdsolve(wave, ibc, numeric, time = t, spacestep = 0.05);

SKG := pdsolve(KG, ibc, numeric, time = t, spacestep = 0.05);

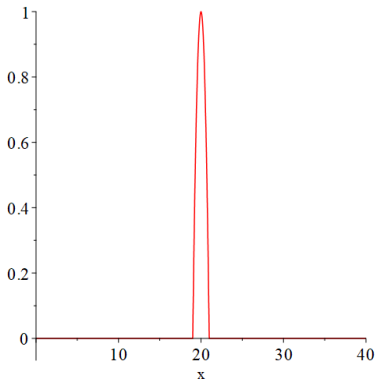
方程的动画分别是：

p1 := SW:-animate(x = 0 .. L, t = 0 .. L/2, frames = 50, color = red);

p2 := SKG:-animate(x = 0 .. L, t = 0 .. L/2, frames = 50, color = blue);

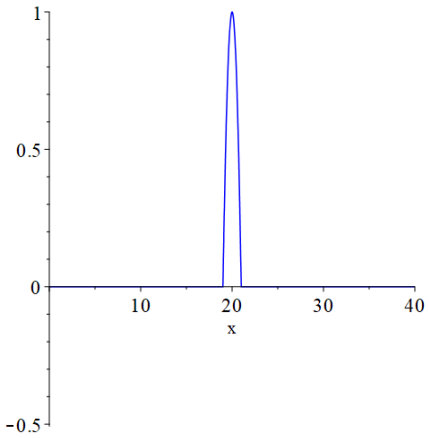
波动方程的动画是：

p1;



Klein-Gordon 传播方程的动画是：

p2;



解析解：

Klein-Gordon 方程产生分离的变量。因此，假设解的形式如下：

U := X(x)*T(t);

因此 Klein-Gordon 方程，这里 c = 1，变为：

q1 := eval(KG, u(x, t) = U);

$$q1 := X(x)\left(\frac{d^2}{dt^2}T(t)\right) = \left(\frac{d^2}{dx^2}X(x)\right)T(t) - X(x)\,T(t)$$

除以 u(x, t) = X(x)*T(t)得到：

q2 := expand(q1/U);

$$q2 := \frac{\frac{d^2}{dt^2}T(t)}{T(t)} = \frac{\frac{d^2}{dx^2}X(x)}{X(x)} - 1$$

-1 移到左边：

q3 := q2 + (1 = 1);

$$q3 := \frac{\frac{d^2}{dt^2}T(t)}{T(t)} + 1 = \frac{\frac{d^2}{dx^2}X(x)}{X(x)}$$

基于以前的经验，我们引入 Bernoulli 分离常数$-\lambda^2$，这两个常微分方程变为：

q4 := rhs(q3) = -lambda^2;

$$q4 := \frac{\frac{d^2}{dx^2}X(x)}{X(x)} = -\lambda^2$$

q5 := lhs(q3) = -lambda^2

$$q5 := \frac{\frac{d^2}{dt^2} T(t)}{T(t)} + 1 = -\lambda^2$$

在不同的假设下，得到边界条件：

$$X(0) = X(l) \quad = 0$$

使用其中的第一个得到：

```
dsolve({q4, X(0) = 0}, X(x));
```

$$X(x) = _C1 \sin(\lambda x)$$

使用其中的第二个得到：

```
Xn := sin(n*Pi*x/l);
```

$$Xn := \sin\left(\frac{n \pi x}{l}\right)$$

第二个常微分方程变为：

```
q6 := eval(q5, lambda = n*Pi/l);
```

$$q6 := \frac{\frac{d^2}{dt^2} T(t)}{T(t)} + 1 = -\frac{n^2 \pi^2}{l^2}$$

初始条件 u[t](x, 0) = 0 得到条件 T'(0) = 0，然后得到解。

```
q7 := rhs(dsolve({q6, D(T)(0) = 0}, T(t)));
```

$$q7 := _C2 \cos\left(\frac{\sqrt{\pi^2 n^2 + l^2}\ t}{l}\right)$$

可以写为：

```
Tn := eval(q7, _C2 = 1);
```

$$Tn := \cos\left(\frac{\sqrt{\pi^2 n^2 + l^2}\ t}{l}\right)$$

使用常用的技术，前一段落中的 BVP 的数值解可以表示为级数：

$$u(x, t) = \sum_{n=1}^{\infty} b_n \sin\left(\frac{n \pi x}{l}\right) \cos\left(\sqrt{1 + \left(\frac{n^{\pi}}{l}\right)^2}\ t\right)$$

这里系数 b[n]由下面给出：

```
qb := eval(2*Int(f*Xn, x = 0 .. l)/l, l = L);
```

$$qb := \frac{\left(\int_0^{40} \left(\begin{cases} 0 & x < 19 \\ (-19 + x)\ (21 - x) & x < 21 \\ 0 & otherwise \end{cases} \right) \sin\left(\frac{n\,\pi\,x}{40} \right) dx \right)}{20}$$

最终得到：

```
b := (value(qb) assuming n::integer);
```

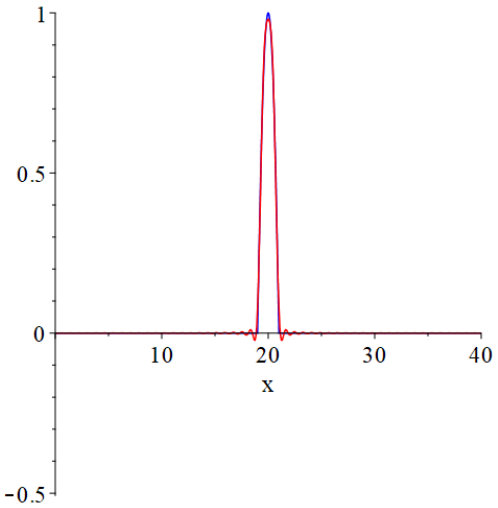
$$b := -\frac{160\left(n\,\pi \sin\left(\frac{21\,n\,\pi}{40} \right) + n\,\pi \sin\left(\frac{19\,n\,\pi}{40} \right) + 40\cos\left(\frac{21\,n\,\pi}{40} \right) - 40\cos\left(\frac{19\,n\,\pi}{40} \right) \right)}{n^3\,\pi^3}$$

得到一个近似部分和：

```
Uxt := eval(sum(b*Xn*Tn, n = 1 .. 100), l = L);
```

叠加解后的动画：

```
p3 := animate(Uxt, x = 0 .. L, t = 0 .. L/2, frames = 50, color = red, numpoints = 500);
display([p2, p3]);
```



可以发现数值解和级数近似值几乎等同。

第七章 创建计算书

Maple 可以编写专业的计算书，透过智能文件界面提供完整的数值和符号计算，可以轻松管理您的计算，让您的计算成为重要的资产。在 Maple 计算书中，您可以将数学、文字、图像、图形等内容组合在一个文档中，这样就可以将所有的假设、推理、见解和结论与计算一起记录下来，以便在需要时随时可用。

Maple 计算管理环境可以帮助您最大限度地提高计算工作的价值。使用 Maple，您可以轻松验证、文档化、重用和修改计算、生成可执行项目文件、GUI 图形用户界面开发、文件和代码加密、应用部署等，从而降低风险，节省时间和精力，保护计算资产。

Maple 内置大量的文字处理工具，用于创建专业外观的报告。这里列举了其中的一些工具。

特征	位置
样式	工具栏列表 
章节	插入 > 分节，或者工具栏 
表格	插入 > 表格
样式控制和定义新的样式	工具栏，以及菜单格式 > 样式...
插入图片	插入 > 图像
数学术语拼写检查	工具 > 拼写检查
超链接和书签	插入 > 超链接 格式 > 书签...
页码，页眉，页脚	插入 > 页眉和页脚
输出为 PDF	文件 > 输出...

7.1 样式

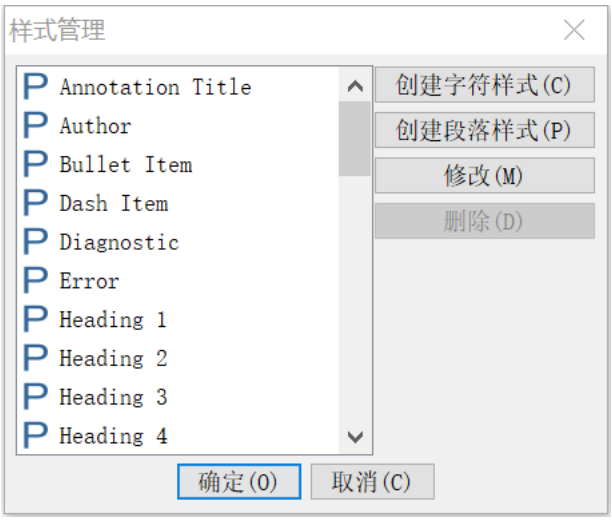
设置默认字体、段落等样式等。我们可以通过工具栏调整 Maple 文件中的字体、大小、颜色、段落、对齐方式等。



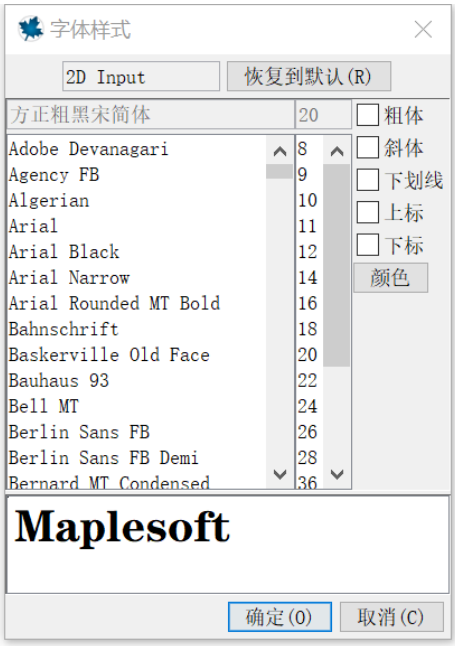
我们经常希望设置一个默认字体、段落等样式，当我们创建一个新的 Maple 文件时系统可以自动采用这些样式配置，或者对已有的文件应用这些样式配置。

修改默认的样式，步骤如下：

1) 打开菜单“格式(R) → 样式(S)...”。



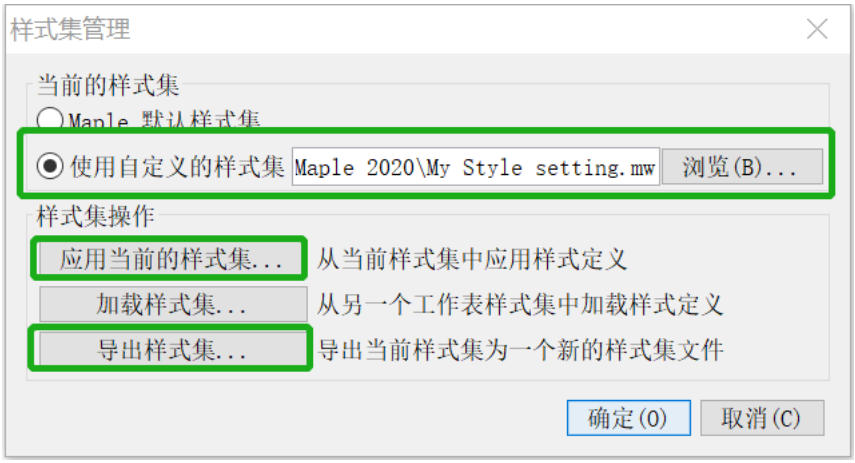
2) 选中其中要修改的项，例如，这里我们选中“2D Input”，然后点击“修改”按钮，在弹出的窗口中修改其样式。以此方法可以设置其他的样式，我们常用的样式有：2D Math，2D Output 和 Text，可以根据自己的洗好或要求进行设置。最后点击确定按钮，退出对话框并保存设置。



注意：当前对 2D Math 格式的设置仅对当前 Maple 文件适用。如果打开新的 Maple 文件，会发现 Maple 文件自动更改为软件原有默认的格式设置。

如果希望将自己所有的文件都采用同一个样式，或者预定义几个样式，如何实现？可以使用“Maple Style Template”，创建样式模板的步骤如下：

3) 打开菜单“格式(R) → 管理样式集(M)...”。



点击“导出样式集...”将当前的配置输出为“Maple Style Template (.mw)”。将该样式模板保存在本地电脑中的任意位置。

4) 指定 Maple 系统的默认样式，让所有的 Maple 文件都采用该设置。

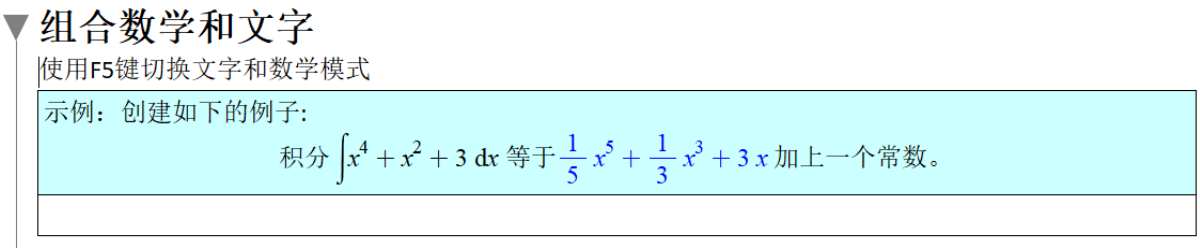
选中上图“样式集管理”窗口中的“使用自定义的样式集”，选择前面创建的样式模板文件。最后点击确定按钮。

依据上面的方法，可以设置默认字体大小、颜色、段落等，并保存到当前文件或者样式模板。

7.2 创建文件

7.2.1 输入数学和文字

使用 F5 键切换文字和数学输入模式。如下图的示例：



步骤如下：

1. 点击工具栏上的章节，章节名称为：组合数学和文字

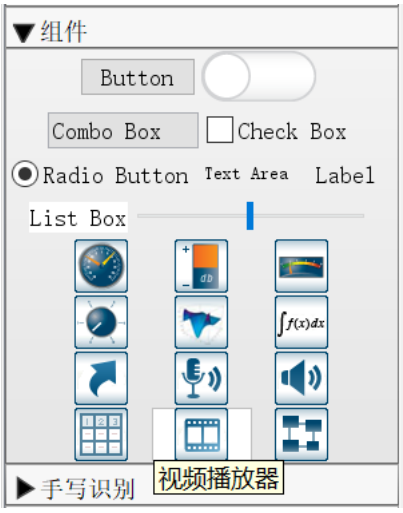
2. 在下面的章节内输入：使用 F5 键切换文字和数学模式
3. 点击菜单插入 > 表格，插入 2x1 表格。
4. 在表格第一行中输入：示例：创建如下的例子:
5. 回车后另起一行输入：积分$\int x^4 + x^2 + 3\,dx$
6. 注意，输入积分可以使用左侧微积分面板中的模板，输入完成后按 CTRL+=，$\int x^4 + x^2 + 3\,dx = \frac{1}{5}x^5 + \frac{1}{3}x^3 + 3x$，将表达式中的“=”替换为中文“等于”，$\int x^4 + x^2 + 3\,dx$ 等于 $\frac{1}{5}x^5 + \frac{1}{3}x^3 + 3x$
7. 继续输入中文：加上一个常数。设置为居中对齐。
8. 修改积分表达式中的任一个参数，例如 4 改为 3 然后点击工具栏上的图标!，可以看到结果将自动更新。

2.2.2 插入图片

你可以插入图片到 Maple 文件中，操作方式是菜单插入 > 图像。支持的图像类型有：gif, jpe, jpeg, jpg, png, bmp, tif, tiff, jfx, pnm, fpx.

7.2.3 插入视频

你可以插入视频到 Maple 文件中，例如插入培训视频。操作方式：打开左侧面板“组件”。

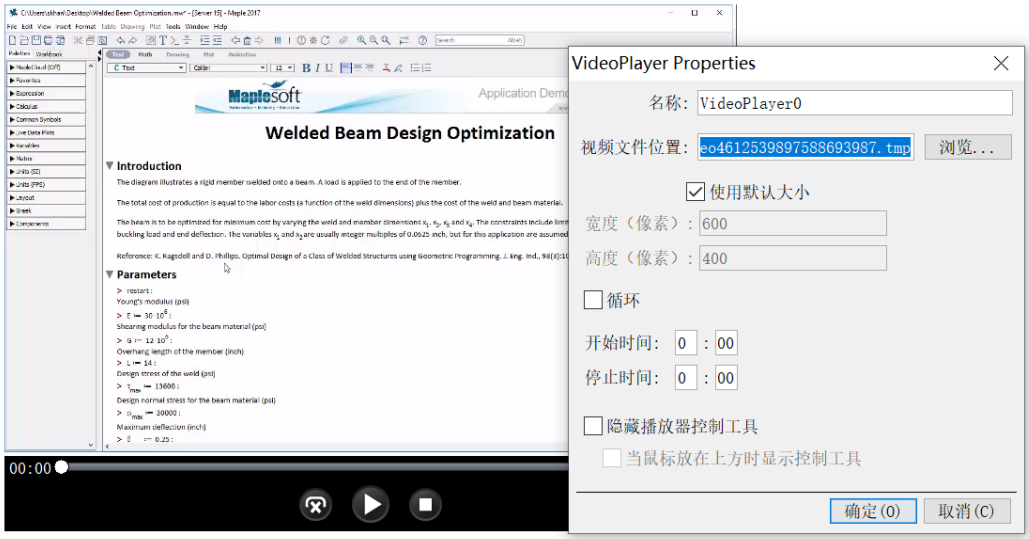


鼠标左键按住其中的“视频播放器“，然后拖放到文件中。鼠标右键单击该视频播放器组

件，从弹出的邮件菜单中选择“GUI 组件属性”。



在视频文件位置中选择视频文件，也可以做其他相关的设置。

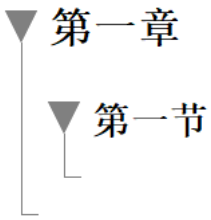


7.2.4 章节管理

Maple 通常通过节和表格来管理文件的组织结构。

- 组织文件内容
- 隐藏散乱的代码和详细信息

插入节的途径：菜单插入 > 节，或者点击工具栏图标 。

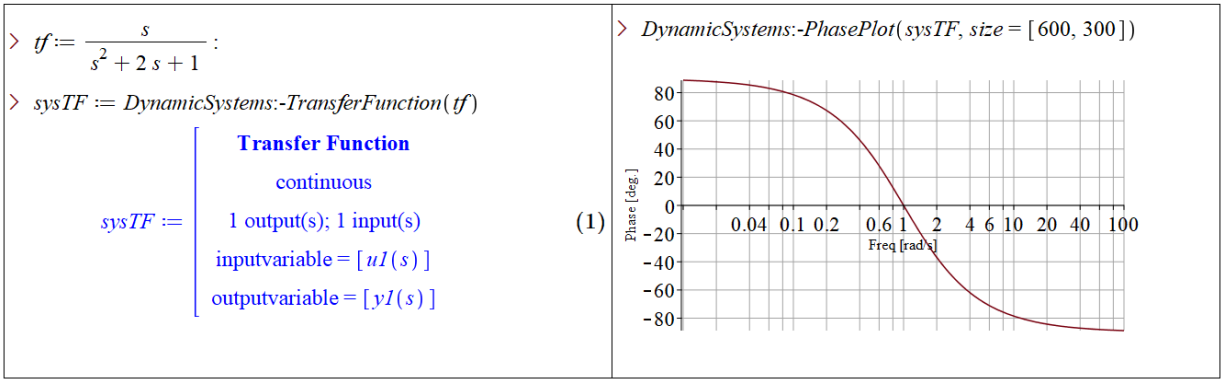


7.2.5 插入表格

表格提供了一种方式组织和管理科学计算、说明性文字和数学、图形和图片、以及 GUIGUI 组件等。操作方式是通过菜单插入 > 表格。

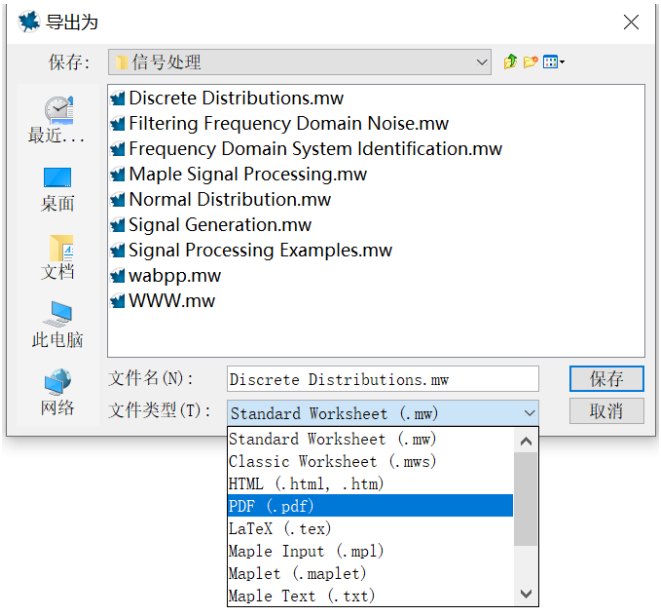
表格的主要用途：

- 有效地组织内容，提高可读性，并减少空间浪费
- 对齐文字和图形
- 控制 GUIGUI 组件的布局



7.2.6 输出为 PDF 文件

Maple 创建的文件可以另存为 PDF 文件，途径是菜单文件 → 输出 → 选择 PDF 格式文件。



输出的 PDF 有两个相关的设置：输出为可搜索文件的格式，以及输出为形状文件。途径是菜单工具 → 选项 → 导出选项卡 → 输出到 PDF 格式。



7.3 Maple Workbook 管理项目中的多个文件

使用 Maple Workbook 工作薄格式，可以收集：

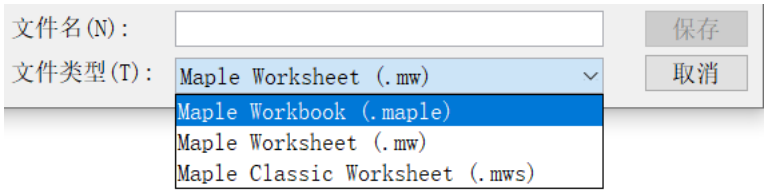
- Maple 工作包文件，库文件，程序文件

- 数据文件（例如图片和电子表格文件）
- 其他（例如 Word, PPT 等）



7.3.1 保存为 Maple Workbook 工作薄文件格式

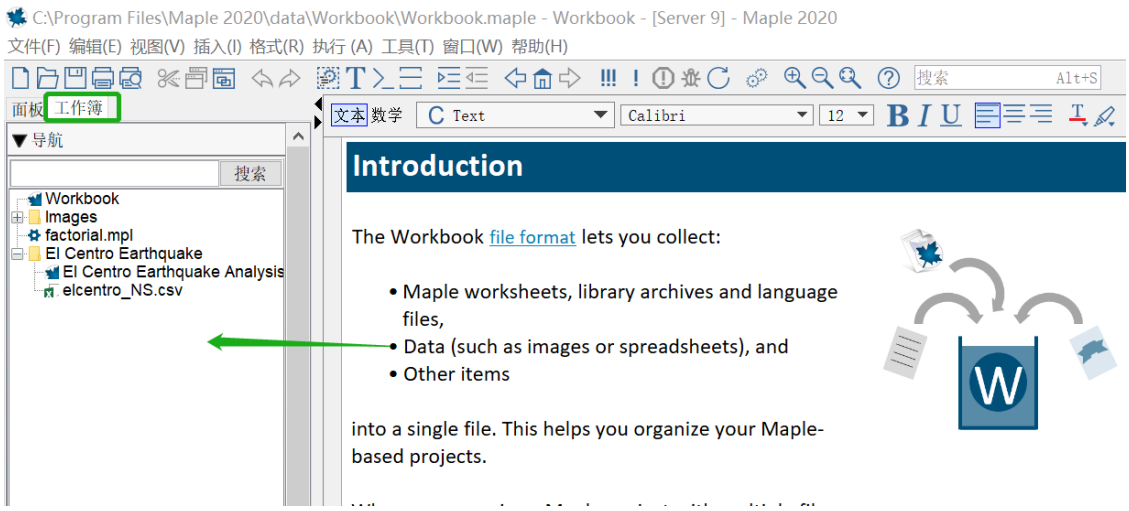
新建或者打开已有的 Maple 文件，然后从文件菜单选择另存为…。存储文件时文件类型选择 Maple Workbook (.maple)。



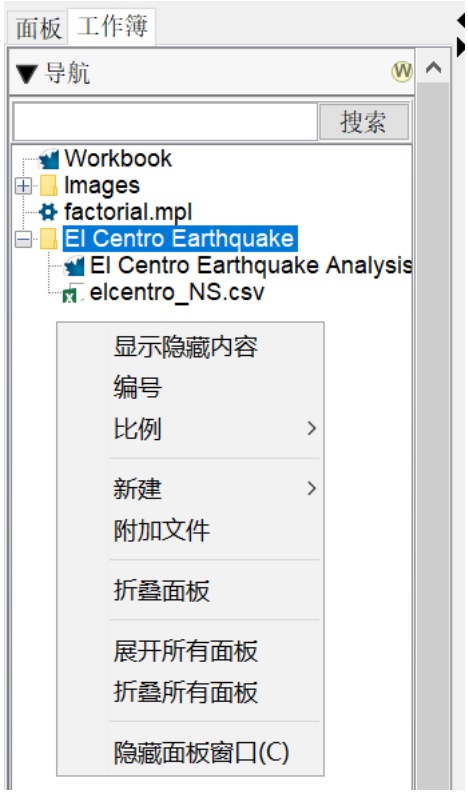
将多个文件放到 Workbook 中保存有很多好处，例如，你只需要跟踪单个文件，而不是多个文件，这样可以消除在文件位置转移时丢失文件依赖性的风险。可以在 Workbook 内部访问和操作所有附带的数据文件，而不是引用其他文件路径。

7.3.2 导航面板

Maple 软件窗口左侧的导航面板显示了 Workbook 中的文件和结构。双击其中的文件可以打开。



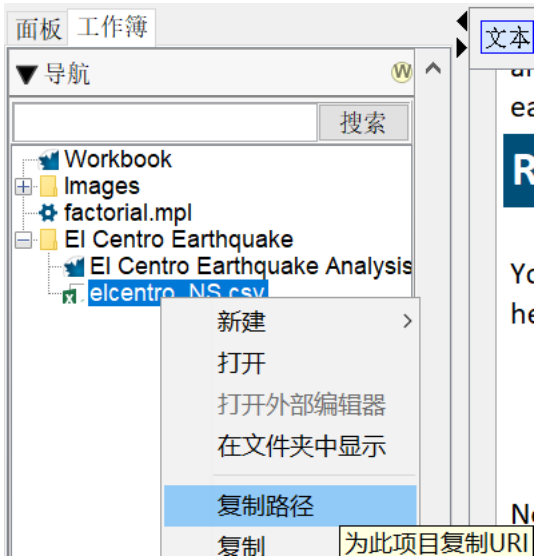
鼠标右键点击导航面板的任意位置或者文件，可以完成各种关联操作，例如新建、附加文件、删除、复制文件路径（相对路径）等操作。



7.3.3 从 Workbook 读取文件

可以读取 Workbook 工作簿附带的数据文件。例如，这里我们读取 CSV 文件中的数据。

打开导航面板，点击其中的 csv 文件，这里是 elcentro_NS.csv，从右键菜单选择“复制路径”。



数据读取命令：

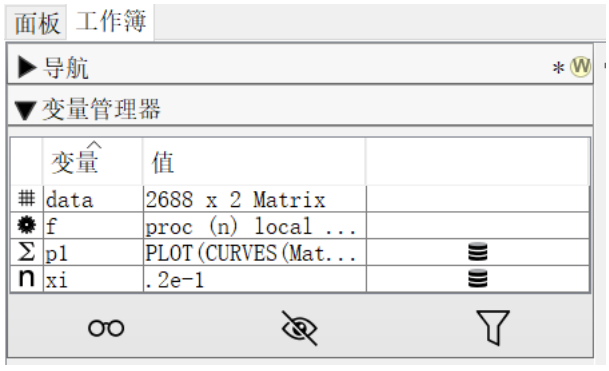
```
[> data := ImportMatrix("this:///El Centro Earthquake/elcentro_NS.csv")
```

其中 this:///是 Workbook 的相对路径。

7.3.4 变量管理器

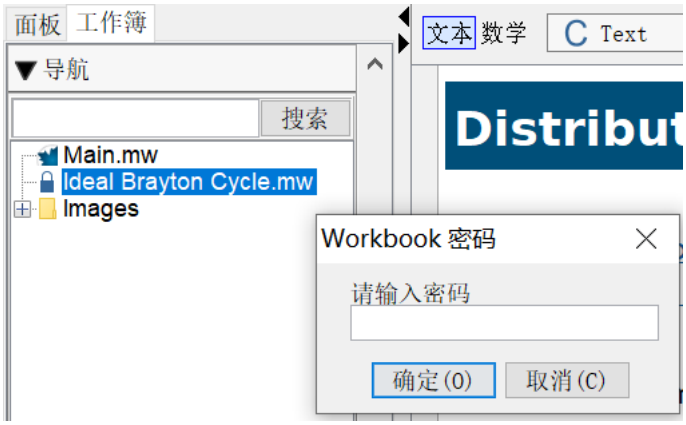
工作薄面板下面含有一个变量管理器，列出了当前文件中的变量名以及相应的类型和值。要查看变量的值，双击相应的项即可。

可以保存变量，这样 Workbook 下的不同文件可以共享变量。要共享变量，只需要右键点击该变量，然后选择保存。



7.4 文件内容加密

你可以使用密码保护 Maple 工作表、文档、代码、应用程序。对于受密码保护的文件，可以将参数传递到受保护的内容中并运行其代码，然后返回结果。



调用 Maple 工作表并返回结果的例子：

[> DocumentTools:-RunWorksheet("this:///Ideal Brayton Cycle.mw", [Tmin = 300*Unit('degC'), Tmax = 1300*Unit('degC'), Pmin = 101*Unit('kPa'), r = 8])

0.4086652304

7.5 分享 Maple 文件

你可以将创建的 Maple 文件分享给朋友、同事或者学生等。其他人可以使用 Maple 软件打开，也可以下载免费的 Maple Player 打开。Maple Player 可以打开 Maple 创建的文件，但不可以编辑其中的内容，如果文件中包含 GUI 组件，用户可以运行其中关联的代码，实现类似于可执行程序的目的。



Maple Player 下载地址：

https://www.maplesoft.com/products/maple/MaplePlayer/mapleplay-download.aspx

第八章 编程

8.1 编写代码的几种方式和工具

Maple 提供多种编写代码的方式和工具。



1) 1D 输入

```
[> restart:
[> squaredDiff := proc( L1, L2 )
     add((L1[i]-L2[i])^2, i=1..nops(L1));
   end proc:
```

上面的 squaredDiff 程序是在 1D 输入提示符下输入的文本字符。

可以在文件空白处点击菜单栏图标插入命令行，格式选择文本。使用 Shift + 回车键换行。按下回车键执行 squaredDiff 程序。

2) 2D 输入

一些情况下，您可能希望直观地查看数学结构。2D 数学适用于较短的程序。例如：

$$dist := \mathbf{proc}(x, y)\ evalf\left(\sqrt{x^2 + y^2}\right)\ \mathbf{end\ proc}:$$

```
[> dist(2, 3)
```

3) 代码编辑区

使用<插入>菜单中的<代码编辑区>。代码编辑区提供编程环境，让你可以使用传统的文本输入界面来编写程序，功能包括：

- 语法高亮显示
- 括号匹配
- 命令补全
- 自动文字封装

- 行号显示
- 按回车键产生新的一行
- 自动缩进，按 Tab 键缩进
- 标记警告和语法错误
- 按 Ctrl + E 执行代码

```
squaredDiff := proc( L1::list, L2::list )
     local length1, length2, i;

     length1 := nops(L1);
     length2 := nops(L2);

     if length1 = length2 then
          add((L1[i]-L2[i])^2, i=1..length1);
     else
          error "列表的长度必须相等";
     end if;
end:
```

4) 调用外部文本文件

对于较大的项目，特别是包含许多程序和数据文件的项目，这些文件是由多个开发人员在修订控制系统中维护的，外部文本文件可能是编码的好选择。使用 "read" 命令执行那些文件。在其他章节，我们还将讨论使用 "savelib"，以及使用伪字节码格式存储程序。

```
[> read "this:///高级编程/codefile.mpl"
Executing statements in codefile.mpl ...
Defining squaredDiff2
Running a test ...
```

8.2 编写 Procedures 过程程序

8.2.1 编写一个简单的过程程序 Procedure

Procedure 翻译为过程，或者程序，是 Maple 中常用的编程结构。

编写 Maple Procedure 实际上是非常简单的，一个 Maple Procedure 实际上一组语句的组合。

创建 Maple Procedure 最容易的方式是在一组命令（这些命令也可以单独使用完成计算）前后分别加上 proc(...) 及 end proc 即可。完成定义后，过程可以在同一个 Maple 程序中用不同的参数值重复使用完成相同的计算，类似于 Maple 中的函数。

Maple 中的 Procedure 类似于其他语言（例如 C，Java）中的函数，或者 Pascal 语言中的过程或函数，或者类似于 FORTRAN 和现代版 BASIC 中的子程序。

一个 Maple Procedure 定义语法如下：

```
proc( parameterDeclarations ) :: returnType;
    description shortDescription;
    option optionSequence;
    local localVariableDeclarations;
    global globalVariableDeclarations;
    statementSequence
end proc
```

术语

在讨论 Maple 或其他编程语言中的 procedures 时经常会用到几个术语，其中几个有时可以互换使用，但它们之间的区别是很重要：

- Procedure - 过程。在 Maple 中，一个 procedure 是一个对象，可以通过函数调用被调用，可以传递参数、执行指定的操作、返回结果。一个过程的定义以关键词 proc 开始，以 end proc 结尾封装。
- Function Call - 函数调用，调用格式是 name(arguments) ，函数调用返回的值是过程返回的值。
- Argument - 实参，参数的值。实参是指过程调用具体调用时的参数值。注意一个默认值并不是一个实参。
- Parameter 或者 Formal Parameter - 参数，或形参。形参是在过程中定义中的参数名，获取实参的值。形参名称用于指向过程中的值。
- Actual Parameter - 指形参的值，既不是实参也不是形参。这个术语是指形式参数在过程执行过程中使用的值。这个值来自于一个实参或一个默认值。这里为了完整性引入该术语，但在本章中不会进一步使用，取而代之我们将使用参数的值。

我们由下面的简单程序来看看 Maple 程序的结构：

```
[> plus := proc(x,y)
        x+y;
  end;
```

$$plus := \mathbf{proc}(x, y)\ x + y\ \mathbf{end\ proc}$$

解释：这个程序只有 2 个参数，在程序内部它的名称是 x, y, 这是 Maple 最简单的程序结

构，仅仅在 proc( )和 end 中间加上在计算中需要的一条或者多条命令即可，Maple 会把最后一个语句的结果作为整个子程序的返回结果，这一点需要引起注意。再看下例：

```
[> P:=proc(x,y)
      x-y;
      x*y;
      x+y;
   end:
[> P(3,4);
```

7

显然，尽管程序 P 有三条计算命令，但返回的只是最后一个语句 x+y;的结果. 要想输出所有的计算结果，需要在程序中增加 print 语句：

```
[> P := proc(x,y)
        print(x-y);
        print(x*y);
        print(x+y);
     end:
```

-1

12

7

再看下面几个例子：

```
[> for i from 2 to 6 do
      expand((x+y)^i );
   end do;
```

$$x^2+2xy+y^2$$

$$x^3+3x^2y+3xy^2+y^3$$

$$x^4+4x^3y+6x^2y^2+4xy^3+y^4$$

$$x^5+5x^4y+10x^3y^2+10x^2y^3+5xy^4+y^5$$

$$x^6+6x^5y+15x^4y^2+20x^3y^3+15x^2y^4+6xy^5+y^6$$

```
[> F := proc(n::integer)
```

```
    if n mod 12=0 then true
else false
   end if
end:
[> F(123^123), F(1234567890^9);
```

false, true

从上面几个简单的例子可以看出 Maple 子程序主要包含以下一些内容：

1) 把定义的子程序赋值给程序名 procname，以后就可以用子程序名 procname 来调用程序；
2) 子程序一律以 proc( )开头，括号里是程序的输入参数，如果括号中什么都没有，表示这个子程序没有任何输入参数；
3) 子程序中的每一个语句都用分号(或冒号)分开(这一点不是主要的， 程序设计时， 在可能的时候—过程当中的最后一个语句、for-循环、if 语句中的最后一个语句省略终结标点也是允许的， 这并不是为了懒惰， 而是因为在终结语句后面插入一个语句产生的影响要比仅仅执行一个新语句产生的影响大)；
4) 在定义完子程序之后，Maple 会显示它对该子程序的解释 (除非在 end 后用冒号结束)，它的解释和你的定义是等价的，但形式上不一定完全相同；
5) Maple 会自动地把除了参数以外的变量都作为局部变量(local variable)，这就是说，它们仅仅在这个子程序的定义中有效，和子程序以外的任何同名变量无关。

在定义了一个子程序以后，执行它的方法和执行任何 Maple 系统子程序一样—程序名再加上一对圆括号( )，括号中包含要调用的参数，如果子程序没有参数，括号也是不能省略的。

一些简单的示例和说明

8.2.2 一些简单的示例和说明

说明	示例
程序就是一组 Maple 语句。	f := proc() 3; end proc; f();

	3
除非另有说明，否则 Maple 将返回最后一条语句的结果 — 但是，可能并不总是这样。因此，最好指定要返回的语句。	f := proc(a, b) return a + b; a - b; end proc; f(1, 2); 3
传递参数给程序	f := proc(a, b) return a + b; end proc; f(1, 2);

定义局部变量
如果想要抑制警告信息，使用'local‘声明局部变量
	f := proc(a, b) local c; c := a + b; return c;
	end proc
	Warning, `c` is implicitly declared local to procedure `f`
	f(1, 2);
	3
	f := proc(a, b) local c; c := a + b; return c;
	end proc;
	f(1, 2);
	3

可以在程序声明中指定参数类型	f := proc(a::integer, b::integer) local c; c := a + b; return c; end proc
	f(1, 2);
	3
	f(1.1000000000, 2);
	Error, invalid input: f expects its 1st argument, a, to be of type integer, but received 1.1

不能在程序主体中定义传递参数	f := proc(n) n := 1; end proc;

	f(4); Error, (in f) illegal use of a formal parameter

程序可以有不同的参数类型	f := proc(x::numeric, str::string) cat(str, " = ", x); end proc;
	f(3, "apples");
	"apples = 3"

调用程序时，可以传递更多的参数，但 Maple 会忽略多余的参数，除非有特别声明	f := proc(a, b) return a^b; end proc;
	f(3, 4, 5, 6, 7, 8);
	81

procname 是指调用程序的名称	delphi := proc() return procname; end proc;
	delphi();
	delphi

_params 是指传递给程序的参数列表	f := proc(a, b) print(_passed); end proc;
	f(2, 3, 4, 5);
	2, 3, 4, 5

可以在程序中加入循环语句，例如 for 和/或 while 循环	f := proc(n) local temp, i; temp := 1; for i to n do temp := temp*i; end do; return temp; end proc; f(5); 120
可以在程序中加入 if 语句	f := proc(a, b) if b < a then return a + b; else return a - b; end if; end proc; f(5, 4); 9

8.3 条件和循环语句

8.3.1 程序流控制

与所有编程语言一样，Maple 包含一些语句，允许你通过条件语句和迭代来控制程序流。

条件语句: if...then

Maple 中的条件语句具有以下语法：

if 条件 then 语句 end if;

if 条件 then 语句 else 语句 end if;

if 条件 then 语句 elif 语句 elif 语句 ...else 语句 end if;

```
[> if 条件表达式_1  then
        程序语句_1
   else if 条件表达式_2  then
        程序语句_2
   else if 条件表达式_3  then
        程序语句_3
   ...
   else
        程序语句_N
  end if
```

if 语句会执行哪一个语句的取决于对条件表达式的判断，如果可能，if 语句认为条件表达式是一个布尔量（或真或假），它会尝试把其他数据类型转化为布尔量类型。

条件表达式（条件表达式_1，条件表达式_2 ...）可以是任意的。可以用下面的算子构建布尔表达式：

- 关系算子：<, <=, =, >=, >, <>
- 逻辑算子：and, or, xor, implies, not
- 逻辑名：true, false, FAIL

条件表达式必须具有布尔值 true，false 或者 FAIL，否则将会出现错误。程序语句（程序语句_1，程序语句_2，...，程序语句_N）可以是任意的 Maple 命令语句，包括 if 语句。

示例 1：

```
[> x := 5;
   if x > 3 then
      print("大于 3");
```

```
    end if;
```

x := 5

"大于 3"

示例 2：

```
[> x := 2;
    if x > 3 then
       print("大于  3");
    elif x > 0 then
       print("正数");
    else
       print("负数");
    end if;
```

x := 2

"正数"

在同时包含 elif 和 else 从句的 if 语句中，Maple 会依序对条件表达式求值，直到其中一个返回 true。Maple 执行对应的语句，然后退出 if 语句。如果没有条件表达式返回 true，Maple 会执行 else 从句中的语句。

8.3.2 返回结果: return 和 error 语句

默认情况下，程序返回最后的执行结果。 但是，您也可以通过使用 return 语句强制从程序返回指定的结果。

下面的程序返回内置的值 FAIL，表示计算被意外中断。

```
[> Division := proc(x,y)
        if y=0 then
            return FAIL
        else
            x/y;
        end if:
    end proc:
[> Division(3, 0);
```

FAIL

您还可以使用 error 语句从函数返回。 当遇到 error 语句，返回标准的 Maple 错误（异常）信息。

```
[> Division2 := proc(x,y)
        if y=0 then
            error "除零错误"
        else
            x/y;
        end if:
    end proc:
Division2(2, 0);
```

Error, (in Division2) 除零错误

使用 error 的好处包括：

- 系统会自动通知您哪个程序遇到了错误，这在您有调用其他程序时很有用。
- 以这种方式生成的错误与 Maple 的内置命令生成的错误消息的处理方式相同。这意味着您可以利用 Maple 的错误处理工具，包括 stoperror() 调试命令和异常处理工具。

8.4 迭代: for 循环和 while 循环

for 循环有两种基本形式。 在下面，| | 之间的所有内容是可选的。

| for <name> | | from <expr> | | by <expr> | | to <expr> | | while <expr> |

do <statement sequence> end do;

或者

| for <name> | | in <expr> | | while <expr> |

do <statement sequence> end do;

与大多数其他语言不同，for 循环不是迭代的唯一选择。 在 Maple 中，通常通过使用 seq, map 以及通过单个命令执行数学运算是最好的迭代处理方式。

话虽如此，对于每个程序员来说，for 循环都是必不可少的工具。 在 Maple 中，for 循环非常灵活，并具有不同的选项。

使用循环语句，用户可以重复执行程序语句序列。三种方式实现循环语句：

1. 在一个计数器变量值超过极限值之前 (for/from 循环)
2. 对表达式中的每个运算对象 (for/in 循环)
3. 当一个布尔条件不成立时 (while 循环)

8.4.1 for/from 循环

for/from 循环重复执行一个程序语句序列，直到计算变量值超过极限值。

```
[> for counter from initial by increment to final do
      Statement_sequence;
   end do;
```

其中 initial 是 counter 的初始值，final 是 counter 的终值。increment 是步长。在 for 循环的结构中，可以省略下列任何一部分 for conter，from initial，by increment，to final 都可以省略。

for/from 循环的动作：

1. 将 initial 初始值赋给变量名 counter.
2. 将变量 counter 的值与 final 终值对比，如果 counter 的值超过 final 的值，Maple 退出循环。
3. 执行程序语句 statement_sequence.
4. counter 的值增加 increment 步长值。
5. 重复步骤 2 到 4，直到 Maple 退出循环。

除了 for 部分必须放置于首位外，其余的部分 for，by，和 to 可以按任意的顺序放置。省略的部分有自己的缺省值，见下表：

语句	缺省值
from initial	1
By increment	1
to final infinity	∞

这是一个从 1 到 10 的简单循环：

```
[> for i from 1 to 10 do
        i^2
   end do;
```

示例：

```
[> for x from -5 to 4 by 2 do
      print(x);
   end do;
```

8.4.2 for/in 循环

for 循环还有一种形式，我们称为 for-in 循环。for/in 循环语句重复执行表达式中的每一个元素（运算对象），如列表中的元素。

```
[> for variable in expression do
    Statement_sequence;
   end do;
```

for 从句必须首先出现。这种循环的含义是循环变量 variable 遍历表达式对象 expression 的每一个元素。这个对象可以是列表、集合，也可以是一个和式的项或乘积的项。

for/in 循环的动作：

1. 将表达式 expression 的第一个运算对象赋值给变量名 variable。
2. 执行程序语句 statement_sequence。
3. 将表达式 expression 的下一个运算对象赋值给变量名 variable。
4. 对表达式 expression 中的每个运算对象重复步骤 2 和 3。如果遍历了表达式 expression 所有运算对象，则 Maple 退出循环。

示例：

```
[> nonprimes := remove(isprime, {seq(1 .. 25)}):
   for x in nonprimes do
      print(x);
   end do;
```

8.4.3 while 循环

while 循环重复执行程序语句，直到布尔表达式不成立。while 循环的语法结构如下：

```
[> while conditional_expression do
        statement_sequence
   end do;
```

表达式 conditional_expression 称为 while 的条件。它必须是布尔值表达式，也就是说：它的值必须是 true，false，或者 FAIL。

while 循环的工作方式如下：

首先 Maple 计算 conditional_expression 条件，如果值为 true，Maple 运行循环体，重复循环直到 conditional_expression 条件的值为 false 或 FAIL。

注意，Maple 在执行循环体之前计算 conditional_expression 条件。

示例：

```
[> collatz := 7;
   while collatz <> 1 do
      print(collatz);
      if collatz mod 2 = 0 then
         collatz := collatz / 2;
      else
         collatz := 3*collatz + 1;
      end if;
   end do;
```

8.4.4 嵌套循环

用户可以在 for/from 或 for/in 循环中加入 while 语句。

常规的 for/from 循环有下面的语法结构：

```
[> for counter from initial by increment to final
      while conditional_expression do
      statement_sequence
            end do;
```

常规的 for/in 循环有下面的语法结构：

```
[> for variable in expression
      while conditional_expression do
         statement_sequence
            end do;
```

在 for 循环的每个迭代开始时，Maple 首先测试 for 循环条件，然后对 conditional_expression 求值。

- 如果 conditional_expression 的值等于 false 或 FAIL，Maple 退出循环。
- 如果 conditional_expression 的值等于 true，Maple 执行 statement_sequence。

示例：

```
[> lst1 := [2, 5, 8, 11];
   for n in lst1 while isprime(n) do
      print(n);
   end do;
   lst2 := [2, 7, 3, 7, 13];
   for k to numelems(lst2) while isprime(lst2[k]) do
      print(lst2[k]);
   end do;
```

8.5 创建一个模块 Module

Maple 的 Procedure 将一组命令与单个命令关联在一起，Module（模块）是更加高级的程序结构，类似于面向对象语言中的类（Class），让用户关联相关的程序和数据。

模块的一个重要功能是 export 变量，这些被导出的变量可以被模块外部的 Maple 语句使用。

```
[> z5 := module()
        option package:
        export plus, times;
        local priv:
        plus  := (a, b) -> a + b mod 5;
        times := (a, b) -> a * b mod 5 + priv(a,b);
        priv  := (a, b) -> a ^ b:
   end module;
```

z5:=module() ... end module

```
[> z5:-plus(2,4);
```

1

```
[> z5:-times(2,4);
```

19

```
[> z5:-priv(1,2)
```

Error, priv is not a command in the z5 package

另一种使用方式：

```
[> m := module()
   local x, yproc;
   export xproc, y;
      x := 0;
      y := 0;
```

```
    yproc := proc()
      y := y + 1;
      return y;
    end proc;

    xproc := proc()
      x := x + yproc();
      return x;
    end proc;
  end module;
```

m:=module() ... end module

```
[> m:-xproc(), m:-y;
```

1, 1

```
[> m:-xproc(), m:-y;
```

3, 2

用户不能从外部使用局部变量：

```
[> m:-x;
```

Error, module does not export `x`

```
[> m:-yproc();
```

Error, module does not export `yproc`

大部分 Maple 函数包是用 Module 实现的，通过在定义中加入 option package，Packages 中的命令是 Module 的输出。

通过输入 with(SomePackageName); 然后可以使用该函数包中的所有命令，无需重复写出函数包名。

```
[> m2 := module()
  local x;
  export y;
  option package;
    x := 0;
    y := proc($)
      x := x + 1;
      return x;
    end proc;
  end module;
```

module () local x; export y; option package; end module

```
[> m2:-y();
```

3

```
[> with(m2);
                                    [y]
[> y();
                                     4
[> m2[y]();
                                     5
```

8.6 创建一个函数包 Package

一个 "package" 是将一些算法（可能有其他数据）打包在一起。通常，一个 package 提供一定范围内的功能，用于解决某些明确定义的问题域中的问题。

Maple 中大部分算法库都是以 packages 的形式出现的。例如，Maple 中的群论函数包（GroupTheory package），几何（geometry 和 geom3d packages），数论（NumberTheory），线性代数（LinearAlgebra 和 Student[LinearAlgebra]）。

```
[> savelib
[> savelib(z5, "c://temp//my_first_package.mla")
[> restart
[> z5
[> libname
"C:\Program Files\Maple 2020\lib", "C:\Users\skhan\maple\toolbox\AISC Shapes Database
                v14.1\lib","C:\Users\skhan\maple\toolbox\Google Maps and
                 Geocoding\lib","C:\Users\skhan\maple\toolbox\Syrup\lib"
[> libname := libname, "c://temp"
libname := "C:\Program Files\Maple 2020\lib", "C:\Users\skhan\maple\toolbox\AISC Shapes
Database v14.1\lib", "C:\Users\skhan\maple\toolbox\Google Maps and Geocoding\lib",
                "C:\Users\skhan\maple\toolbox\Syrup\lib", "c://temp"
[> with(z5)
                                [plus, times]
[> plus(3,4)
                                     2
```

8.7 代码调试和分析

8.7.1 编程习惯：注释

*** 强烈建议在代码中加入大量的注释，即使您只是在实验! ***

至少应该有以下注释：

- 程序摘要
- 所有输入参数的类型和含义
- 所有返回值的类型和含义
- 可能的错误情况
- 循环不变式

建议，代码中应该每 5 行有一个注释，并且解释接下来的 5 行应该完成什么。

8.7.2 调试

1. trace

trace 是一个调试工具，对程序使用 trace 命令，可以让 Maple 输出调用参数，以及输出每次退出时的返回值。

```
[> restart:
[> f := proc(x, y :: posint, $)
   local i, z;
      z := x;
      for i to y do
         z := z^2;
      end do;
      return z;
   end proc:
[> trace(f):
[> f(5, 2):
   {--> enter f, args = 5, 2

   <-- exit f (now at top level) = 625}
```

如果 f 命令的结尾是分号，而不是冒号，那么将输出 f 中所有内部语句的结果。

```
[> f(5, 2);
```

```
{--> enter f, args = 5, 2
                                        z := 5
                                        i := 1
                                        z := 25
                                        i := 2
                                        z := 625
                                        i := 3
<-- exit f (now at top level) = 625}
                                        625
```

g 是一个不含参数的程序。

```
[> g := proc($)
       return f(5, 2);
   end proc:
[> to 3 do
       g();
   end do;
   {--> enter f, args = 5, 2
                                        z := 5
                                        i := 1
                                        z := 25
                                        i := 2
                                        z := 625
                                        i := 3
   <-- exit f (now in g) = 625}
                                        625
   {--> enter f, args = 5, 2
                                        z := 5
                                        i := 1
                                        z := 25
                                        i := 2
```

```
                                z := 625

                                i := 3

    <-- exit f (now in g) = 625}

                                625
```

```
[> untrace(f);
```

2. showstat 和 stopat

可用 showstat 命令查看所有使用 Maple 语言编写的程序的源代码。

```
[> showstat(f);
```

也可以查看 Maple 内置的算法。

```
[> showstat(sin);
```

但是并非所有的内置算法/命令都是用 Maple 语言编写的。

```
[> showstat(evalf);
```

当发生意外的错误时，使用可以

```
[> tracelast;
```

帮助找到错误的位置。

```
[> h := proc(x)
    local y,z;
      y := x^2-1;
      z := 1/y;
      return sin(y)+cos(z);
    end proc:
[> h(1.3);
[> h(-1);
[> tracelast;
```

3. Debugger 调试器

Maple 内置交互式代码调试器。使用它，您可以单步执行计算以确定正在发生的事情。

进入调试器的两种方法：

- 使用 stopat, stoperror 或者 stopwhen 设置断点
- 在运行计算过程中，按下工具栏上的调试按钮（非计算机状态下该图标显示为灰色非活动状态）

调试器单步执行的命令：

- step: 细粒度的单步，执行下一条语句，无论是嵌套、组合语句（循环、条件等)还是调用函数
- into: 中等粒度的单步，执行下一条语句，进入组合语句（循环、条件等)，但跳过程序调用
- next: 粗粒度的单步，执行下一条语句，同时跳过复合语句和程序调用
- outfrom: 运行到当前程序的结尾，然后停止
- continue: 运行到下一个断点
- where: 显示动态程序调用堆栈
- list: 显示源代码的摘要，包括当前断点

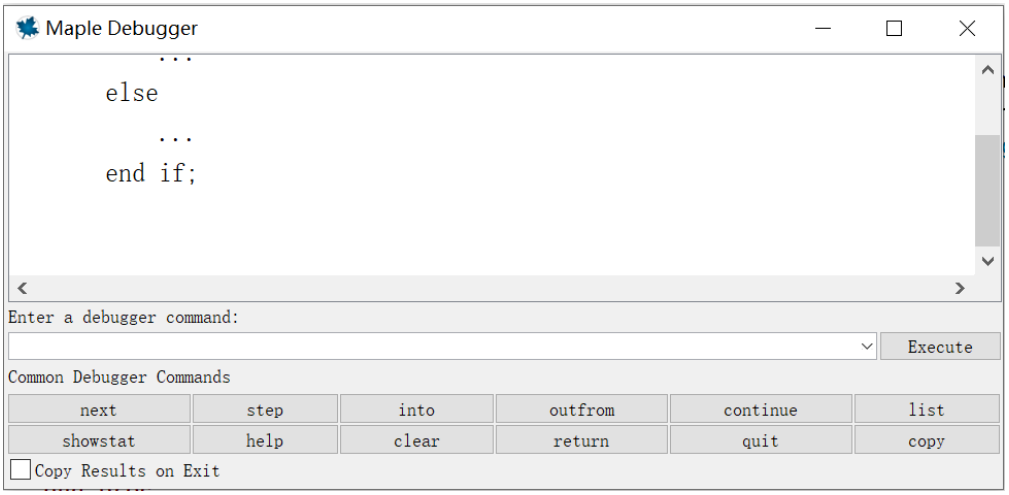
stopat 设置一个断点，目的是启动 Maple debugger.

```
[> f := proc(n :: nonnegint, $)
    local i;
      for i to n do
        print(i);
      end do;
    end proc;
    g := proc(m :: nonnegint, $)
    local i;
      for i to m do
        f(i);
      end do;
    end proc;
```

$$f := \mathbf{proc}(n::nonnegint,\ \$)\ \mathbf{local}\ i;\ \mathbf{for}\ i\ \mathbf{to}\ n\ \mathbf{do}\ print(i)\ \mathbf{end\ do\ end\ proc}$$

$$g := \mathbf{proc}(m::nonnegint,\ \$)\ \mathbf{local}\ i;\ \mathbf{for}\ i\ \mathbf{to}\ m\ \mathbf{do}\ f(i)\ \mathbf{end\ do\ end\ proc}$$

```
[> g(3);
[> stopat(g);
```




```
[> g(3);
[> unstopat(g);
```

此外，tracelast，stoperror 也是跟踪错误原因的有用工具。

```
[> stoperror("numeric exception");
[> h(-1);
[> unstoperror("numeric exception");
```

注意：当任意错误发生时，stoperror("") 都会进入 Maple Debuger。

8.7.3 代码分析

1. CodeTools

发现一个命令消耗的时间和内存的一个快速的方法是 CodeTools:-Usage。

```
restart:
CodeTools:-Usage(ifactor(10^59+1));
```

memory used=142.68MiB, alloc change=72.00MiB, cpu time=1.94s, real time=1.95s, gc time=46.88ms

$$(11)\,(109080584206809867783 7)\,(441192277099607410964453536285 1087)\,(1889)$$

2. CodeTools:-Profiling

可用 CodeTools:-Profiling 列出每段程序执行所花时间、内存信息。

```
[> restart;
[> quicksort := proc(l :: list(numeric), $)
    local less, greater, pivot, i;
```

```
        if l = [] then
            return l;
        end if;

        less := [];
        greater := [];
        pivot := l[1];
        for i from 2 to numelems(l) do
            if l[i] < pivot then
                less := [less[], l[i]];
            else
                greater := [greater[], l[i]];
            end if;
        end do;

        return [quicksort(greater)[], pivot, quicksort(less)[]];
    end proc;
[> quicksort([seq(1 .. 400)]);

[> CodeTools:-Profiling:-Profile(quicksort):

[> to 10 do
        quicksort([seq(1 .. 400)]):
    end do:

[> showstat(quicksort);

quicksort := proc(l::list(numeric), $)
local less, greater, pivot, i;
     |Calls Seconds  Words|
PROC | 8010   5.984 114918858|
   1 | 8010   0.016  24030| if l = [] then
   2 | 4010   0.015      0|     return l
                         end if;
   3 | 4000   0.000      0| less := [];
   4 | 4000   0.000      0| greater := [];
   5 | 4000   0.000      0| pivot := l[1];
   6 | 4000   0.764   8000| for i from 2 to numelems(l) do
   7 |798000   2.036 2394000|     if l[i] < pivot then
   8 |    0   0.000      0|         less := [less[], l[i]]
                           else
   9 |798000   3.043 111446580|         greater := [greater[], l[i]]
                           end if
                         end do;
  10 | 4000   0.110 1046248| return [quicksort(greater)[], pivot, quicksort(
                           less)[]]
end proc
```

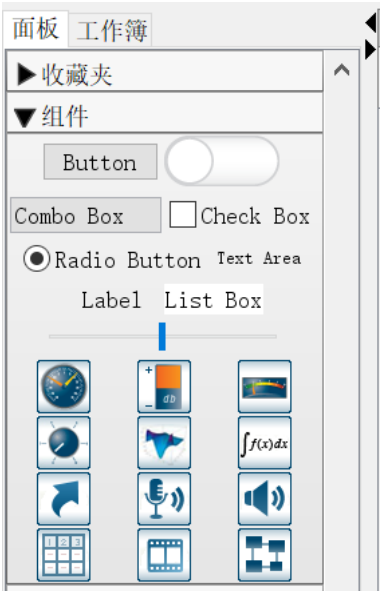
备注：showstat 显示三列信息：|Calls Seconds Words|，第一列 Calls 表示该语句已执行的次数；第二列 Seconds 表示语句花费的 CPU 时间量；第三列 Words 表示语句执行所分配的内存量。

第九章 图形用户界面 GUI 应用开发

Maple 内置丰富的 GUI（图形用户界面）组件，例如按钮、滑动条、文本框等，你可以直接将它们拖入到 Maple 文件中。通过插入 GUI 组件到 Maple 文件中，让你的技术文件变成了完整的应用程序，使用者可以在文件中直观地操作。这些 GUI 组件帮助你创建专业的技术文件，方便同事或学生使用和分享，由于这些 GUI 组件属性中写入了 Maple 代码，使用者无需理解这些代码就可以交互式使用 Maple 技术文件。本章将描述如何在 Maple 文件模式下使用这些 GUI 组件创建图形化用户界面应用程序。

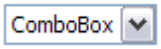
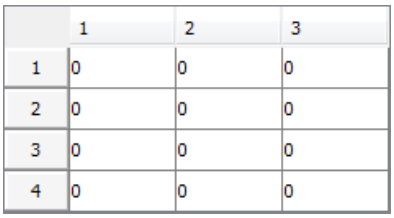
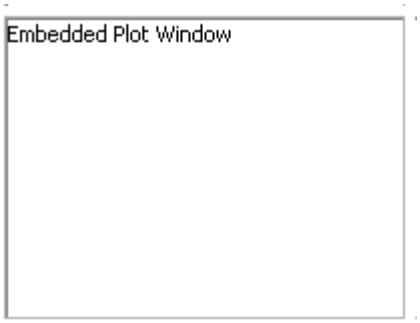
9.1 常用的 GUI 组件

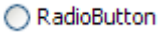
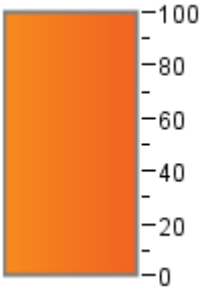
GUI 组件面板在 Maple 软件窗口左侧，如果找不到，可以鼠标右键点击面板，从右键菜单中选择显示面板“GUI 组件”。



GUI 组件可以让使用者透过 GUI 组件执行 Maple 代码，而不是直接执行命令语句。GUI 组件可以单独使用，例如点击一个按钮执行代码，也可以组合在一起使用，例如从下拉菜单中选择一个项，执行结果显示在一个 GUI 组件中。

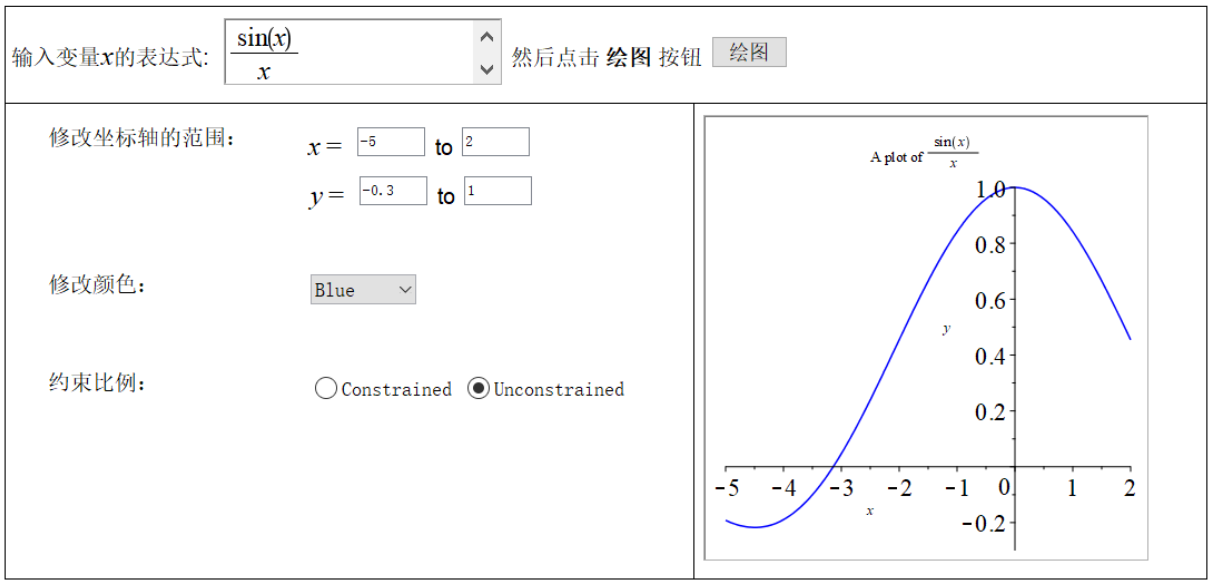
GUI组件的名称和描述	图标
Button – 按钮，点击时完成一个动作；也就是执行代码。	Button
Check Box - 复选框，选择或取消选择。改变标题，输入当	CheckBox

值变化时要运行的代码。	
Combo Box - 组合框，从下列菜单中选择其中一项。改变列出的项，输入当值变化时要运行的代码。	
Data Table - 数据表，将该GUI组件与工作表中的矩阵、矢量、或数据连接。	
Dial - 刻度盘，选择或显示一个整数值或浮点数值。改变显示，输入当值改变时要运行的代码。	
Label - 标签，显示一个标签。标签值会随着文件或其他GUI组件中的代码同步更新。	
List Box - 列表框，显示一列项。改变列表项，输入当其中一项被选中时要运行的代码。	
Math Expression - 数学表达式，输入或显示一个数学表达式。表达式会随着文件或其他GUI组件中的代码同步更新。	
Meter - 仪表，选择或显示一个整数值或浮点数值。改变显示，输入当值变化时要运行的代码。	
Microphone Component – 麦克风，从录音设备中获得声音。可以修改属性选项，输入代码启动或停止录音动作。	
Plot - 图形，显示2-D或3-D图形或动画。这个图形或动画可以使用与其他图形相同的方式交互式操作。值会随着文件中或其他GUI组件中的代码同步更新。也可以输入当excute code指针用于在图形区域中点击或拖动时运行的代码。	

Radio Button - 单选按钮，与其他单选按钮一起使用，选择一组中的一个。输入当值变化时要运行的代码。	
Rotary Gauge - 旋转式计量器，选择或显示一个整数或浮点数值。改变显示，输入当值变化时要运行的代码。	
Slider - 滑动条，选择或显示一个整数或浮点数值。改变显示，输入当值变化时要运行的代码。	
Text Area - 文字区域，输入或显示纯文本。文字会随着文件中或其他GUI组件中的代码同步更新，你也可以输入当文字变化时要运行的代码。	
Toggle Button - 触发按钮，选择或显示两项中的一项。改变显示的图片，输入当值变化时要运行的代码。	
Video Player – 视频播放器，可以输入代码，当视频播放器播放到一个标记时将触发一个动作。	
Volume Gauge - 体积计量器，选择或显示一个整数或浮点数值。改变显示，输入当值变化时要运行的代码。	
Shortcut Component – 快捷键，用于链接不同类型的内容，例如帮助页面、MapleCloudon中的文件、网址。	

Speaker Component – 扬声器，播放声音。	

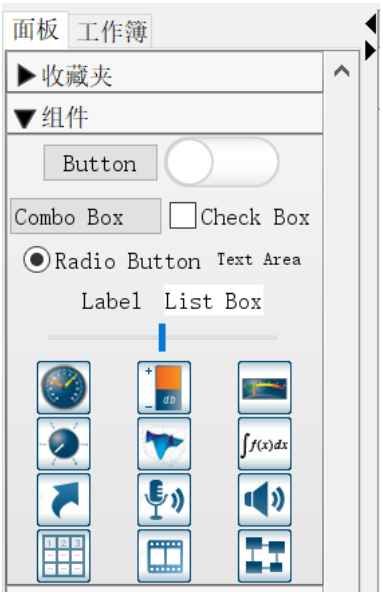
下面的示例演示了如何使用多个 GUI 组件共同工作。用户输入表达式，点击按钮绘制表达式的图形。图形选项由文本区域、组合框、数学表达式和单选按钮控制。



9.2 GUI 组件操作

9.2.1 插入 GUI 组件

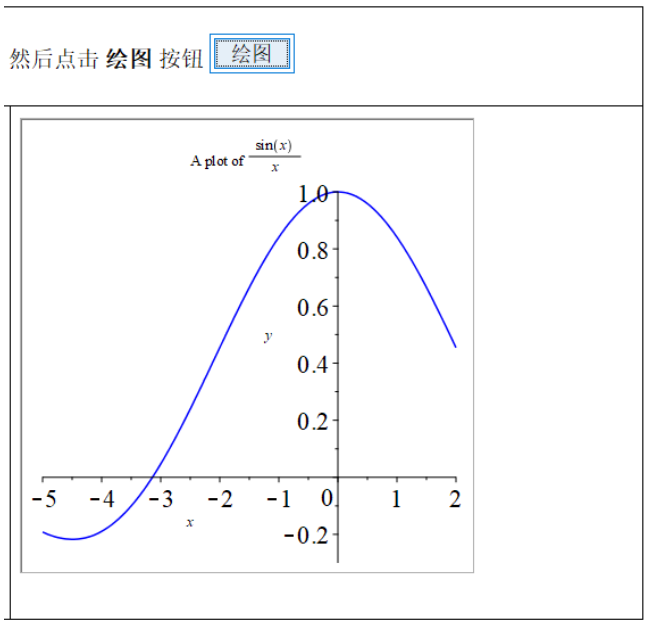
可以从 Maple 界面左侧的组件面板（如下图）插入图形界面控件，也可以剪切、拷贝、粘贴 Maple 文件中已有的 GUI 组件到文件中的其他区域。拷贝 GUI 组件时，GUI 组件名称会自动改变，其他属性基本保持不变，因此这两个 GUI 组件是有区别的。



9.2.2 编辑组件的属性

编辑文件中 GUI 组件的属性：

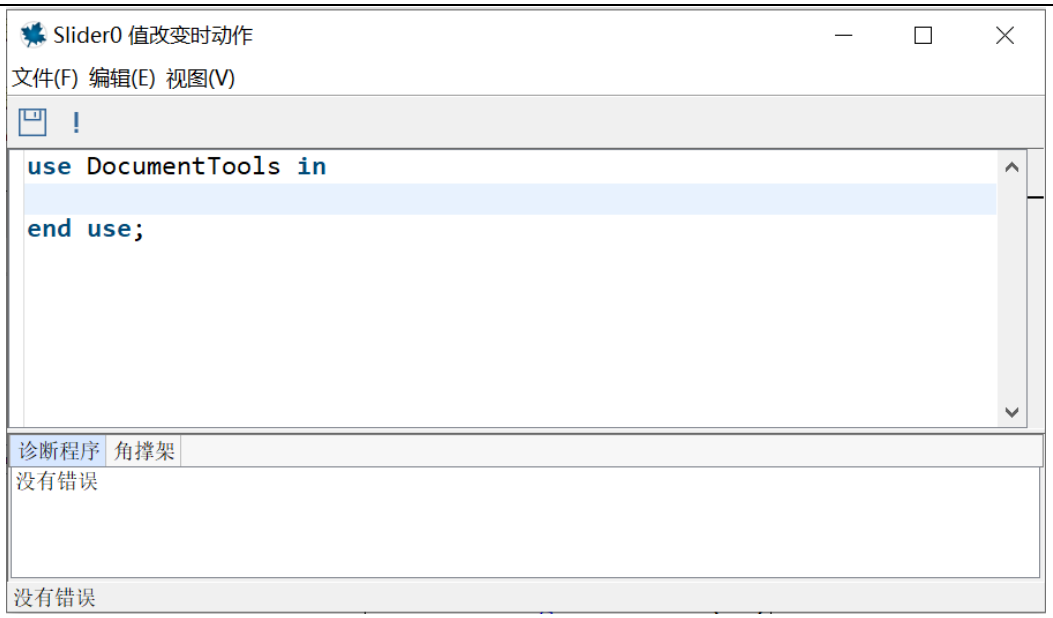
1. 点击该组件显示关联菜单。





2. 在关联面板中，基于需要输入数值或者内容。

3. 定义一个动作，例如移动滑动条时要执行的动作。首先点击 GUI 组件，然后从关联菜单选择“编辑 值改变时 的动作”。点击后将弹出代码编辑器窗口，在其中输入事情发生时执行的 Maple 代码。



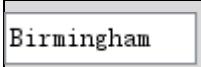
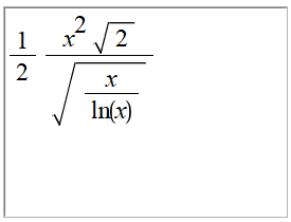
函数包 DocumentTools 提供了一系列命令实现互动操作 GUI 组件的属性。

例如，使用命令 Do(%ComponentName);获得 GUI 组件的值。

使用命令 Do(%ComponentName = value);设置 GUI 组件的值。

```
with(DocumentTools);   # 加载函数包
```

[*AddIcon, AddPalette, AddPaletteEntry, Components, ContentToString, CreateTask, Do, GetDocumentProperty, GetProperty, InsertContent, InsertTask, Layout, RemovePalette, RemovePaletteEntry, RemoveTask, Retrieve, RunWorksheet, SetDocumentProperty, SetProperty, Tabulate*]

GUI 组件	名称	获取和设置值
	Slider0	Do(%Slider0); 60 Do(%Slider0=40); 40
	TextArea0	Do(%TextArea0); Birmingham Do(%TextArea0=Birmingham); Birmingham
	MathContainer0	Do(%MathContainer0 = x^2/(sqrt(x/ln(sqrt(x))))); $\frac{x^2\sqrt{2}}{2\sqrt{\frac{x}{\ln(x)}}}$ Do(%MathContainer0);

		$\frac{1}{2}\frac{x^2\sqrt{2}}{\sqrt{\frac{x}{\ln(x)}}}$

9.2.3 删除 GUI 组件

可以通过下列途径删除 GUI 组件：

- 使用删除键。
- 使用空格键。
- 用鼠标选中该 GUI 组件，然后从文件菜单中选择编辑→删除单元。

9.2.4 在文件中使用 GUI 组件

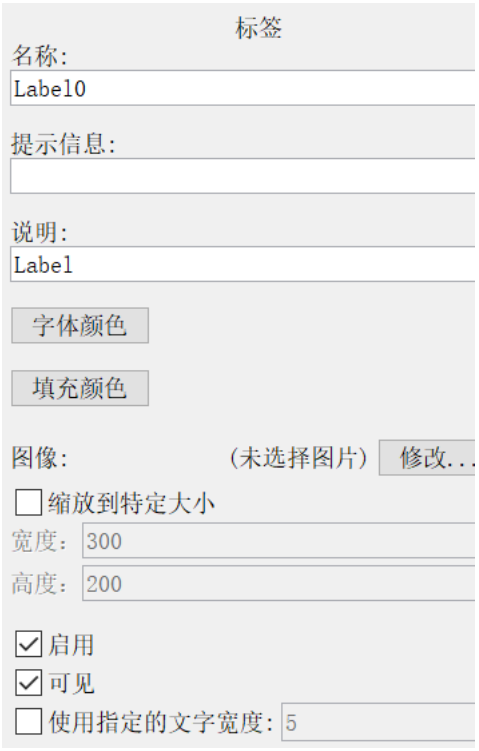
使用 GUI 组件显示的信息包括计算、获取读者的输入、当点击按钮时完成计算等，所有的这些无需使用者理解 Maple 命令。这些组件可以添加到 Maple 文件中的任意位置，包括文件块或表格。关于各个 GUI 组件的详细信息，请参阅相应的帮助页。

下面是一个简单的示例，插入滑动条，以及显示滑动条的当前值。

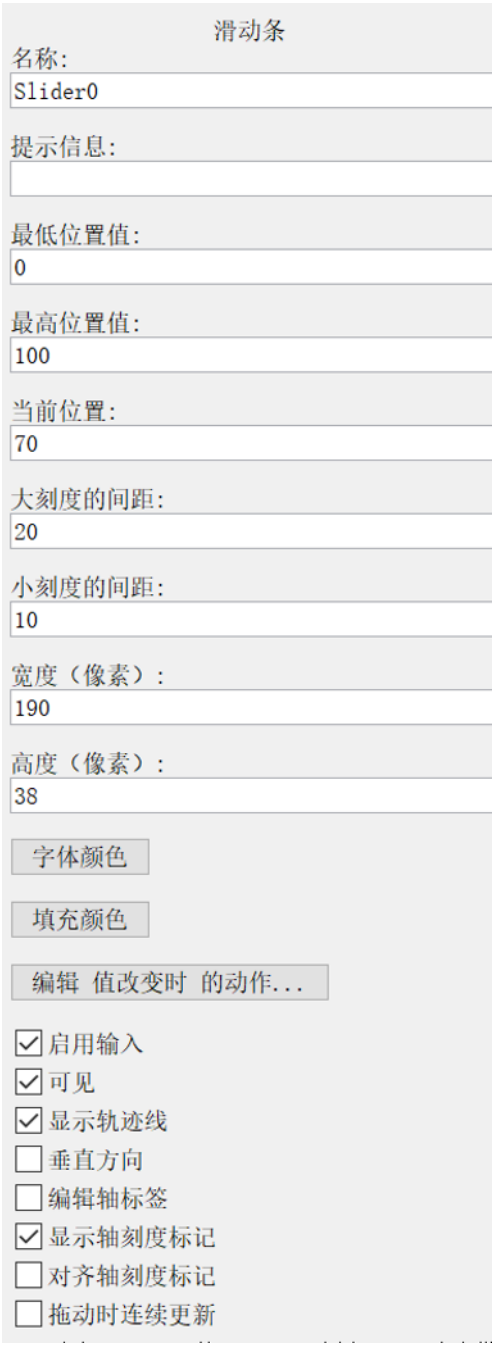
1. 打开一个 Maple 文件，将光标放在要插入 GUI 组件的地方。
2. 找到 GUI 组件面板，点击其中的滑动条，一个滑动条图标会插入到文件中。
3. 从 GUI 组件中找到标签项，点击它，一个标签 GUI 组件会插入到文件中。



4. 鼠标点击“标签”组件，关联菜单显示在右侧窗口。



图：标签属性对话框



图：滑动条属性对话框

5. GUI 组件命名为 SliderLabel，点击确认按钮。
6. 鼠标点击滑动条，关联菜单见上图。
7. GUI 组件命名为 Slider1。
8. 输入最小位置的值为 0，最大位置上的值为 100。
9. 输入最大刻度线为 20，最小刻度线为 10。
10. 为了定义一个动作，点击当值变化时发生的动作右侧的编辑按钮。打开的对话框允许你写入在标签 GUI 组件中显示滑动条当前值的动作的对应程序。对话框中包含了提示

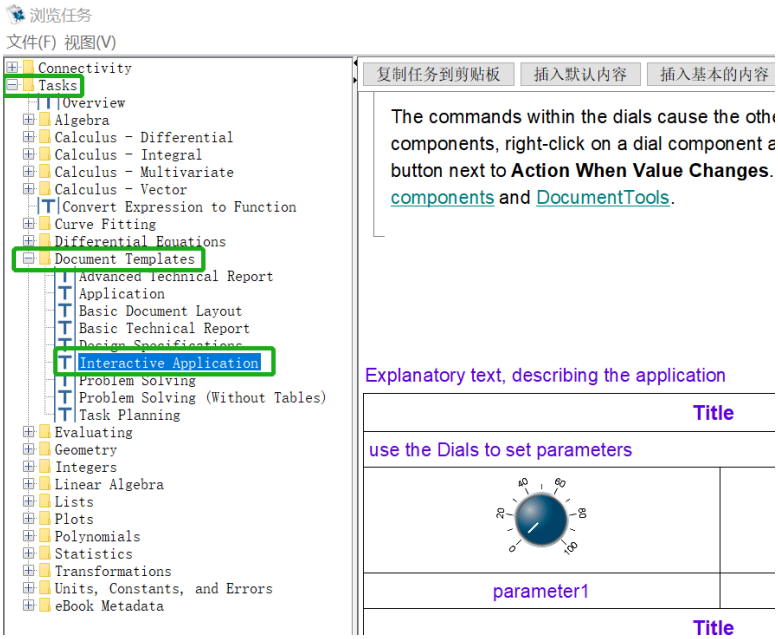
你如何在 GUI 组件中如何编程的说明。use...in/end use;语句允许你使用短格式使用程序包中的命令，无需激活程序包。关于该命令的详细信息，请参阅 use 帮助页。

11. 在对话框中的 end use;语句前输入下面的命令。
12. Do(%SliderLabel(caption)=%Slider1(value));
13. 点击确认按钮。
14. 确认拖动时连续更新复选框被选中。

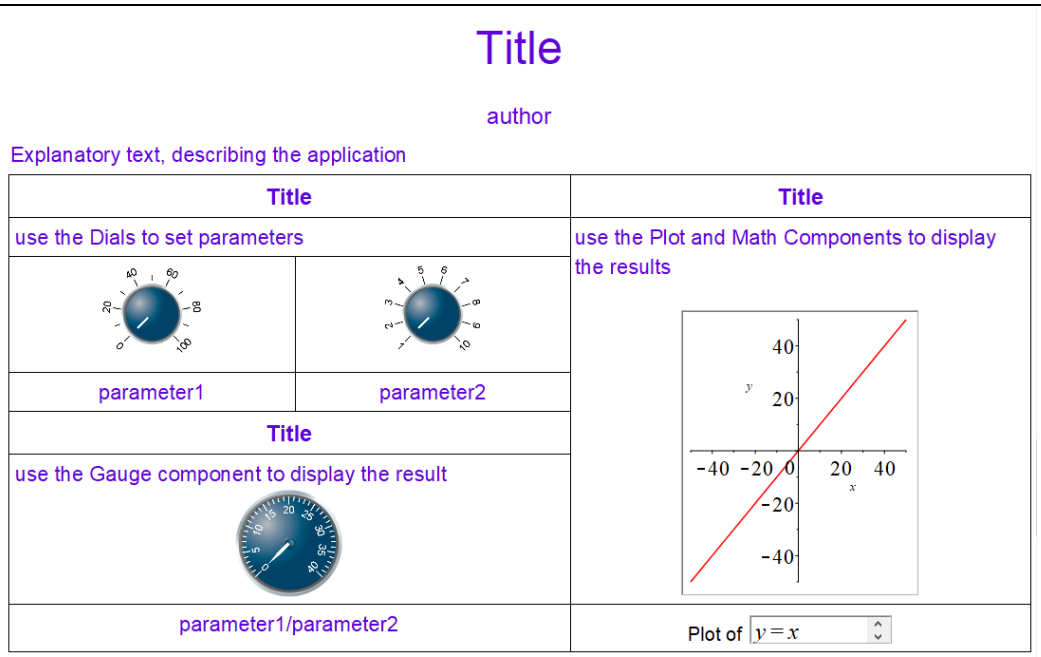
当你移动滑动条上的箭头指示图标时，滑块 **Slider1** 对应的值将显示在标签 **SliderLabel** 标题区域。

9.3 示例 – 创建包含 GUI 组件的文件

Maple 内置大量的任务模板，我们以其中一个模板为例，详细描述创建的补一个步骤。任务模板的路径是：工具 → 任务 → 浏览，在任务浏览器窗口中选择 Document Templates 然后选择 Interactive Application。



点击“插入基本的内容”，下面的内容将插入到文件中光标所在的位置。

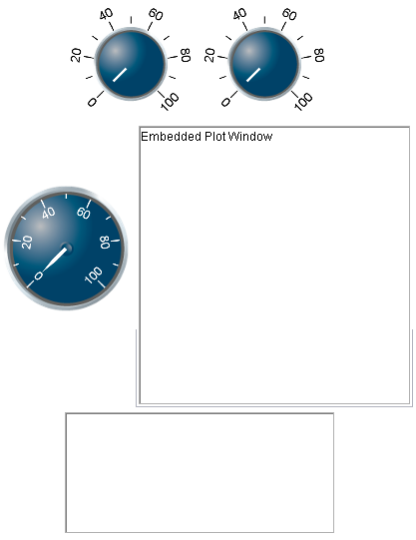


这里，我们重新对图元件配置。这个示例中使用了两个参数 a 和 b 作为输入，然后对函数 $y=bx+a$ 绘图，然后计算 $\frac{a}{b}$。

1. 创建 GUI 组件

当图元件创建完成后，最好使用表格布局。

在文件中拖入两个“刻度盘”，对应地设置参数 a 和 b，其中一个“旋转计量仪”显示结果 $\frac{a}{b}$，“绘图元件”显示图形，一个“数学表达式”组件显示函数。



2. 编辑 GUI 组件的显示

打开第一个“刻度盘”的关联菜单，注意到它已经有一个命名。这个命名用于从其他组件或程序代码关联到该组件，并且是唯一的。按如下方式改变各个 GUI 组件的显示：

- Dial0：不需要改变。
- Dial1：改变最大位置上的值为 10，大刻度标记的间距为 1，小刻度标记的间距为 1。
- RotaryGauge0：改变最大位置上的值为 40，大刻度标记的间距为 5，小刻度标记的间距为 1。
- Plot0：不需要改变。
- MathContainer0：改变以像素为单位的宽度为 200，以像素为单位的高度为 45。

注意：在做上面的设置时要注意所有图元件的命名。

3. 创建 GUI 组件的动作

当值改变时 GUI 组件可以完成动作，因此要运行的代码需要写在刻度盘内。因此，无论它们中的任一个发生变化时，其他的 GUI 组件将同步更新反映这个变化。

下面的 Maple 命令获取参数的值，然后将它们显示在其他三个 GUI 组件上：

```
> parameter1:=Do(%Dial0):
> parameter2:=Do(%Dial1):
> Do(%RotaryGauge0=parameter1/parameter2);
> Do(%Plot0=plot((parameter2*x+parameter1), x=-50..50, y=-50..50));
> Do(%MathContainer0=(y=parameter2*x+parameter1));
```

4. 测试动作

为了测试这些命令，首先使用下面的命令加载 DocumentTools 函数包。

```
> with(DocumentTools)
```

在文件中执行上面的命令，检查一下你插入的图元件是否同步更新：计量器应该变为计算的值，图形应该显示在图形组件上，函数应该出现在数学表达式组件上。

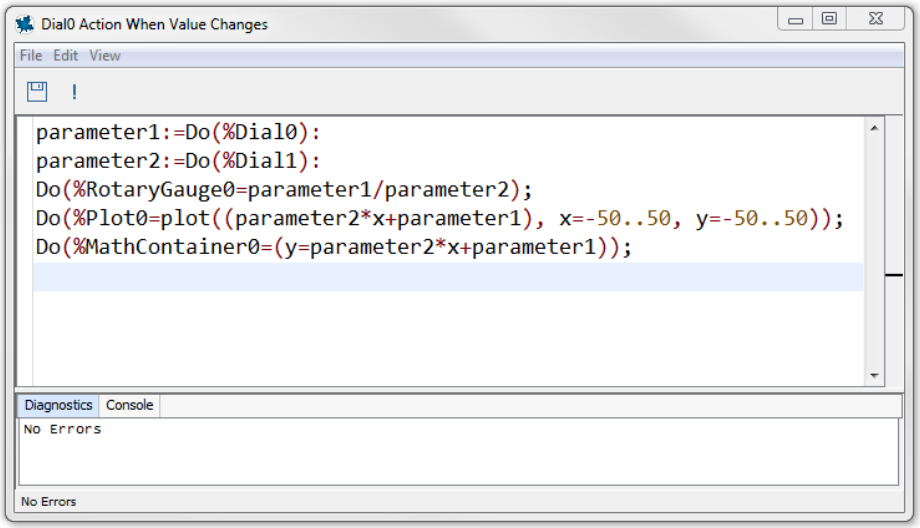
5. 发现并解决问题

第一个 Do 命令返回错误，这是因为第二个参数是 0。避免这个问题的一个方法是改变第二个刻度盘的取值范围。打开第二个“刻度盘”的组件属性对话框， 将最低位置的值 0 修改为 1。或者，你也可以通过改变代码，使用 if 语句，实现同等的效果。

6. 拷贝动作到 GUI 组件中

当命令能够正常工作后，你可以拷贝它们到组件中。

选中第一个“刻度盘”组件，从关联菜单选择“编辑 值改变时 发生的动作”。拷贝和粘帖上面的命令到 use 语句之间的空间内。



保存代码，然后退出代码编辑器。

用相同的方法设置第二个“刻度盘”，现在移动刻度将更新“旋转计量仪”、图形和公式。

7. 创建 GUI 组件的布局

创建第二个表格，然后剪切和粘帖这些图元件到表格中，并配以说明文字。重要：你必须剪切，而不是拷贝，否则它们的命名为了避免重名会被自动改变。

