

Wolfram Mathematica 在数学物理方法课程中的应用

Wolfram Mathematica 是一款符号计算软件，广泛使用于科学、工程、数学、计算等领域。经过 30 多年的发展，截止 2024 年已经到了版本 14，功能已经得到了极大的扩充，除了强大的符号与数值计算功能如微积分、幂级数展开、线性代数计算、求解微分方程、积分变换等之外，也可以进行并行计算、GPU 计算，还能搭建神经网络、进行机器学习。使用 Wolfram Alpha 知识搜索引擎，Mathematica 还能进行地理、金融、社会、语言等各方面的计算，号称“全球现代技术计算的终极系统”。限于篇幅，本章只介绍与本课程内容相关的内容。

在目前的数学软件中，Mathematica 的帮助文档是最好的之一。其帮助文档内置了《虚拟全书》，这是一份官方的入门教程。《虚拟全书》可以通过在帮助文档地址栏输入 `tutorial/VirtualBookOverview` 进入，或者直接在地址栏搜索“虚拟全书”。而且 Mathematica 帮助文档可以直接在上面进行代码的修改和运行，中文版软件的帮助文档绝大部分内容已经翻译，降低了学习的难度。此外中文版的帮助文档还支持输入中文内容进行搜索。在学习 Mathematica 的过程中，应该经常查阅帮助文档。学习以下内容时，请打开软件运行例子并时刻查询帮助文档。以下所遇见的函数不做具体用法说明，请读者自行用鼠标选定函数按 **F1** 键查询其用法。

另外，用于 Mathematica 的编程语言是一种严谨且优雅的语言，称为 Wolfram 语言。Wolfram 语言同时含有面向过程编程和函数式编程，但函数式编程在 Mathematica 里面更加方便，运行速度也更快，也更符合 Wolfram 语言的编程风格。想要深入了解的读者，可以参考帮助文档 `tutorial/CoreLanguageOverview`。以下内容可能涉及一点 Mathematica 语法的内容，出现时只做简单介绍。在下面的举例中，输入和输出的内容都尽量与屏幕显示的格式一致，但也存在一些不同之处。本文附有一个 Mathematica notebook 文件，包含本文中所有的可运行代码。本文中很多绘图的结果不在本文中展示，请读者自行参考代码文件。



图 1: Mathematica 帮助文档首页

1 Mathematica 基础知识

以下内容只是简单的介绍，详见帮助文档 `tutorial/IntroductionOverview`。

1.1 进行输入和计算

Mathematica 可以直接当作计算器使用。打开 Mathematica，新建一个笔记本（.nb）文件，就可以进行表达式的输入和计算。把光标移动到笔记本窗口内，键入要操作的表达式，如

```
1+1
```

然后按下 `Shift+Enter` 或者小键盘的 `Enter`，就会在显示屏上显示：

```
In[1]:=1+1
```

```
Out[1]=2
```

其中 `In[1]` 和 `Out[1]` 是自动出现的，表示第 1 次的计算结果。若想重用以前的计算结果，可以使用 `%` 调用，其中 `%` 表示前一个计算的结果，`%%` 表示倒数第二个计算的结果，`%n` 表示第 `n` 个计算的结果：

```
In[2]:=Sqrt[%]
```

```
Out[2]=√2
```

```
In[3]:=N[%2]
```

```
Out[3]=1.41421
```

除了一般的输入，Mathematica 还支持二维输入，比如按下 `Ctrl + 2` 键直接输入根号，`Ctrl + /` 键输入二维的分号，或者在面板——数学助手里面点击输入二维表达式。想要了解更多的二维表达式输入，可以参考帮助文档 `tutorial/TwoDimensionalExpressionInput`。

1.2 算术运算符号

+（加号，Plus），-（减号，Subtract），*（乘号，Times），/（除号，Divide），^（乘方，Power）。此外 * 可以用空格代替，**（NonCommutativeMultiply）表示没有交换律的乘号，通常用于四元数的运算以及自定义算符的运算中。

1.3 数与数学常数

Mathematica 中数包括整数、有理数、实数、复数。在计算表达式的过程中，除非遇到小数，否则 Mathematica 会给出精确结果：

```
In[1]:=Sqrt[3]
```

```
Out[1]=√3
```

```
In[2]:=Sqrt[3.]
```

```
Out[2]:=1.73205
```

如果想要得到近似值的话，Mathematica 的函数 `N` 可以给出任意精度的近似解，如：

```
In[3]:=N[Pi,100]
```

```
Out[3]=3.141592653589793238462643383279502884197169399375105820974944592307816
406286208998628034825342117068
```

Mathematica 也可以指定输入的数的精度，比如 `3.1`30` 就代表精确度为 30 的数，再如

```
In[4]:=3.1`30 + 2.1`20 + 1
```

```
Out[4]:=6.20000000000000000000
```

由于参与计算的三个数精确度依次是 30、20 和无穷大，所以最后给了精确度为 20 左右的结果。更多信息参考函数 `Precision` 和 `Accuracy`。

Mathematica 中虚数的单位是大写的 `I`， π 为 `Pi`，自然对数底数 e 为 `E`，如

```
In[5]:=E^(1+Pi I)
```

```
Out[5]=-e
```

Mathematica 中还有其他常数，具体见帮助文档 `tutorial/MathematicalConstants`，下面列举了一些常用的常数：

Pi	圆周率 π	E	自然对数的底 e	EulerGamma	欧拉常数 γ
Degree	角度单位 $\pi/180$	I	虚单位 i	Infinity, -Infinity	$\infty, -\infty$

除了用上述列表中的方式输入数学常数外, Mathematica 还有额外的输入方式。比如圆周率Pi的输入方式为依次按下键盘的 `[ESC][P][I][ESC]`, 虚数I的输入方式为 `[ESC][I][I][ESC]`, 无穷大Infinity为 `[ESC][I][N][F][ESC]`。更多相关可以参考帮助文档 `tutorial/MathematicalAndOtherNotation`。

1.4 Mathematica 中的括号

Mathematica 中有四种括号: 表示运算级别的小括号 `()`, 表示函数的方括号 `[]` (`f[x]`), 表示列表的花括号 `{ }` (`{a, b, c}`), 以及表示列表索引的双括号 `[[]]` (`{a, b, c} [[1]]`)。

1.5 Mathematica 中的函数

作为一种函数式编程语言, Mathematica 中所有对象都是表达式 (见 `tutorial/ExpressionsOverview`), 函数是 Mathematica 的基础。Mathematica 中函数名或者变量名为字符串, 中间不能有空格 (否则认为是两个符号相乘), 不能有以下划线 (否则认为是模式, 模式见 `tutorial/PatternsOverview`), 不能以数字开头 (否则认为是乘积, 如 `2a` 会当做 2 和 `a` 的乘积), 不能有小数点 (否则认为是矩阵点积 `Dot`)。一般 Mathematica 中内置函数以首字母是大写的驼峰命名法命名, 而且通常是英文单词的全拼。

Mathematica 中的函数的调用一般是 `Func[x1, x2, ..., options->values, ...]` 的形式, 其中 `Func` 为函数名, `x1, x2` 等为参数, 参数可以为任意对象 (比如数字, 函数, 代数表达式, 图片等), `options->values` 是选项, 通常用于限定函数的行为模式, 比如限定算法或者绘图样式等。

初等数学函数, 参见帮助文档 `guide/ElementaryFunctions`, 这里列举一些如下:

Sqrt[z]	z ^s					幂函数
Exp[z]	Log[z]	Log[b, x]	Log10[x]	Log2[x]		指数函数, 对数函数
Sin[z]	Cos[z]	Tan[z]	Cot[z]	Sec[z]	Csc[z]	三角函数
ArcSin[z]	ArcCos[z]	ArcTan[z]	ArcCot[z]	ArcSec[z]	ArcCsc[z]	反三角函数
Sinh[z]	Cosh[z]	Tanh[z]	Coth[z]	Sech[z]	Csch[z]	双曲函数
ArcSinh[z]	ArcCosh[z]	ArcTanh[z]	ArcCoth[z]	ArcSech[z]	ArcCsch[z]	反双曲函数

特殊函数见帮助文档 `guide/SpecialFunctions`, 广义函数见 `guide/GeneralizedFunctions`, 这里列举一些如下:

UnitStep[x]	η 函数 (亥维赛的单位阶跃函数)
Gamma[z]	Γ 函数 $\Gamma(z)$
Beta[p, q]	B 函数 $B(p, q)$
PolyGamma[z]	ψ 函数 $\psi(z)$
MoebiusMu[n]	默比乌斯函数 $\mu(n)$
Zeta[z]	黎曼 ζ 函数 $\zeta(z)$
Erf[z]	误差函数 $\text{erf}(z)$
Erfc[z]	余误差函数 $\text{erfc}(z)$
LegendreP[n, x]	勒让德多项式 $P_n(x)$
LegendreP[nu, x], LegendreQ[nu, x]	勒让德函数 $P_\nu(x), Q_\nu(x)$
LegendreP[nu, mu, x], LegendreQ[nu, mu, x]	连带勒让德函数 $P_\nu^\mu(x), Q_\nu^\mu(x)$
SphericalHarmonicY[l, m, u, v]	球面调和函数 $Y_l^m(u, v)$
BesselJ[nu, z], BesselY[nu, z]	贝塞耳函数 $J_\nu(z), Y_\nu(z)$
BesselI[nu, z], BesselK[nu, z]	虚宗量贝塞耳函数 $I_\nu(z), K_\nu(z)$
AiryAi[z], AiryBi[z]	艾里函数 $\text{Ai}(z), \text{Bi}(z)$
SinIntegral[z]	正弦积分 $\text{Si}(z)$
CosIntegral[z]	余弦积分 $\text{Ci}(z)$
SinhIntegral[z]	双曲正弦积分 $\text{Shi}(z)$
CoshIntegral[z]	双曲余弦积分 $\text{Chi}(z)$
ExpIntegralEi[z], ExpIntegralE[n, z]	指数积分 $\text{Ei}(z), E_n(z)$
LogIntegral[z]	对数积分 $\text{li}(z)$
FresnelC[z], FresnelS[z]	费涅耳积分 $C(z), S(z)$
GegenbauerC[n, m, z]	盖根鲍尔多项式 $C_n^{(m)}(z)$
ChebyshevT[n, z], ChebyshevU[n, z]	切比雪夫多项式 $T_n(z), U_n(z)$
JacobiP[n, a, b, z]	雅可比多项式 $P_n^{(a,b)}(z)$
HermiteH[n, z]	厄米多项式 $H_n(z)$
LaguerreL[n, z], LaguerreL[n, a, z]	拉盖尔多项式 $L_n(z), L_n^a(z)$
Hypergeometric2F1[a, b, c, z]	超几何函数 $F(a, b; c; z)$
Hypergeometric1F1[a, b, z]	合流超几何函数 $F(a; b; z)$

关于 Mathematica 内置的符号计算和画图的函数，将在后文一一介绍。

自定义函数（详见 tutorial/TransformationRulesAndDefinitionsOverview）：Mathematica 中最常见的自定义函数的方式如下：

f[x_] := exp	定义一元函数
f[x_, y_, z_, ...] := exp	定义多元函数

如：

```
In[1] := f[x_] := x^2
In[2] := f[a]
Out[2] = a^2
In[3] := g[x_, y_] := x y
In[4] := g[2, x]
Out[4] = 2 x
```

1.6 Mathematica 中变量的清除

很多时候，因为有些变量已经经过了赋值，如果没有注意的话可能会报错或者得不到想要的结果，比如运行以下代码就会报错：

```

x=1;
Integrate[x Sin[x], x]

```

这时候就要清除变量 x:

```

Clear[x];
Integrate[x Sin[x], x]
(*-x Cos[x] + Sin[x]*)

```

另外, 想要清除所有变量, 可以运行 `Clear["Global`*"]`, 或者在菜单 计算—退出内核—*Local* 退出内核再 计算—启动内核—*Local* 启动内核, 这样之前所有的定义就会清空。

1.7 Mathematica 初学者应该注意的事情

初学者比较容易犯的错误是将 Mathematica 当作一个高级计算器而忽视了 Mathematica 其实也是一门严谨的编程环境, 以下总结了初学者应注意的事情: (1) 不能乱用括号; (2) 不能混淆中文逗号和英文逗号, 使用中文逗号的常见结果是 Mathematica 把中文逗号当做一个符号与逗号前后的内容相乘; (3) 内置函数首字母一定要大写; (4) 注意 `=` 和 `==` 的区别, `=` 是赋值 (Set), `==` 是相等 (Equal); (5) 在对 Mathematica 的语法不是非常了解的情况下, 不要使用下标 (Subscript); (6) 注意 Mathematica 界面代码的颜色, 其中黑色是已赋值的函数或者变量, 特别的, 内置函数一定是黑色的; 绿色是局部变量, 红色是错误提示。注意这些点能够避免很多错误。

2 Mathematica 的语言结构

Mathematica 是一门函数式编程语言, 里面的一切对象都是表达式。比如 `a=1` 这个赋值语句在 Mathematica 中其实是表达式 `Set[a,1]`, 定义函数的语句 `f[x_]:=x^2` 的完整形式是

```

SetDelayed[f[Pattern[x, Blank[]]], Power[x, 2]]

```

在 Mathematica 中, 我们可以用函数 `FullForm` 来查看一个表达式的完整形式, 如

```

In[1]:=FullForm[Hold[f[x_] := x^2]]
Out[1]=Hold[SetDelayed[f[Pattern[x, Blank[]]], Power[x, 2]]]

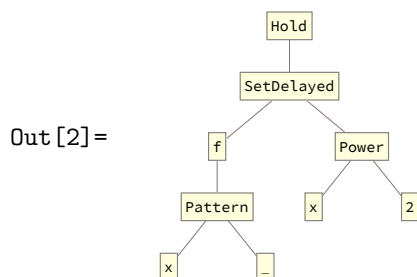
```

这里 `Hold` 的作用是让里面的表达式不进行运算。此外我们可以通过函数 `TreeForm` 来查看一个表达式的结构树, 如

```

In[2]:=TreeForm[Hold[f[x_] := x^2]]

```



Mathematica 中一切皆表达式的含义远远不止如此, 很多编程语言中的所谓语法在 Mathematica 中都是表达式的一种语法糖。比如 Mathematica 中依次运算几个语句需要用“;”隔开, 实际上这种使用方式是函数 `CompoundExpression` 的语法糖, 我们可以用鼠标选中“;”再按下 `F1` 跳转到帮助文档。我们可以使用 `FullForm` 来查看这个函数的完整形式, 如

```

In[3]:=FullForm[Hold[a = 2; b = 3; a + b]]
Out[3]=Hold[CompoundExpression[Set[a,2],Set[b,3],Plus[a,b]]]

```

再比如对列表索引也是函数, 如

```

In[4]:=FullForm[Hold[{1, 2, 3, 4, 5}[[2 ;; 3]]]]
Out[4]=Hold[Part[List[1,2,3,4,5],Span[2,3]]]

```

再比如, 之前遇到的 `%`, `%%`, `%n` 也是表达式, 代表的含义是 `Out[n]`。甚至 Mathematica 里面的图像也是表达式, 如 `FullForm[Plot[Sin[x], {x, 0, 2 Pi}]]` 将会返回函数 `Sin[x]` 图像的表达式结构。如果目的仅仅是查看 Mathematica 中对象的结构的话, 我们可以使用函数 `InputForm` 来简化输出, 如

```
InputForm[Plot[Sin[x], {x, 0, 2 Pi}]]
```

Mathematica 中为了简化代码长度使用了大量语法糖。如上文中的“`;`”、“`[[]]`”、“`;;`”、“`x_`”等都是语法糖。在初次遇到这些语法糖时, 我们可以选中这些符号按下 `F1` 键跳转到对应函数的帮助文档。Mathematica 还有前缀表达式、中缀表达式、后缀表达式, 比如 `Sin[a]` 等价于 `Sin@a` 和 `a//Sin`, `{a,b}` 等价于 `List[a,b]` 和 `a~List~b`, 此处不再细述。使用前后缀表达式可以使所写代码更加简洁直观。此外, Mathematica 中还有纯函数 (Function), 或者说匿名函数, 如

```
f=Function[x,x^2];
f[2]
(*4*)
```

纯函数可以用含 `#` (Slot) 和 `&` (Function) 的语法糖来简化表示, 比如上面的代码等价于

```
f=#^2&;
f[2]
(*4*)
```

利用前缀和后缀表达式语法糖, 上述又等价于 `#^2&[2]`, `#^2&@2` 或者 `2//#^2&`。更多关于纯函数的内容, 参考函数 `Function` 的文档。常见的语法糖列表如下:

<code>a+b</code>	Plus	<code>a-b</code>	Subtract
<code>a*b, a b</code>	Times	<code>a/b</code>	Divide
<code>a^b</code>	Power	<code>a**b</code>	NonCommutativeMultiply
<code>f@x</code>	Prefix	<code>x//f</code>	Postfix
<code>f/@list</code>	Map	<code>f@@exp</code>	Apply
<code>f@@@list</code>	MapApply	<code>patt->exp</code>	Rule
<code>patt:>exp</code>	RuleDelayed	<code>< , ></code>	Association
<code>x/.rule</code>	ReplaceAll	<code>x//.rule</code>	ReplaceRepeated
<code>x=y</code>	Set	<code>x:=y</code>	SetDelayed
<code>x==y</code>	Equal	<code>exp1===exp2</code>	SameQ
<code>f/:lhs=rhs</code>	TagSet	<code>f/:lhs=rhs</code>	TagSetDelayed
<code>{, }</code>	List	<code>[[,]]</code>	Part
<code>;;</code>	Span	<code>exp1;exp2;exp3</code>	CompoundExpression
<code>a~f~b</code>	Infix	<code>s1~~s2~~s3</code>	StringExpression
<code>s1<>s2<>s3</code>	StringJoin	<code>#</code>	Slot
<code>&</code>	Function	<code>A&&B</code>	And
<code>!A</code>	Not	<code>n!</code>	Factorial
<code>A B</code>	Or	<code>x_</code>	Blank
<code>x__</code>	BlankSequence	<code>x___</code>	BlankNullSequence
<code>_?test</code>	PatternTest	<code>patt/;test</code>	Condition
<code>f@g</code>	Composition	<code>x -> x^2</code>	Function
<code>f',f'',f'[x]</code>	Derivative	<code>x: patt</code>	Pattern
<code>A.B</code>	Dot	<code>patt..</code>	Repeated
<code>patt...</code>	RepeatedNull	<code>x=.</code>	Unset
<code>%, %%</code>	Out		

函数式编程不存在算符优先级的问题, 但是语法糖存在优先级的问题。例如我想把函数 `f2` 映射到一个列表上, 再把这个列表作为 `f1` 的参数, 由于 `@` 的优先级高于 `/@`, 以下的代码得不到想要的结果

```
In[1]:=f1@f2 /@ {1, 2}
```

```
Out[1]={f1[f2][1], f1[f2][2]}
```

为了得到想要的结果，需要添加括号

```
In[2]:=f1@f2 /@ {1, 2}
```

```
Out[2]=f1[{f2[1], f2[2]}]
```

再比如，我想把一个表达式里面的Sin替换成我自定义的一个纯函数，由于&的优先级低于->，以下的代码得不到想要的结果

```
In[1]:=Sin[1] /. Sin -> # + 1 &
```

```
Out[1]=Sin[1] /. Sin -> # + 1 &
```

解决方法同样是加上括号

```
In[2]:=Sin[1] /. Sin -> (# + 1 &)
```

```
Out[2]=2
```

所以在语法糖的时候，需要注意语法糖的优先级。

3 Mathematica 的符号计算

3.1 代数转换

此处内容，详见参考文档 [guide/AlgebraicTransformations](#)，以下举几个简单的例子，至于其他用法和选项参考帮助文档。

1. 代数式的化简 (Simplify, FullSimplify)

```
In[1]:=Simplify[(x - 1) (x + 1) (x^2 + 1) + 1]
```

```
Out[1]=x^4
```

2. 多项式展开 (Expand)

```
In[1]:=Expand[(x + 1)^3]
```

```
Out[1]=1 + 3 x + 3 x^2 + x^3
```

3. 分解因式 (Factor)

```
In[1]:=Factor[x^3 + x^2 - x - 1]
```

```
Out[1]=(-1 + x) (1 + x)^2
```

4. 合并同类项 (Collect)

```
In[1]:=Collect[a x + b y + c x, x]
```

```
Out[1]=(a + c) x + b y
```

5. 通分 (Together)

```
In[1]:=Together[a/b + c/d]
```

```
Out[1]= $\frac{a \frac{d+b}{b} c}{b d}$ 
```

6. 化为部分分式 (Apart)

```
In[1]:=Apart[1/((1 + x) (5 + x))]
```

```
Out[1]= $\frac{1}{4(x+1)} - \frac{1}{4(x+5)}$ 
```

此外 Mathematica 还有其他的代数转换函数，比如三角函数转指数函数 (TrigToExp) 等，更多相关函数见参考文档 [guide/AlgebraicTransformations](#)，限于篇幅此处不再举例。

3.2 解方程与消除变量

1. 解方程 (组)

与解方程(组)有关的函数,主要就是Solve和Reduce,考虑数值解的话还有NSolve和FindRoot,下以Solve为例:

```
In[1]:=Solve[x^2+b x+ c == 0,x]
```

```
Out[1]:={{x -> 1/2 (-sqrt(b^2-4c)-b)},{x -> 1/2 (sqrt(b^2-4c)-b)}}
```

其中 \rightarrow 表示规则(函数 Rule),想要提取 x 的解的话,使用 /. (函数 ReplaceAll),以下表示提取第一个解:

```
In[2]:=x /. %[[1]]
```

```
Out[2]=1/2 (-sqrt(b^2-4c)-b)
```

2. 消除变量 (Eliminate)

```
In[1]:=Eliminate[{x == 2 + y, y == z}, y]
```

```
Out[1]=2 + z == x
```

```
In[2]:=Eliminate[{f == x^5 + y^5, a == x + y, b == x y}, {x, y}]
```

```
Out[2]=a^5 - 5 a^3 b + 5 a b^2 - f == 0
```

3. 使方程有解的参数值 (SolveAlways)

如果我们有一系列方程 eqns 和变量 vals, 要求对与变量任意变化方程始终保持, 则可以使用函数 SolveAlways[eqns, vals]来求解对于方程里面的参数的要求,比如若使得 x 为任意值时等式 $ax+b=0$ 都成立,则对应的代码为

```
In[1]:=SolveAlways[a x + b == 0, x]
```

```
Out[1]={{a -> 0, b -> 0}}
```

从逻辑上来讲,SolveAlways[eqns, vals] 等价于 Solve[! Eliminate[! eqns, vals]].SolveAlways 在本课程中的级数求解微分方程中有着重要作用。

3.3 复数与四元数计算

Mathematica 中与复数计算的函数详见帮助文档 guide/ComplexNumbers, 相关函数举例如下:

Re[z]	求实部	Im[z]	求虚部
Abs[z]	求模	Arg[z]	求辐角
Conjugate[z]	求共轭		

(注意: Arg[z]给出的 z 的辐角取值范围在 $-\pi$ 和 π 之间)

例如:

```
In[1]:=(1 + 2 I) (3 + 4 I)
```

```
Out[1]:=-5 + 10 I
```

```
In[2]:=Re%
```

```
Out[2]:=-5
```

```
In[3]:=Arg%%
```

```
Out[3]:=pi-ArcTan[2]
```

想要展开一个复表达式, 可以使用 ComplexExpand, 下面是从帮助文档摘录的用法:

ComplexExpand[expr]	假定所有变量都是实数来展开 expr
ComplexExpand[expr,{x1,x2,...}]	假设匹配任何 xi 的变量都是复数来展开 expr

例如:

```
In[4]:=ComplexExpand[Sin[x + I y]]
```

```
Out[4]:=Cosh[y] Sin[x] + I Cos[x] Sinh[y]
```

```
In[5]:=ComplexExpand[Sin[x], x]
```



```
Out[5]:=Cosh[Im[x]] Sin[Re[x]] + I Cos[Re[x]] Sinh[Im[x]]
```

Mathematica 也内置了四元数的计算,但不同的是以程序包的形式。在执行四元数的运算之前,需要加载程序包

```
In[1]:=<< Quaternions`
```

四元数可以以 $a + b I + c J + d K$ 的形式表示,也可以用 `Quaternion[a,b,c,d]` 的形式表示,如

```
In[2]:=3 + 2 I + Quaternion[2, 0, -6, 1] - 3 Quaternion[1, 3, -2, 0]
```

```
Out[2]=Quaternion[2, -7, 0, 1]
```

```
In[3]:=FromQuaternion[%]
```

```
Out[3]=(2 - 7 I) + K
```

由于四元数的乘法是非交换乘法,所以四元数用简写为 `**` 的非交换乘法:

```
In[4]:=Quaternion[2, 0, -6, 3] ** Quaternion[1, 3, -2, 2]
```

```
Out[4]=Quaternion[-16, 0, -1, 25]
```

```
In[5]:=(1 + 2 I + J) ** (2 + 3 K)
```

```
Out[5]=(2 + 7 I) - 4 J + 3 K
```

此外也可以进行其他运算,如指数运算 `E^Quaternion[2, 3, 1, 6]` 等。更多用法参考帮助文档 `tutorial/Quaternions`。

3.4 复变分析和微积分计算

此处内容,详见 `guide/Calculus`, 以下举几个简单的函数和例子,其他函数以及用法和选项参考帮助文档:

1. 极限 (Limit)

```
In[1]:=Limit[(1+x/n)^n, n->Infinity]
```

```
Out[1]=e^x
```

2. 求和 (Sum)

```
In[1]:=Sum[i^2, {i, 1, n}]
```

```
Out[1]= $\frac{1}{6}n(n+1)(2n+1)$ 
```

```
In[2]:=Sum[1/i^2, {i, 1, Infinity}]
```

```
Out[2]= $\frac{\pi^2}{6}$ 
```

```
In[3]:=Sum[1/(j^2 (i + 1)^2), {i, 1, Infinity}, {j, 1, i}]
```

```
Out[3]= $\frac{\pi^4}{120}$ 
```

3. 乘积 (Product)

```
In[1]:=Product[1 - 1/i^4, {i, 2, Infinity}]
```

```
Out[1]= $\frac{\text{Sinh}[\pi]}{4\pi}$ 
```

4. 求导 (D, Derivative)

求一阶导数:

```
In[1]:=D[x Sin[x], x]
```

```
Out[1]=x Cos[x] + Sin[x]
```

求二阶导数:

```
In[2]:=D[x Sin[x], {x, 2}]
```

```
out[2]=2 Cos[x] - x Sin[x]
```

混合偏导:

```
In[3]:=D[x^y, x, y]
```

```
Out[3]= $x^{y-1} + y x^{y-1} \text{Log}[x]$ 
```

散度:

```
In[4]:=D[x Sin[y] Cos[z], {{x, y, z}}]
```

```
Out[4]:={Cos[z] Sin[y], x Cos[y] Cos[z], -x Sin[y] Sin[z]}
```

海森矩阵:

```
In[5]:=D[x^2 y^3, {{x, y}, 2}]
```

```
Out[5]:={{2y^3, 6xy^2}, {6xy^2, 6x^2y}}
```

雅克比矩阵:

```
In[6]:=D[{x y^2, x^3 y}, {{x, y}}]
```

```
Out[6]:={{y^2, 2x y}, {3x^2 y, x^3}}
```

如果是表示一个抽象函数的 n 阶导, 对应的函数是 `Derivative`, `Derivative[n1, n2, ...][f]` 代表表示对 f 的第一个参数微分 n_1 次, 再对第二参数微分 n_2 次, 如此依次进行后得到的函数。如

```
In[7]:=Derivative[1][Sin]
```

```
Out[7]=Cos[#1] &
```

```
In[8]:=Derivative[n][Sin]
```

```
Out[8]=Sin[ $\frac{n\pi}{2}$  + #1] &
```

```
In[9]:=Derivative[0, 1][BesselJ]
```

```
Out[9]=1/2 (BesselJ[-1 + #1, #2] - BesselJ[1 + #1, #2]) &
```

对于这种使用方式, `Derivative` 会返回一个纯函数。函数 `Derivative` 有个语法糖, 常见的 $y'[x]$ 就是 `Derivative[1][y][x]`。这一语法糖可以直接作用在纯函数上, 如

```
In[10]:=Derivative[1][y]
```

```
Out[10]=y'
```

```
In[11]:=Sin'
```

```
Out[11]=Cos[#1] &
```

```
In[12]:=#^2 &'
```

```
Out[12]=2 #1 &
```

5. 积分 (Integrate)

不定积分:

```
In[1]:=Integrate[x Sin[x], x]
```

```
Out[1]=-x Cos[x] + Sin[x]
```

定积分:

```
In[2]:=Integrate[Exp[-x^2], {x, -Infinity, Infinity}]
```

```
Out[2]= $\sqrt{\pi}$ 
```

计算柯西主值:

```
In[3]:=Integrate[1/x, {x, -1, 2}, PrincipalValue->True]
```

```
Out[3]:=Log[2]
```

多重积分:

```
In[4]:=Integrate[x Sin[y + x], x, y]
```

```
Out[4]=-Cos[x + y] - x Sin[x + y]
```

多重定积分:

```
In[5]:=Integrate[Exp[-2 x^2 - 3 y^2], {x, -1, 1}, {y, -1, 1}]
```

```
Out[5]= $\frac{\pi \text{Erf}[\sqrt{2}] \text{Erf}[\sqrt{3}]}{\sqrt{6}}$ 
```

在区域上积分:

```
In[6]:=Integrate[x^2 - 2 y + z, Element[{x, y, z}, Ball[]]]
```

```
Out[6]= $\frac{4\pi}{15}$ 
```

上述的 `Element` 函数也可以用符号 \in 表示, Mathematica 中的输入方式为 `[ESC][E][L][ESC]`, 如

```
In[7]:=Integrate[x^2 - 2 y + z, {x, y, z} ∈ Ball[]]
```

```
Out[7]= $\frac{4\pi}{15}$ 
```

6. 幂级数展开 (Series)

```
In[1]:=Series[Exp[x], {x, 0, 3}]
```

```
Out[1]=1 + x +  $\frac{x^2}{2}$  +  $\frac{x^3}{6}$  + O[x4]
```

```
In[2]:=Series[Sin[x + y], {x, 0, 2}, {y, 0, 2}]
```

```
Out[2]=(y + O[y3]) + x (1 -  $\frac{y^2}{2}$  + O[y3]) + x2 (- $\frac{y}{2}$  + O[y3]) + O[x3]
```

使用 Normal 函数可以把含阶的幂级数展开式变成普通的表达式:

```
In[3]:=Normal//Simplify
```

```
Out[3]=- $\frac{x^2 y}{2}$  -  $\frac{x y^2}{2}$  + x + y
```

Series 也可在无穷远处展开:

```
In[4]:=Series[Exp[1/x], {x, Infinity, 3}]
```

```
Out[4]=1 +  $\frac{1}{x}$  +  $\frac{1}{2x^2}$  +  $\frac{1}{6x^3}$  + O[ $\frac{1}{x}$ ]4
```

7. 傅里叶级数和变换

Mathematica 中与此相关的函数较多, 详见帮助文档 guide/FourierAnalysis, 以下举几例:

(1) 傅里叶变换 (FourierTransform)

```
In[1]:=FourierTransform[Exp[-t^2] Sin[t], t, s]
```

```
Out[1]= $\frac{I (-1 + \text{Cosh}[s] + \text{Sinh}[s]) (\text{Cosh}[\frac{1}{4}(1+s)^2] - \text{Sinh}[\frac{1}{4}(1+s)^2])}{2\sqrt{2}}$ 
```

由于不同科学和技术领域中对于傅里叶变换的定义有些不一样, 不同的定义选择可以使用选项 FourierParameters 指定, 设置 FourierParameters->{a,b}, 由 FourierTransform 计算的傅里叶变换公式是 $\sqrt{\frac{|b|}{(2\pi)^{1-a}}} \int_{-\infty}^{\infty} f(t) e^{ibt\omega} dt$, 详见文档。

(2) 逆傅里叶变换 (InverseFourierTransform)

```
In[2]:=InverseFourierTransform[1/(1 + s^2), s, t]
```

```
Out[2]=e-Abs[t]  $\sqrt{\frac{\pi}{2}}$ 
```

逆傅里叶变换的选项类似于傅里叶变换。

(3) 傅里叶级数展开 (FourierSeries)

这个函数是把函数展开成复指数幂形式的傅里叶级数:

```
In[3]:=FourierSeries[t/2, t, 3]
```

```
Out[3]= $\frac{1}{2}I e^{-I t} - \frac{1}{2}I e^{I t} - \frac{1}{4}I e^{-2 I t} + \frac{1}{4}I e^{2 I t} + \frac{1}{6}I e^{-3 I t} - \frac{1}{6}I e^{3 I t}$ 
```

(4) 傅里叶级数展开系数 (FourierCoefficient)

FourierCoefficient 给出的是展开成复指数幂形式时的系数 $\frac{1}{2\pi} \int_{-\pi}^{\pi} f(t) e^{-int} dt$:

```
In[4]:=FourierCoefficient[t^2 + t, t, 5]
```

```
Out[4]=- $\frac{2}{25}$  -  $\frac{1}{5}$ 
```

除了上述所讲之外, Mathematica 还可以把函数展开成傅立叶正弦级数、傅立叶余弦级数以及给出它们的系数等, 此处不再细述。

8. 拉普拉斯变换

(1) 拉普拉斯变换 (LaplaceTransform)

```
In[1]:=LaplaceTransform[t^4 Sin[t], t, s]
```

```
Out[1]= $\frac{24 (1 - 10 s^2 + 5 s^4)}{(1+s^2)^5}$ 
```

(2) 逆拉普拉斯变换 (InverseLaplaceTransform)

```
In[2]:=InverseLaplaceTransform[Log[s]^2/s, s, t]
```

```
Out[2]=- $\frac{\pi^2}{6}$  + (-EulerGamma - Log[t])2
```

至于其他的积分变换 (如 Mellin 变换, Hankel 变换等), 见帮助文档 guide/IntegralTransforms, 此处不再细述。

9. 留数 (Residue)

```
In[1]:=Residue[Gamma[z] Gamma[z - 1] Gamma[z - 2], {z, 0}]
```

```
Out[1]= $\frac{1}{8}(-17 + 30 \text{ EulerGamma} - 18 \text{ EulerGamma}^2 - \pi^2)$ 
```

10. 卷积 (Convolve)

```
In[1]:=Convolve[Exp[-x] UnitStep[x], Sin[x], x, y]
```

```
Out[1]= $\frac{1}{2}(-\text{Cos}[y] + \text{Sin}[y])$ 
```

11. 变分法

Mathematica 内置变分法程序包, 在使用前先要加载:

```
In[1]:=<< VariationalMethods`
```

比如计算泛函 $F = \int_{x_{\min}}^{x_{\max}} y(x) \sqrt{y'(x)^2 + 1} dx$ 的变分:

```
In[2]:=VariationalD[y[x] Sqrt[1 + Derivative[1][y][x]^2], y[x], x]
```

```
Out[2]= $\frac{1+y'[x]^2-y[x]y''[x]}{(1+y'[x]^2)^{3/2}}$ 
```

再比如由拉格朗日量计算单摆的欧拉拉格朗日方程:

```
In[3]:=EulerEquations[1/2 m l^2 theta'[t]^2 + m g l Cos[theta[t]], theta[t], t]
```

```
Out[3]=-1 m (g Sin[theta[t]] + l theta''[t]) == 0
```

更多关于变分法程序包的内容, 参考帮助文档 VariationalMethods/tutorial/VariationalMethods。

12. 线积分, 面积分与围道积分

在版本 13.3 中, Mathematica 引入了新的积分函数, 包括线积分 (LineIntegrate)、面积分 (SurfaceIntegrate) 以及围道积分 (ContourIntegrate), 如

半圆弧上的积分:

```
In[1]:=LineIntegrate[2 x, {x,y} ∈ Circle[{0, 0}, 1, {0, Pi/2}]]
```

```
Out[1]:=2
```

标量函数在球面的积分:

```
In[2]:=SurfaceIntegrate[x^2, {x, y, z} ∈ Sphere[]]
```

```
Out[2]= $\frac{4\pi}{3}$ 
```

向量场在球面的积分:

```
In[3]:=SurfaceIntegrate[{x^2, x - y, 1}, {x, y, z} ∈ Sphere[]]
```

```
Out[3]= $-\frac{4\pi}{3}$ 
```

在本课程中, 可能运用较多的是围道积分, 如在圆心在原点, 半径为 R 的围道上积分:

```
In[4]:=ContourIntegrate[(z^3 (1 - 3 z))/((3 z + 2) (1 + 2 z^4)), z ∈ Circle[{0, 0}, R]]
```

```
Out[5]=
$$\begin{cases} -\frac{48 i \pi}{113} & \frac{2}{3} < R \leq \frac{1}{2^{1/4}} \\ -i \pi & R > \frac{1}{2^{1/4}} \\ 0 & \text{True} \end{cases} \quad \text{if } 3 R \neq 2 \ \&\& \ R \neq \frac{1}{2^{1/4}}$$

```

Mathematica 按照 R 的不同值给出了不同结果。如果需要提取上面的结果表达式, 可以用 InputForm 查看上述内容的具体表达式, 再使用 Part 函数提取。再比如, 计算一个复平面的上半部分闭合的围道积分, 从实轴上半绕过极点 -2, 从实轴下半绕过极点 2:

```
In[5]:=ContourIntegrate[I E^(-I p)/(p^2 - 4), p ∈ {"UpperSemicircle", 2, -2}]
```

```
Out[5]= $-\frac{1}{2}e^{-2}i\pi$ 
```

13. 渐进计算

在版本 11.3 以及后续版本中, Mathematica 陆续引入了一系列渐进计算的函数, 包括 Asymptotic、AsymptoticIntegrate、AsymptoticSum 等, 具体可以参考帮助文档 guide/Asymptotics。例如在无穷远处展开 $\Gamma(x)$ 的首项:

```
In[1]:=Asymptotic[Gamma[x], {x, Infinity, 1}]
Out[1]=e-x√(2πx)-1/2+x
在 x = 0 处展开  $\frac{\sin(x)}{1+x}$  到无穷阶:
In[2]:=Asymptotic[Sin[x]/(1 + x), {x, 0, Infinity}]
Out[2]= 
$$\sum_{n=0}^{\infty} \frac{i x^n (i^n \text{HypergeometricPFQ}[\{1, -n\}, \{\}, -i] - (-i)^n \text{HypergeometricPFQ}[\{1, -n\}, \{\}, i])}{2 n!}$$

```

计算 x^x 的积分在 $x = 0$ 处的渐进展开到 2 阶:

```
In[3]:=AsymptoticIntegrate[x^x, x, {x, 0, 2}]
Out[3]= x +  $\frac{1}{4}x^2(-1 + 2 \text{Log}[x])$ 
计算  $\sum_{k=1}^n \frac{1}{k}$  在  $n = \infty$  处的展开到 2 阶:
In[4]:=AsymptoticSum[1/k, {k, 1, n}, {n, Infinity, 2}]
Out[4]=EulerGamma -  $\frac{1}{12n^2} + \frac{1}{2n} + \text{Log}[n]$ 
更多的使用方法参考具体函数的帮助文档。
```

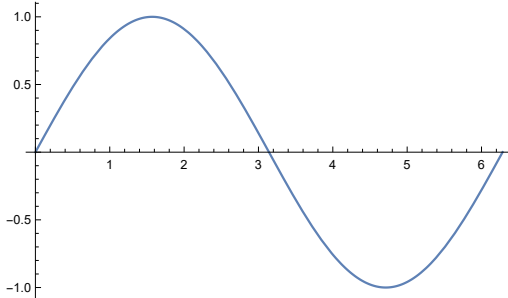
4 Mathematica 中的绘图

这里讲 Mathematica 中简单的函数可视化。

正如前面所说, Mathematica 里面所有对象都是表达式, 图形也是。虽然在前端是一个图形, 但是内核处理和传递到前端都是表达式。Mathematica 中图形表达式对结构详见帮助文档 `tutorial/TheStructureOfGraphicsAndSoundOverview`, 至于数据可视化以及使用低级函数 `Graphics` 或 `Graphics3D` 画图不在此处讨论范围之列。

应该说, Mathematica 中函数可视化是所有数学软件中最简单的, 只需要给定要作图的函数以及作图的区域, 输入对应的表达式, 剩下的工作 Mathematica 都会自动完成, 比如:

```
In[1]:=Plot[Sin[x], {x, 0, 2 Pi}]
Out[1]=
```



与函数可视化有关的所有函数在帮助文档 `guide/FunctionVisualization` 处, 以下来做简明介绍。

通常 Mathematica 用于绘图的函数的选项用于指定绘图样式, 而且一般画同一类型图形的函数选项绝大部分都是一样的, 举例时只以一个函数为例。

4.1 二维绘图

最基本的画一元函数图形的函数是 `Plot`, 具体用法请查询帮助文档, 此处仅举几例, 如绘制 $\Gamma(x)$ 在 $(-5, 5)$ 内的图像:

```
|| Plot[Gamma[x], {x, -5, 5}]
```

或者是同时绘制几个函数的图像:

```
|| Plot[{Sin[x], Cos[x], Tan[x]}, {x, -2 Pi, 2 Pi}]
```

函数 `Plot` 的选项是及其多的, 共有 128 个, 绝大部分选项看名称就知道作用, 比如 `AspectRatio` 控制图像的宽高比, `AxesOrigin` 控制图像坐标轴原点位置, `PlotRange` 控制绘图范围等。选项的作用可以参阅 `Plot` 的帮助文档, 当然也可以去选项的帮助文档单独查看某个选项的具体使用方法, 如:

```
Plot[Gamma[x], {x, -5, 5}, PlotRange -> {-10, 10},
  PlotLabel -> Style[Gamma[x], Black, Bold, 16],
  AxesStyle -> Directive[Arrowheads[0.03], Italic, Thick],
  AxesLabel -> (Style[#, Black, Bold, 16] & /@ {x, y}),
  ColorFunction -> Hue]
```

由于 Mathematica 的版本变化, 以前 4.0 版本需要加载额外程序包才能进行的绘图, 现在也可以直接绘制了, 比如参数函数绘图, 是 ParametricPlot:

```
ParametricPlot[{Sin[u], Sin[2 u]}, {u, 0, 2 Pi}]
```

极坐标绘图, 是 PolarPlot:

```
PolarPlot[Sin[3 t], {t, 0, Pi}]
```

等高线作图, 有 ContourPlot:

```
ContourPlot[Cos[x] + Cos[y], {x, 0, 4 Pi}, {y, 0, 4 Pi}]
ContourPlot[Cos[x] + Cos[y] == 1/2, {x, 0, 4 Pi}, {y, 0, 4 Pi}]
```

密度图, 有 DensityPlot:

```
DensityPlot[Sin[x] Sin[y], {x, -4, 4}, {y, -3, 3}]
```

至于绘制向量图, 有 StreamPlot (流线图)、VectorPlot (矢量图)、StreamDensityPlot (向量场和标量场函数的叠加图) 等, 不再细述。

4.2 三维绘图

最基本的画二元函数图形的函数是 Plot3D, 比如画 $\sin(xy)$ 的图像:

```
Plot3D[Sin[x y], {x, -Pi, Pi}, {y, -Pi, Pi}]
```

画三维图的选项和画二维图的选项有点区别, 比如控制图形比例的选项从二维图的 AspectRatio 在三维图里就变成了 BoxRatios 等。Plot3D 的选项共有 103 个, 详见参考文档。

对于复变函数的图形, 由于我们所在的空间只有三维, 所以我们只能分别画出实部和虚部 (或者模和辐角), 比如:

```
pic1=Plot3D[Re[Sin[x + I y]], {x, 0, 5 Pi}, {y, 0, 2}, PlotStyle -> Green]
pic2=Plot3D[Im[Sin[x + I y]], {x, 0, 5 Pi}, {y, 0, 2}, PlotStyle -> Pink]
Show[pic1, pic2]
```

其中 PlotStyle 是指定图形的样式, 这里是设置了颜色; Show 是把图形合并到一张图片的函数。但是 Show 只能合并相同维度的图形, 比如合并几张二维图或者几张三维图, 但是二维图和三维图就不能够合并在一起。如果有这样的需求的话, 就得使用模式匹配或者使用 Texture 来拼装图形, 这要求对 Mathematica 掌握的程度比较高, 此处不再细述。当然, 也可以直接在一张图里面同时画实部和虚部, 比如

```
Plot3D[{Re[Sin[x + I y]], Im[Sin[x + I y]]}, {x, 0, 5 Pi}, {y, 0, 2}]
```

这时候 Mathematica 就会自动给不同的图着不同的颜色。若想以等高线图或者密度图的形式画复变函数的图形, 可以考虑前面提到的 ContourPlot 和 DensityPlot, 如:

```
ContourPlot[Abs[Sin[x + I y]], {x, 0, 5 Pi}, {y, 0, 2}]
```

ContourPlot 和 DensityPlot 有时会因为采样的原因导致边缘不光滑, 如果出现边缘不平滑的情况, 可以添加选项 PlotPoints, 比如:

```
ContourPlot[Abs[Sin[x + I y]], {x, 0, 5 Pi}, {y, 0, 2}, PlotPoints -> 50]
```

当然对所有的绘制函数图像的函数, 选项 PlotPoints 都是可用的。此外, 由于选项 ColorFunction 的存在, 我们可以给一个三维图着色, 这意味着我们可以用高度和颜色分别表示复变函数的实部和虚部 (或者模和辐角), 如:

```
Plot3D[Abs[Sin[x + I y]], {x, 0, 5 Pi}, {y, 0, 2},
ColorFunction ->
Function[{x, y}, Hue[Arg[Sin[x + I y]]/(2 Pi) + 1/2]],
ColorFunctionScaling -> False]
```

这里就用了高度表示模,颜色表示辐角。选项 `ColorFunction` 给出的必须是一个纯函数,这里就是 `Function[{x, y}, Hue[Arg[Sin[x + I y]]/(2 Pi) + 1/2]]`, 其中 `Hue[x]` 是用色相表示的颜色, `x` 的取值为 $0 \sim 1$ 之间, 由于 `Arg` 取值为 $(-\pi, \pi]$, 所以除以 2π 再减去 $1/2$ 保证落在 $0 \sim 1$ 之间。注意当使用 `ColorFunction` 指定颜色函数时,要加上选项 `ColorFunctionScaling -> False`, 否则就会把颜色函数的自变量按尺度按比例缩放放到 0 和 1 之间,得不到我们想要的结果。

画三维图还有其他诸多函数,比如 `ParametricPlot3D` 绘制三维参数曲线或者曲面, `ContourPlot3D` 绘制三维等值面, `SphericalPlot3D` 绘制三维球面图形, `RevolutionPlot3D` 绘制三维旋转图形等,详见帮助文档 `guide/FunctionVisualization`, 不再细述。

除了 Mathematica 内置的绘图函数外, Wolfram 函数库¹里也有一些独立开发者提供的补充函数,使用这里面的函数需要联网。在本课程里比较有用的一个绘图函数是绘制复变函数的黎曼面,这个函数的名称为 `RiemannSurfacePlot3D`,其在线文档在 <https://resources.wolframcloud.com/FunctionRepository/resources/RiemannSurfacePlot3D/>。比如绘制函数 \sqrt{z} 的黎曼面的代码为

```
ResourceFunction["RiemannSurfacePlot3D"][w == Sqrt[z], Re[w], {z, w}]
```

第一次运行的时候需要下载这个函数的源码,运行速度稍慢,之后运行就会变快。更多的黎曼面绘制例子可以参考该函数的在线文档。

5 Mathematica 中的 (偏) 微分方程

Mathematica 具有很强大的符号和数值解 (偏) 微分方程的功能,就在这里做简要介绍。而与此有关的函数,主要是 `DSolve` 和 `NDSolve`。这两个函数的功能非常强大,可以应用于各种场景,具体内容请参考帮助文档,这里只是举例介绍。

5.1 符号解 (DSolve, DSolveValue)

```
In[1]:=Clear["Global`*"]
```

解常微分方程:

```
In[2]:=DSolve[y'[x] + 2 y'[x] + y[x] == 0, y[x], x]
```

```
Out[2]={y[x] -> e^-x C[1] + e^-x x C[2]}
```

`C[1]`和`C[2]`是常数,如果把上面的`y[x]`换成`y`的话,返回的就是一个纯函数:

```
In[3]:=DSolve[y'[x] + 2 y'[x] + y[x] == 0, y, x]
```

```
Out[3]={y -> Function[{x}, e^-x C[1] + e^-x x C[2]]}
```

这样的好处是可以调用计算的结果,比如我要计算`y[0]`:

```
In[4]:=y[0]/.%[[1,1]]
```

```
Out[4]=C[1]
```

或者求导:

```
In[5]:=y'[x]/.%[[1,1]]
```

```
Out[5]=- e^-x C[1] + e^-x C[2] - e^-x x C[2]
```

`y[0]/.%2[[1,1]]` 这个代码首先把 `y[0]` 中的 `y` 替换了变成 `Function[{x}, e^-x C[1] + e^-x x C[2]][0]`, 然后计算得结果为 `C[1]`。

如果用 `DSolveValue` 的话,就会直接返回计算的结果而不是一个规则,比如:

```
In[6]:=DSolveValue[y'[x] + 2 y'[x] + y[x] == 0, y, x]
```

```
Out[6]:=Function[{x}, e^-x C[1] + e^-x x C[2]]
```

加上边界条件或者初始条件:

¹<https://resources.wolframcloud.com/FunctionRepository/>

```
In[7]:=DSolve[{y'[x] + 2 y[x] + y[x] == 0, y[1] == 1, y'[1] == 1}, y, x]
Out[7]={y -> Function[{x}, e1-x (-1 + 2 x)]]}
```

对于绝大部分可以解析解的常微分方程, Mathematica 都是可以给出解析解的。

偏微分方程:

```
In[8]:=DSolve[c^2 D[u[x, t], x, x] - D[u[x, t], t, t] == 0, u[x, t], {x, t}]
Out[8]={u[x, t] -> C[1][t -  $\frac{x}{\sqrt{c^2}}$ ] + C[2][t +  $\frac{x}{\sqrt{c^2}}$ ]}}
```

这里显示了 Mathematica 的一个重要特性, 那就是如果你没有指定范围, Mathematica 中的符号计算默认所有的符号都是复数, 所以上面出现了 $\sqrt{c^2}$, 以至于有时候计算出现了 `ConditionalExpression` 而且计算速度较慢, 解决的办法是加上一些假设:

```
In[9]:=Assuming[c > 0, DSolve[c^2 D[u[x, t], x, x] - D[u[x, t], t, t] == 0,
  u[x, t], {x, t}]]
Out[9]={u[x, t] -> C[1][t -  $\frac{x}{c}$ ] + C[2][t +  $\frac{x}{c}$ ]}}
```

当然, 这里的 `C[1]` 和 `C[2]` 是任意形式的函数。

也可以指定边界条件和初始条件解偏微分方程:

```
In[10]:=DSolve[{Laplacian[u[x, y], {x, y}] == 0, u[0, y] == y, u[a, y] == 0,
  u[x, 0] == 0, u[x, b] == 0}, u[x, y], {x, 0, a}, {y, 0, b}]
```

```
Out[10]={u[x, y] ->  $\sum_{K[1]=1}^{\infty} \frac{2(-1)^{K[1]+1} b \text{Csch}[\frac{\pi a K[1]}{b}] \text{Sin}[\frac{\pi y K[1]}{b}] \text{Sinh}[\frac{\pi(a-x) K[1]}{b}]}{\pi K[1]}$ }}
```

```
In[11]:=DSolve[{D[u[x, t], t, t] - c^2 D[u[x, t], x, x] == 0,
  u[x, 0] == x (1 - x), (D[u[x, t], t] /. t -> 0) == 0,
  u[0, t] == 0, u[1, t] == 0}, u[x, t], {x, t}]
```

```
Out[11]={u[x, t] ->  $\sum_{K[1]=1}^{\infty} -\frac{4(-1+(-1)^{K[1]}) \text{Cos}[\pi \sqrt{c^2} t K[1]] \text{Sin}[\pi x K[1]]}{\pi^3 K[1]^3}$ }}
```

我们可以通过把求和上限的无穷大替换成有限的数字实现对于求和的截断, 如

```
|| % /. Infinity -> 5 // Activate
```

其中函数 `Activate` 的作用是激活给出的解中未激活求和的 `Sum`。在 Mathematica 版本 14 中, 可以用函数 `TruncateSum` 求截断和, 如 `TruncateSum[%%, 5]` 可以给出上述求和前五项之和的表达式。

不过 Mathematica 解带有初始条件和边界条件的偏微分方程会有些限制, 比如所要解的偏微分方程必须是齐次方程, 然后自变量只能有两个, 而且必须是矩形区域。更多符号求解偏微分方程的内容, 可以参考帮助文档 `tutorial/SymbolicSolutionsOfPDEs`。

5.2 级数解

我们可以用已知的函数手动求解常微分方程的级数解, 以下以常微分方程 $y''(x) - 6y(x)^2 - x = 0$, $y'(0) = 0$, $y(0) = 1$ 为例。

首先定义方程和初始条件以及这个微分方程的算符:

```
In[1]:=ode = y'[x] - 6 y[x]^2 - x == 0;
In[2]:=init = {y'[0] == 0, y[0] == 1};
In[3]:=op = D[#, {x, 2}] - 6 #^2 - x &;
```

接下来, 我们设要求解的级数解的表达式为 `ys`, 通过 `Series` 函数可以很快写出 `ys` 的表达式, 这里只展开到 5 项:

```
In[4]:=ys = Series[y[x], {x, 0, 5}]
Out[4]=y[0] + y'[0]x +  $\frac{1}{2}y''[0]x^2 + \frac{1}{6}y^{(3)}[0]x^3 + \frac{1}{24}y^{(4)}[0]x^4 + \frac{1}{120}y^{(5)}[0]x^5 + O[x^6]$ 
```

接下来, 我们将 `ys` 的表达式带入方程, 再加上初始条件, 用函数 `SolveAlways` 求解, 得到

```
In[5]:=sol = SolveAlways[Join[{op[ys] == 0}, init], x]
Out[5]={y(4)[0] -> 72, y(5)[0] -> 12, y(3)[0] -> 1, y'[0] -> 6, y'[0] -> 0, y[0] -> 1}}
```

将上面的结果带入 `ys` 的表达式, 即可得到方程的级数解

```
In[6]:=ys /. sol[[1]]
Out[6]=1 + 3x^2 +  $\frac{x^3}{6} + 3x^4 + \frac{x^5}{10} + O[x^6]$ 
```


可以将上述求解思路打包成一个函数:

```
ODESeriesSolution[ode_Equal, inits_List, y_, x_, order_ : 5] :=
Block[{ys, equation, sol},
  ys = Series[y[x], {x, 0, order}];
  equation = ode /. y -> (Evaluate[ys /. x -> #] &);
  sol = SolveAlways[Join[{equation}, inits], x];
  ys /. sol // First
]

ODESeriesSolution[y''[x]-6 y[x]^2-x == 0, {y'[0]==0, y[0]==1}, y, x, 20]

ODESeriesSolution[y''[x]-6 y[x]^2-x == 0, {}, y, x, 5]
```

在引入了渐进求解的函数之后, 级数解微分方程也可以通过 `AsymptoticDSolveValue` 求解, 比如上述方程求解为

```
In[7]:=AsymptoticDSolveValue[{ode, init}, y[x], {x, 0, 5}]
Out[7]=1 + 3x^2 +  $\frac{x^3}{6}$  + 3x^4 +  $\frac{x^5}{10}$ 
```

5.3 数值解 (NDSolve, NDSolveValue)

Mathematica 具有强大的数值解微分方程的能力, 以下为简要介绍。

Mathematica 可以求解几乎所有类型的常微分方程组, 不过数值求解微分方程必须要求求解的区域是有限的, 而且要求除了自变量外, 所有参数都是具体的数, 而且必须给定能够唯一确定方程解的初始条件和边界条件。对于常微分方程的数值解, 写法和微分方程基本一样, 只是要加上求解区域:

```
Clear["Global`*"]
sol=NDSolveValue[{y'[x] + 2 y[x] + y[x] == 0, y[1] == 1,
  y'[1] == 1}, y, {x, 1, 10}]
```

所得到的是一个纯函数, 如果想要求在某一点的值的话, 可以这样:

```
sol[2]
(*1.10364*)
```

或者绘图:

```
Plot[sol[x], {x, 1, 10}]
```

然而有时候在数值解常微分方程 (组) 的时候, 可能会因为方程 (组) 的特性导致求解失败, 比如方程 (组) 是刚性的, 或者是求解过程中遇到了方程的奇点。这时候可以添加选项 `Method` 来指定针对特定类型的常微分方程 (组) 的算法, 相关内容比较庞大, 想深入了解的话需要知道常微分方程的算法, 至于 Mathematica 中对于 `NDSolve` 的选项说明和一些算法注释可以参考帮助文档 `tutorial/NDSolveOverview`, 不再细述。

如果是数值解偏微分方程的话, 情况就有点复杂。目前 Mathematica 可以解二阶线性偏微分方程组, 至于一些非线性的方程有时候会求解失败。Mathematica 版本 10.0 以后引入了函数狄利克雷条件 (`DirichletCondition`) 和诺依曼边值 (`NeumannValue`), 数值解偏微分方程的边界条件必须由这两个函数指定。而且因为这两个函数的存在, Mathematica 中的偏微分方程可以在任意形状区域内数值求解。而且现在 Mathematica 内置了有限元程序包, 可以使用有限元方法求解偏微分方程, 详见帮助文档 `FEM-Documentation/tutorial/FiniteElementOverview`, 内容过于庞大, 不再细述。以下以一个矩形区域的波动方程为例。

一均匀各向同性弹性薄膜, $0 \leq x \leq 1$, $0 \leq y \leq 1$, 四周夹紧。初始位移为 $xy(1-x)(1-y)$, 初始速度为 0, 波速为 1, 求解这个系统。

首先求解这个方程:

```
sol = NDSolveValue[{D[u[t, x, y], t, t] - Laplacian[u[t, x, y], {x, y}] == 0, u
  [0, x, y] == x y (1 - x) (1 - y), Derivative[1, 0, 0][u][0, x, y] == 0,
  DirichletCondition[u[t, x, y] == 0, True]}, u, {t, 0, 10}, Element[{x, y},
  Rectangle[{0, 0}, {1, 1}]], Method -> {"PDEDiscretization" -> {"
  MethodOfLines", "SpatialDiscretization" -> {"FiniteElement"}}}]
```

这里就在选项里设定了方法为有限元, 计算得到的`sol[t,x,y]`就是方程的解, 为了可视化结果, 我们可以先画出不同时刻的图像的列表, 然后把这个列表中的图片做成动画:

```
list = Table[Plot3D[sol[t, x, y], {x, 0, 1}, {y, 0, 1}, AxesOrigin -> {0, 0, 0},
  Boxed -> False, SphericalRegion -> True, PlotRange -> {{0, 1}, {0, 1},
  {-1/12, 1/12}}], {t, 0, 10, 0.1}];
ListAnimate[list]
```

或者导出为 gif 文件:

```
SetDirectory@NotebookDirectory[];
Export["wave.gif", list]
```

以上内容涉及 Mathematica 的许多内容比如列表操作以及文件导入导出等, 这里只举了一个例子, 有兴趣的同学可以查询以上代码里面所涉及到的函数。而且 Mathematica 中求解的区域可以是一些简单规则区域的组合, 也可以是随便得到的一个区域, 限于篇幅不再细述。

当然 Mathematica 还有其他的求解 (偏) 微分方程的函数, 比如求解格林函数 (`GreenFunction`), 求解微分方程的特征函数和特征值 (`DEigensystem`), 详见帮助文档, 不再细述。