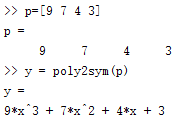
MATLAB的数学运算

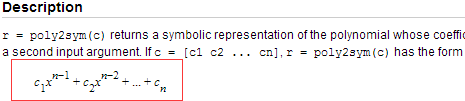
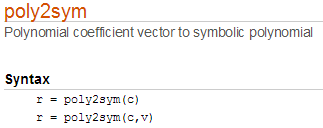
# 多项式与插值

## 多项式的表示

**多项式利用一个行向量表示**，向量中的元素就是多项式的系数，按照降序排列。

如 **9x3+7x2+4x+3** 可以表示为向量p=[9 7 4 3]



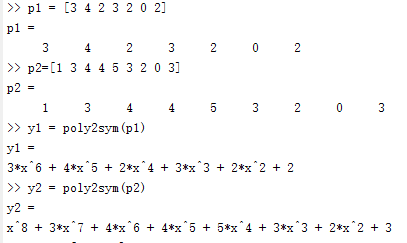


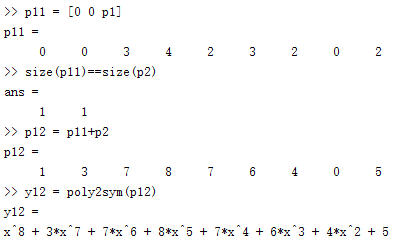
## 多项式的四则运算

### 多项式的加减运算

多项式的加减就是对应项系数的加减，因此可以通过向量的加减来实现。

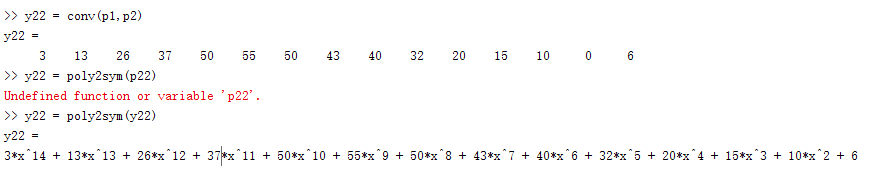
不过，**向量必须具有相同的长度，短的向量前面补0**。

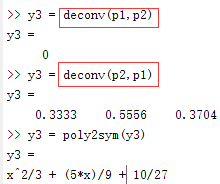




### 多项式的乘除运算

多项式的乘法和除法是多项式系数向量之间的卷积运算，可以通过**conv**来完成。多项式的除法为乘法的逆运算，可以通过**反卷积函数deconv**来实现。





## 多项式的其他运算

roots 多项式求根

polyval 多项式求值

polyvalm 矩阵多项式求值

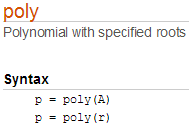
polyder 多项式求导

poly 求矩阵的特征多项式；或者求一个多项式，其根为指定的数值

polyfit 多项式曲线拟合

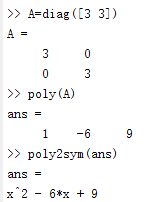
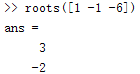
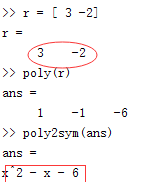
residue 求解余项

### poly ： 求一个矩阵的特征多项式

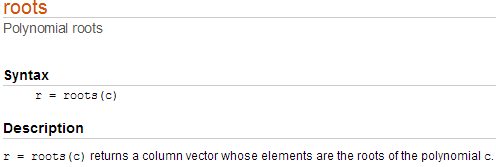


poly(A) 返回矩阵A的特征多项式；

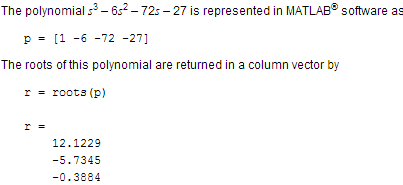
poly(r) 返回**所有的根是r向量元素**的多项式的系数。

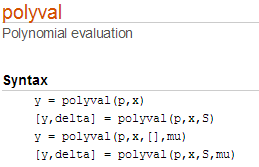
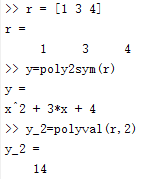
### roots函数 : 求解多项式的根。



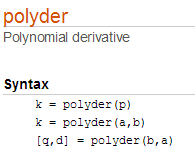
roots 用于求解多项式的根。输入参数为**多项式的系数组成的行向量**，返回值为多项式的**根组成的列向量**。

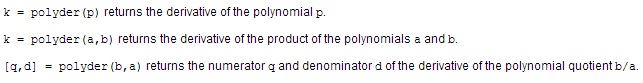


### polyval ： 用于多项式求值。

### polyder ： 对多项式求导。

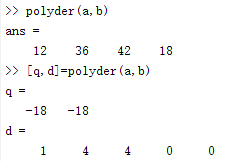
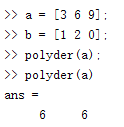




k=polyder(p)： 表示对p对应的多项式的求导

k=polyder(a,b) : 表示对a和b对应多项式的乘积求导；

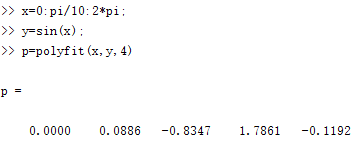
[q ,d]=polyder(a,b) : 表示对a和b对应的多项式的商求导，结果为q/d 。前面的q和a都是分子，后面的d和b都是分母。

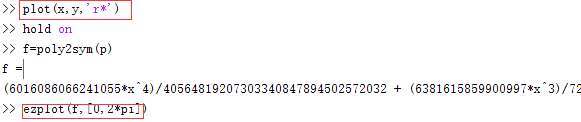


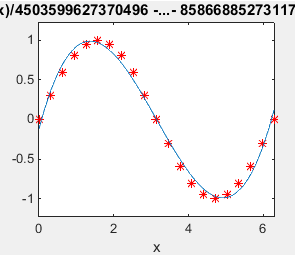
### 曲线拟合：多项式曲线拟合

函数polyfit 给出在最小二乘意义下最佳拟合系数。

调用方法： p = polyfit(x,y,n) ,其中x、y分别为待拟合数据的x坐标和y坐标，n用于指定返回多项式的次数。n决定了p的长度，p的长度是n=1，p是生成拟合多项式的系数。







这里利用四阶多项式拟合正弦函数的结果。

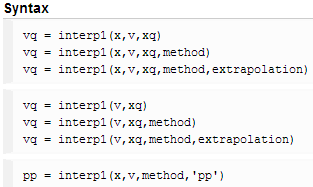
## 数据插值

根据已知数据推断未知数据，就需要使用数据插值。

**插值运算**：根据已有数据的分布规律，找到一个函数表达式可以连接已知的各点，并用这一函数表达式来预测已有数据两点之间任意位置上的数据。

MATLAB提供了对数组的任意一维进行插值的工具。

一维插值在曲线拟合和数据分析中具有重要的地位。在MATLAB中，一维插值主要是由**函数interp1**实现。



x v为采用数据的x坐标和y坐标，xq为待插值的位置。method为采用的插值方法。返回值为点xq处的插值结果。

method方法有：

**nearest 最近邻插值**

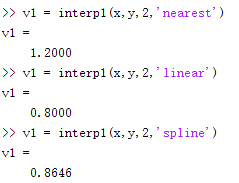
**linear 线性插值**

**spline 三次样条插值**

**pchip 或 cubic 三次插值**

示例：





插值算法比较：

**最近邻插值**：将插值点xi的值设置为距离最近的点的对应值。

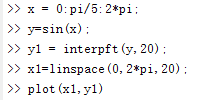
**线性插值法**：用分段线性函数拟合已有数据，返回拟合函数在xi处的值。

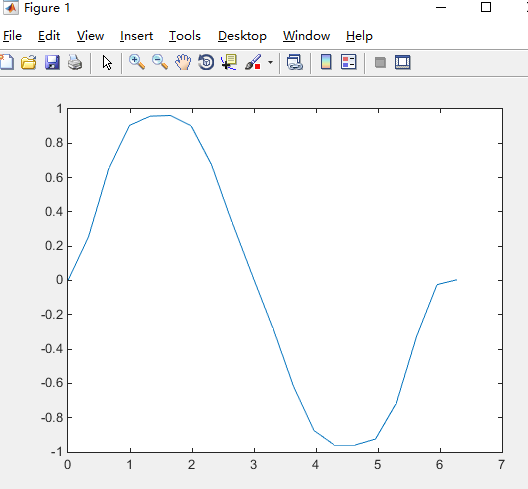
**三次样条插值**：采用样条函数对数据进行拟合，并且在任意两点之间的函数三次函数，最后返回拟合函数在xi的值。

**三次插值**: 一组算法，通过pchip函数对数据进行三次Hermite插值，这种方法可以保持数据的一致性和数据曲线的形状。

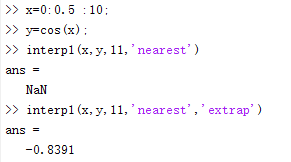
**最近邻插值算法效果最差，但是计算简单、快速**，在考虑算法运行时间并且对结果要求不是很高时，通常采用这种方法。

除此之外，还有一种方法，是基于快速傅里叶变换的方法。这种方法将输入数据视为周期函数的采用数据，对数据进行傅里叶变换，然后对更多点进行傅里叶逆变换。MATLAB中利用interpft 函数完成这一功能。y= interft（x,n），x为周期函数的均匀采样数据，n为待返回的数据个数。





外插运算： 需要通过interp1 函数添加参数“extrap”，指明所用的插值算法也用于外插运算。若没有指定外插运算，则函数返回NaN。

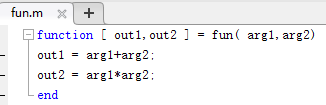


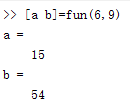
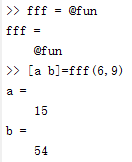
# 函数运算

## 函数的表示

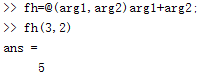
**利用M文件将函数定义为MATLAB函数；或采用匿名函数的方式。**

**利用@符号可以为函数起别名。**



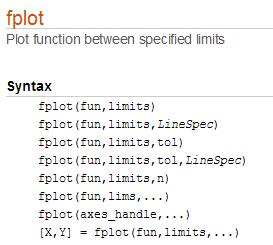
 

匿名函数：**@(arg1,...)表达式**；



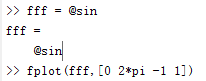
## 数学函数图像的绘制

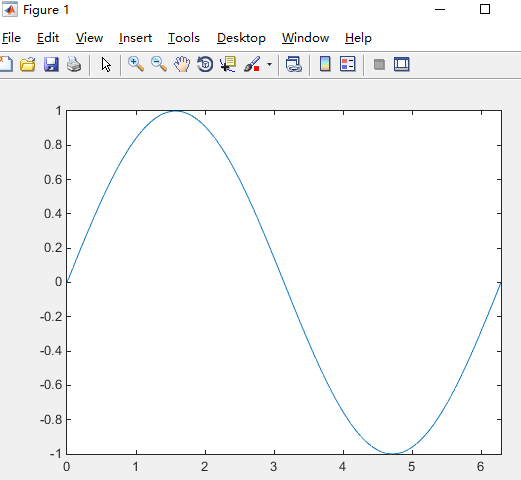
函数图像利用函数**fplot**绘制。



fun可以是M文件函数，也可以是包含变量x的字符串，还可以是函数句柄。

limits 用于限定x轴范围或同时限定x、y轴范围，即[xmin xmax]、[xmin xmax ymin ymax]。





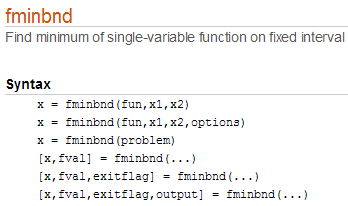


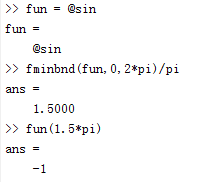
返回点的坐标，但是不会作图。

## 函数极值

### 一元函数的极小值

利用fminbnd 函数求的给定区间内的局部极小值。





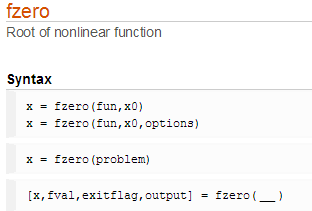
### 多元函数的极小值

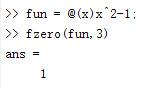
利用函数fminsearch完成。fminsearch函数的输入参数为初始值。

## 函数求解

利用fzero来求一元函数的零点。

可以指定一个初始值或者指定一个区间。





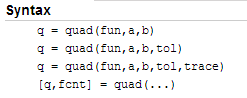
## 数值积分

利用函数quad、quadl、quadv、dblquad、triplequad函数。

### 一元函数的积分

**quad** 一元函数的数值积分，采用低阶自适应的Simpson方法；

**quadl** 一元函数的数值积分，采用高阶自适应的Lobatto方法。

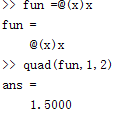


第一个为函数fun在[a,b]区间上的积分，精确度为1e-6；

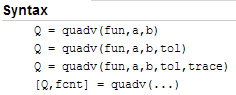
第二个指定允许误差tol，且tol必须大于1e-6；

第三个跟踪迭代过程，具体见文档介绍。

第四个，输出函数值得同时输出计算函数值的次数。



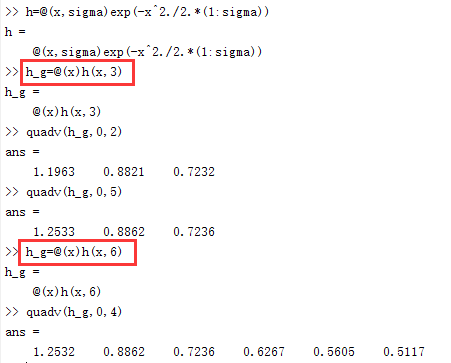
### 一元函数的矢量积分



与quad相似。

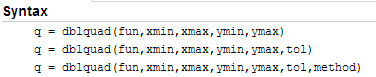
**矢量积分返回一个向量，每个元素的值为一个一元函数的积分值**。quadv与quad相似，可以设置积分参数和结果输出。

示例：



### 二重积分和三重积分

二重积分：



**函数参数为函数句柄、两个自变量的积分限。**

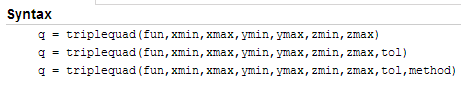
**tol为指定积分结果的精度。**

**method：指定积分方法，method可以是@quad，也可以是自定义的积分函数句柄。**

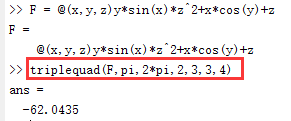
**示例：**



**三重积分：**



参数为函数句柄、三个自变量的积分限。tol为指定精度，method为积分方法。

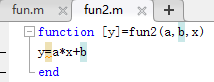
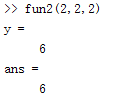


## 含参数函数的使用（不明白）

大很多情况下，函数都是带有参数的。在MATLAB中，使用含参数函数有两种方式：使用嵌套函数和使用匿名函数。

### 用嵌套函数提供函数参数

在函数M文件中，把参数当做是函数参数输入。调用时传入参数的具体数值。最后把含参函数以嵌套函数的方式包含在M文件中。

### 用匿名函数提供函数参数

首先创建含参函数，调用函数前，对参数赋值，然后利用含参函数创建匿名函数，利用匿名函数的句柄计算即可。

# 微分方程

MATLAB提供了偏微分方程工具箱求解偏微分方程问题。

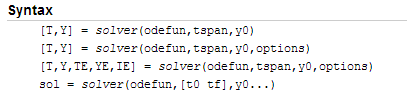
## 常微分方程初值问题

MATLAB求解的常微分方程主要是三种类型：

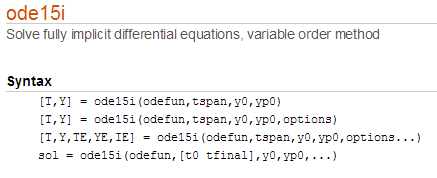
显式常微分方程：y’ = f(t,y)

线性隐式常微分方程：M(t,y)y’ = f(t,y)

全隐式常微分方程f（t,y,y’）= 0



This page contains an overview of the solver functions: **ode23, ode45, ode113, ode15s, ode23s, ode23t, and ode23tb.** You can call any of these solvers by substituting the placeholder, solver, with any of the function names.



## 常微分方程边值问题

bvp4c函数用于求解常微分方程边值问题。

