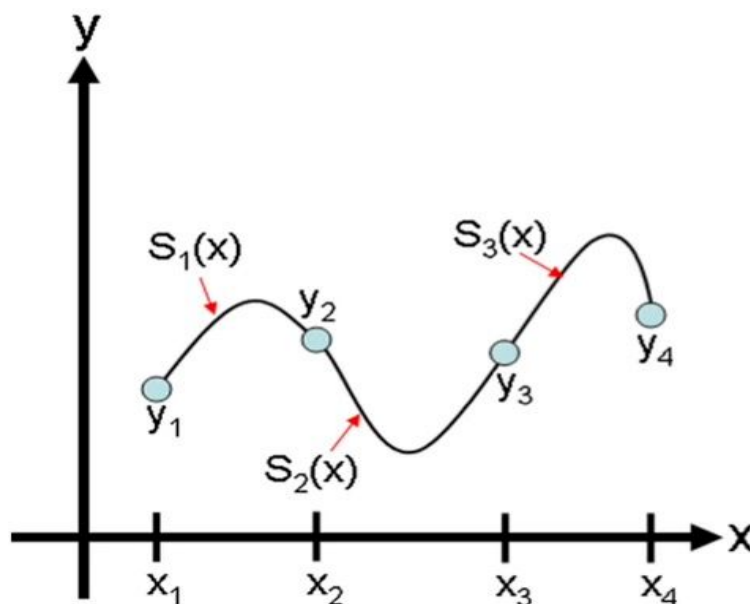


计算物理 第一部分

第4讲 内插与函数的计算、数据拟合



李强 北京大学物理学院西楼227

qliphy0@pku.edu.cn, 15210033542

内插面对的基本问题大致如下:

假定我们在一系列控制变量(自变量) x 取值处—例如我们在 $x=x_0, x_1, \dots, x_n$ 这 $(n+1)$ 个两两不同的点处—获得了相应函数 $y=f(x)$ 的数值: y_0, y_1, \dots, y_n , 并且如果我们**对函数 $y=f(x)$ 的宏观性状做某些合理的假定**(例如, 它是什么类型的函数等等), 我们**是否可以根据这些信息完全确定, 或者近似地确定这个函数本身?**如果能够“近似地”确定这个函数, 那么我们就不再需要去测量其他 x 处(即 $x \neq x_i, i=0, \dots, n$ 处)的函数值, 而可以直接“近似地”计算它。

这种方法是有意义的。

因为一方面在我们没有测量同时又希望了解的那些点处的实验可能是十分耗时, 甚至是不可能实现的;

另一方面, 我们有时候需要在一定的定义域内获得相应物理量 Y 的一个简单的表达式, 这个表达式可以用于进一步计算。

不失一般性，我们可以假设已经测量的这些自变量的点是按照由小到大排列的。它们被称为内插问题中的**支撑点(support points)或者节点(nodes)**。即: $x_0 < x_1 < \cdots < x_n$, 而整数 n 则称为支撑点的数目。我们要求内插的函数 $f(x)$ 必须满足:

$$f(x_i) = y_i, \quad i = 0, 1, \cdots, n$$

显然，**支撑点数目越大，我们对函数的了解就越精细**。如果我们希望计算其他点 $x \in [x_0, x_n]$, 这个问题称为内插问题; 而如果 $x < x_0$ 或 $x > x_n$, 这个问题称为外推问题。当然，笼统地说我们也可统称其为内插问题。

外推和内差还是不太一样。因为这时候我们感兴趣的是函数在某个已知区间之外的点的函数值。如果我们不能够对函数的形式进行任何的限制的话，外推比内差更可能得到完全疯狂的结果。

注意内插问题与拟合问题是不太一样的。**内插问题中我们假定在支撑点 x_i 处测得的函数值 y_i 是严格的**。在拟合问题中，我们一般是需要考虑 y_i 的误差的。

由于在内插问题中，我们认为在支撑点处的值是严格的。因此，**在待定函数之中的参数的数目一般也与支撑点的数目相同**。所不同的是函数的形式。**函数形式的选择则依赖于我们对问题的物理理解**。一般来说，如果我们认为待定的函数在我们研究的区间应当是无限光滑解析的，我们会选择诸如**多项式这类的函数**；反之，如果我们认为所研究的函数在该区间中可能有奇异性，我们则可以选择**具有极点的分式函数，或者称为有理分式内插**。在下面的几节中我们依次来讨论这些内插方法。最后我们会讨论使用**样条函数进行分段拟合**的情况，这种情形是函数在整个区间中并没有一个统一的形式。样条函数特别是三次样条函数在计算机图像学中有非常广泛的应用。

本章中我们还将讨论函数的计算问题。这包括两种情形。一种情形是**函数具有明确的解析表达式**。这个表达式可能是由级数或者一个积分表达式给出。我们需要的是从数值上准确地计算这个函数值。**另外一类是首先利用某种近似来描写目标函数**，然后再计算相应的函数。这一类与内插问题有类似之处。

需要强调的是，函数的计算是我们后面几章讨论函数的积分、方程求根以及函数极值问题的基础。

多项式内插:拉格朗日多项式

本节中我们讨论多项式的内插。考虑区间 $[x_0, x_n]$ 上面的 n 次多项式

$$P_n(x) = a_0 + a_1x + \cdots + a_nx^n$$

它具有 $(n+1)$ 个参数: $a_i, i=0, 1, \cdots, n$ 。我们试图利用这个多项式来解决区间 $[x_0, x_n]$ 上面的内插问题。也就是说, 我们希望有, $P_n(x_i)=y_i, i=0, \cdots, n$ 。这个问题的一般解是著名的**拉格朗日多项式**。

我们考虑如下形式的 $n+1$ 个 n 次多项式:

$$L_j(x) = \prod_{0 \leq m \leq n, m \neq j} \frac{x - x_m}{x_j - x_m}, \quad j = 0, 1, \cdots, n.$$

因为分母上 $x_j - x_m$ 是已知的数, 所以这个表达式是个典型的 n 次多项式。按照构造, 这个多项式满足

$$L_j(x_i) = \delta_{ij}, \quad 0 \leq i, j \leq n.$$

$$P_n(x) \equiv \sum_{i=0}^n y_i L_i(x),$$

满足 $P_n(x_i)=y_i, i=0, 1, \cdots, n$ 。

这个公式一般称为拉格朗日内插公式

多项式内插: 牛顿内插

另外一种多项式的内插方法是牛顿提出来的。它的特点是随着支撑点的增多, 在利用已有的多项式的情形下, 逐阶地提高多项式的阶数。

$$\begin{cases} N(x) = \sum_{i=0}^n a_i n_i(x), & n_1(x) = x - x_0 \\ n_i(x) = \prod_{k=0}^{i-1} (x - x_k), \quad n_0(x) \equiv 1 & n_2(x) = (x - x_0)(x - x_1) \\ & \dots \end{cases}$$

这个多项式的好处是, 如果我们已经从 n 个支撑点的信息中获得了 $n_i(x)$, $i = 0, 1, \dots, n-1$, 现在假设我们又增加了一个支撑点 (x_n, y_n) , 我们需要的仅仅是计算 a_n 和 $n_n(x)$ 。显然, $n_{n+1}(x) = (x - x_n)n_n(x)$, 而系数 a_n 则可以通过求解下面的线性方程组来获得:

$$\begin{bmatrix} 1 & \dots & \dots & 0 \\ 1 & x_1 - x_0 & \dots & \vdots \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 1 & (x_n - x_0) & \dots & \dots & \prod_{i=0}^{n-1} (x_n - x_i) \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ \vdots \\ y_n \end{bmatrix}$$

这是个下三角矩阵, 可用前一章讨论过的方法进行求解。

加入新的支撑点仅仅需要重新求解线性方程组最后一行即可。

多项式内插:Neville算法

假定我们的 $n+1$ 个节点为 (x_i, f_i) , $i=0, 1, \dots, n$ 。我们利用下面的递推关系定义一系列多项式:

$$P_i(x) \equiv f_i, \quad i = 0, 1, \dots, n$$

$$P_{i_0 i_1 \dots i_k}(x) = \frac{(x - x_{i_0})P_{i_1 \dots i_k}(x) - (x - x_{i_k})P_{i_0 \dots i_{k-1}}(x)}{x_{i_k} - x_{i_0}}$$

根据这个定义并且利用数学归纳法不难证明, 这个多项式满足

$$P_{i_0 i_1 \dots i_k}(x_{i_j}) = f_{i_j}, \quad j = 0, 1, \dots, k$$

因此它满足过各个节点的条件。具体构造的时候比较有效的办法是列出一个**三角形图**来

x_k	$k = 0$	1	2	3
x_0	$P_0(x) \equiv f_0$			
x_1	$P_1(x) \equiv f_1$	$P_{01}(x)$		
		$P_{12}(x)$	$P_{012}(x)$	
x_2	$P_2(x) \equiv f_2$		$P_{123}(x)$	$P_{0123}(x)$
		$P_{23}(x)$	\vdots	\vdots
x_3	$P_3(x) \equiv f_3$	\vdots		
\vdots	\vdots			

多项式内插:Neville算法

$k=0$ 的一列对应于原始节点的函数值; $k=1$ 的一列中各个函数可以通过它的左方一列中上下相邻的两个元素, 根据迭代式给出来, 例如

$$\begin{aligned}P_{01}(x) &= \frac{(x - x_0)P_1(x) - (x - x_1)P_0(x)}{x_1 - x_0} \\P_{12}(x) &= \frac{(x - x_1)P_2(x) - (x - x_2)P_1(x)}{x_2 - x_1} \\P_{23}(x) &= \frac{(x - x_2)P_3(x) - (x - x_3)P_2(x)}{x_3 - x_2} \\&\dots\end{aligned}$$

然后可以进一步获得 $k=2,3\cdots$ 的各个列。

当我们需要增加一个节点的时候, 我们只需要重新计算这个三角图的右下角的一个边上出现的数据。

例如如果我们需要加上第五个节点 x_4 , 我们就只需要在图中的标有“ \cdots ”的地方填入相应的数据即可,

三角图中心部分的数据不再需要再重新进行计算

多项式内插:Neville算法

一个常用的记号上的简化是
这样一来上面的三角图就变为

x_0	$T_{00} = f_0$			
x_1	$T_{10} = f_1$	T_{11}		
		T_{21}	T_{22}	
x_2	$T_{20} = f_2$		T_{33}	
		T_{32}	\vdots	
		T_{31}	\vdots	
x_3	$T_{30} = f_3$	\vdots		
\vdots	\vdots			

$$T_{i+k,k} = P_{i,i+1,\dots,i+k}(x)$$

多项式内插的Neville算法由以下迭代的式子给出。首先填充三角图的第一列: 令 $T_{j0}=f_j$, $j=0,1,2,\dots,n$ 。

For $k=1, 2, \dots, n$
(计算第2列到第 $n+1$ 列)

$$\begin{aligned} T_{j,k} &= \frac{(x - x_{j-k})T_{j,k-1} - (x - x_j)T_{j-1,k-1}}{x_j - x_{j-k}} \\ &= T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{\frac{x - x_{j-k}}{x - x_j} - 1} \end{aligned}$$

End

最后的 n 阶多项式由 $T_{n,n}$ 给定。对于每个给定的 x , 这个计算量大约为 $O(n^2)$

多项式内插: 误差估计

假设函数 $f(x)$ 在区间 $[x_0, x_n]$ 上至少具有 $(n+1)$ 阶的导数, 那么对于任意的 $x \in [x_0, x_n]$, 我们一定可以找到一个 $\xi \in [x_0, x_n]$ 使得,

$$f(x) - P_n(x) = \frac{\omega(x)f^{(n+1)}(\xi)}{(n+1)!}$$

其中 $\omega(x) = (x-x_0)\cdots(x-x_n)$ 。也就是说, 我们必定有

$$|f(x) - P_n(x)| \leq \frac{|x_n - x_0|^n}{(n+1)!} \max_{x_0 \leq \xi \leq x_n} |f^{(n+1)}(\xi)|$$

这个误差估计可以由中值定理得到。定义

$$f(x) - P_n(x) = G(x)\omega(x)$$

把 $x=x^*$ 当成是 $[x_0, x_n]$ 区间中的某个我们感兴趣的点。定义

$$K(x) = f(x) - P_n(x) - G(x^*)\omega(x)$$

多项式内插: 误差估计

$$K(x) = f(x) - P_n(x) - G(x^*)\omega(x)$$

很明显, $K(x)$ 有 x_0, x_1, \dots, x_n 这 $n+1$ 个零点, 且 $x=x^*$ 也是 $K(x)$ 的零点。
 $K(x_i)=0$ 和 $K(x_{i+1})=0$, 必然意味着存在 $\xi_i \in [x_i, x_{i+1}]$, 使得 $K'(\xi_i)=0$, 于是 $K'(x)$ 有 $n+1$ 个零点。以此类推, 我们最后得到, **在 $[x_0, x_n]$ 区间, 必存在 ξ , 使得 $K^{(n+1)}(\xi)=0$** 。其中

$$K^{(n+1)}(x) = f^{(n+1)}(x) - ((n+1)!)G(x^*)$$

于是, 对于任意 $G(x^*)$, $x^* \in [x_0, x_n]$, 都存在 ξ , 使得

$$f^{(n+1)}(\xi) - ((n+1)!)G(x^*) = 0$$

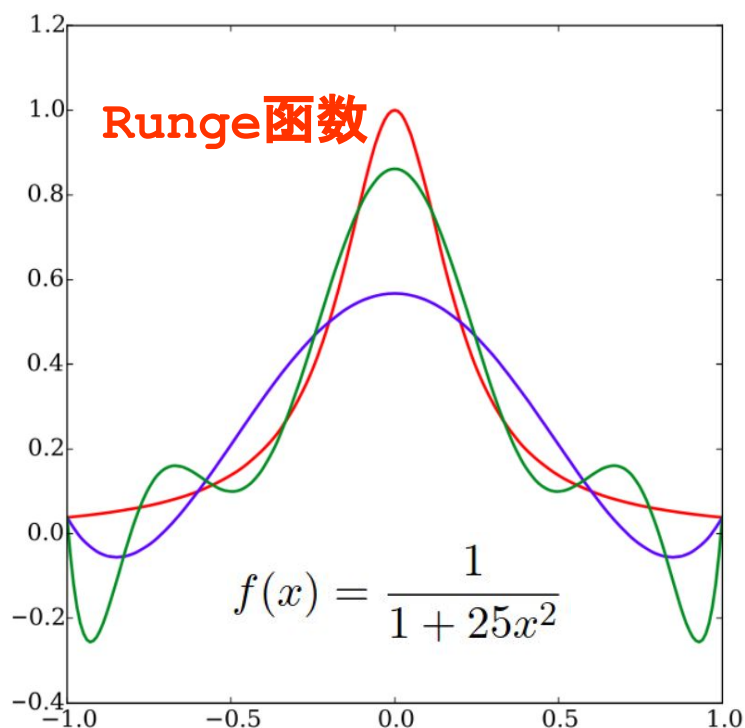
或者说, 对于任意 $x \in [x_0, x_n]$, 都有

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega(x)$$

如果导数 $|f^{(n+1)}(\xi)|$ 不是随着 n 增长得很快快的话, 那么基本上可以保证 $(n+1)! \gg |x_n - x_0|^n$ 。但很多情况下这个误差并不一定能够保证随着 n 的增加, 误差就必定减小。

Runge 现象

初看起来这个构造似乎已经完美解决了内插问题，至少是解决了利用多项式进行内插问题。但是实际上它是有一定的问题的。**这个问题是当我们测量的支撑点的数目增加时，多项式的次数也随之增加。**我们会发现，尽管在所有支撑点处拉格朗日内插公式都满足 $P_n(x_i)=y_i$ ，但是**在那些不是支撑点的地方，拉格朗日内插多项式可能与我们期望内插的函数-这个函数的形式我们并不清楚-可能相差很远。**这个现象被称为Runge现象。



$x \in [-1, +1]$ 之间的Runge函数图像(红线)。同时显示的还有 $n=5$ 阶(紫线)和 $n=9$ 阶(绿线)拉格朗日多项式拟合。前者在6个支撑点处与Runge函数吻合;后者则在10个支撑点处吻合。但只要不在这些支撑点处，随着内插阶数升高，拉格朗日多项式体现出越来越严重的震荡特性。这就是所谓的Runge现象。事实上可以证明:

$$\lim_{n \rightarrow \infty} (\max_{-1 \leq x \leq +1} |f(x) - P_n(x)|) = +\infty$$

随着内插阶数的升高，内插多项式与原函数的最大偏离的绝对值会发散。

Runge 现象

$$\lim_{n \rightarrow \infty} (\max_{-1 \leq x \leq +1} |f(x) - P_n(x)|) = +\infty$$

随着内插阶数升高，内插多项式与原函数的最大偏离的绝对值会发散。

- 在相当多的应用中，**内插的目的恰恰是希望得到一定区间内没有测量过的点处的函数值。**
- 如果我们的内插函数在支撑点之外的地方可能严重偏离实际的函数，那这种偏差对这一类的应用是不可原谅的。因此我们希望克服这种弊端。
- **具体到Runge函数本身，使用分式拟合就可以很好地解决这个问题。**或者使用所谓的三次样条函数来进行内插。
 - 由于Runge函数本身就是一个分式的形式，如果使用正确的分式内插可以完全准确地确定这个函数；
 - 而如果我们使用三次样条函数来进行内插，我们也可以获得相当不错的结果。

有理分式内插

当函数在某个区间内的变化行为比较剧烈时，多项式内插会出现剧烈震荡的行为。这时候利用有理分式进行内插可能会更为合适一些。考虑一个分式：

$$\Phi^{(m,n)}(x) = \frac{P_m(x)}{Q_n(x)}$$

其中分子和分母分别是 m 阶和 n 阶的多项式。**我们下面将假定这两个多项式是互素的，也就是说并不存在一个公共的多项式因子。**

下面我们直接构造这样的有理函数。从 $i=0,1,\dots$ 开始，我们用下面的次序来递推：

i	x_i	y_i			
0	x_0	y_0			
1	x_1	y_1	$\phi(x_0, x_1)$		
2	x_2	y_2	$\phi(x_0, x_2)$	$\phi(x_0, x_1, x_2)$	
3	x_3	y_3	$\phi(x_0, x_3)$	$\phi(x_0, x_1, x_3)$	$\phi(x_0, x_1, x_2, x_3)$
...			...		

有理分式内插

其中的各个函数 ϕ 可以按照下式进行递推地定义

$$\begin{aligned}\phi(x_i, x_j) &= \frac{x_i - x_j}{y_i - y_j} \\ \phi(x_i, x_j, x_k) &= \frac{x_j - x_k}{\phi(x_i, x_j) - \phi(x_i, x_k)} \\ &\dots \\ \phi(x_i, \dots, x_l, x_m, x_n) &= \frac{x_m - x_n}{\phi(x_i, \dots, x_l, x_m) - \phi(x_i, \dots, x_l, x_n)}\end{aligned}$$

显然，**为了获得具体的数而不是发散的结果，我们应当尽可能选择函数单调的一个区间进行内插的构造**。由于我们总是假设各个 x_i 是不相同的，因此如果函数是单调的，那么上述各个构造的差值比 ϕ 就不会发散。一旦获得了这些系数，我们可以构造一个有理分式：

$$\Phi^{(n,n)}(x) = \frac{P_n(x)}{Q_n(x)}$$

它是两个 n 次多项式的比。我们要求它能够经过 $(2n+1)$ 个点 (x_i, y_i) , $i=0, 1, \dots, 2n$ 。事实上这个结果可以写成一个连分数：

有理分式内插

$$\Phi^{(n,n)}(x) = y_0 + \frac{x - x_0}{\phi(x_0, x_1) + \frac{x - x_1}{\phi(x_0, x_1, x_2) + \frac{x - x_2}{\ddots + \frac{x - x_{2n-1}}{\phi(x_0, \dots, x_{2n})}}}}$$

可证明此分式恰好通过给定的 $2n+1$ 个点： $\Phi^{(n,n)}(x_i) = y_i, \quad i = 0, 1, \dots, 2n。$

作为一个例子，我们考虑前面曾经提及的Runge函数

$$f(x) = 1/(1+25x^2)$$

我们选择其单调的区间，比如 $[0, 1]$ 来进行内插。假定知道下列三个点的数

值： $x^2=0, 1/25, 1$ ，相应的函数值分别为： $1, 1/2, 1/26$ 。按照上面的构造，按照 x^2 的有理分式

$$f(x) = \Phi^{(1,1)}(x^2) = 1 + \frac{x^2}{\phi(0, 1/25) + \frac{x^2 - 1/25}{\phi(0, 1/25, 1)}}$$

$$\phi(0, 1/25) = \frac{0 - 1/25}{1 - 1/2} = -\frac{2}{25}$$

$$\phi(0, 1) = \frac{0 - 1}{1 - 1/26} = -\frac{26}{25}$$

$$\phi(0, 1/25, 1) = \frac{1/25 - 1}{-2/25 + 26/25} = -1$$

将这些数值代入，我们竟然可以完全重构 $f(x)$

样条函数(spline function)内插

样条这个词最早源于早期工程师绘图所用的薄木条，将它固定在一些给定的数据点上，就可以绘出一条连接各点的光滑曲线。

我们可以想象一下，要连接2个点，用一个一次多项式就可以，但是我们无法保证在端点处曲线是光滑的，也就是一阶导数连续。**如果我们用一个二次多项式去连接 2 个点，我们就能保证曲线在端点处是光滑的。**这就是所谓的二次样条插值。**进而可以有三次样条插值。**

考虑一个区间 $[a, b]$ 的一个 n 段分割： $a = x_0 < x_1 \cdots x_{n-1} < x_n = b$ 。位于区间中间的那些点 x_i , $i = 1, \cdots, n-1$ 称为节点(knots)。

我们先考虑二次样条函数 $S(x)$ ，满足

- $S(x)$ 在每个区间 $[x_{i-1}, x_i]$ 上是一个二次多项式；
- $S(x)$ 在所有节点处满足 x_i ($i = 1, 2, \cdots, n-1$) 上具有一阶连续导数；
- $S(x)$ 在所有节点处满足 $S(x_i) = y_i$ ($i = 0, 1, \cdots, n$)。

二次样条函数内插

二次样条函数 $S(x)$ 在每个小区间 $[x_{i-1}, x_i]$ 上是一个二次多项式, 有 3 个系数, 因此要确定 $S(x)$ 就要确定**3n个待定参数**, 而由 $S(x_{i-0}) = S(x_{i+0}) = y_i$ ($i=1, \dots, n-1$), 得**2(n-1)个方程**; 在两端点处, $S(x_i) = y_i$, 有**2个方程**。然后根据导数连续 $S'(x_{i-0}) = S'(x_{i+0})$ ($i=1, \dots, n-1$) 再得**(n-1)个方程**。

还需增加1个条件, 这个条件通常是在区间 $[a, b]$ 的两端处给出, 即边界条件, 常见的边界条件类型有:

- 给定初始端点或者终端的一阶导数值: $S'(x_0) = y'_0$ 或者 $S'(x_n) = y'_n$
- 给定初始端点或者终端的二阶导数值: $S''(x_0) = y''_0$ 或者 $S''(x_n) = y''_n$
- 若插值函数为周期函数时, 此时 $y_0 = y_n$, 可以给定周期性边界条件: $S'(x_0) = S'(x_n)$
- 非结点条件: 假定 $[x_0, x_2]$ 区间用一个统一的二次多项式描述, 或者 $[x_{n-2}, x_n]$ 之间用一个统一的二次多项式描述

对最后一个条件, 我们原本在 $[x_0, x_1]$ 和 $[x_1, x_2]$ 区间各有一个二次多项式, 总共6个参数, 方程是 $S(x_0) = y_0$, $S(x_{1-0}) = S(x_{1+0}) = y_1$, $S(x_{2-0}) = y_2$, $S'(x_{1-0}) = S'(x_{1+0})$, 总共5个方程。现在 $[x_0, x_2]$ 区间用一个统一的二次多项式描述, 那么我们一共有3个参数。方程也刚好是3个: $S(x_0) = y_0$, $S(x_1) = y_1$, $S(x_{2-0}) = y_2$ 。原来多出一个参数的情况, 现在变成参数数目和方程数目正好相等。

二次样条函数内插

假设 $S'(x_j) = M_j$ ($j = 0, \dots, n$), 在各个小区间内, $S'(x)$ 为一次多项式, 我们可以写成

$$S'(x) = M_j \left(\frac{x_{j+1} - x}{x_{j+1} - x_j} \right) + M_{j+1} \left(\frac{x - x_j}{x_{j+1} - x_j} \right), \quad x \in [x_j, x_{j+1}]$$

我们可以定义 $h_j = x_{j+1} - x_j$ 。做一次积分以后, 得到

$$S(x) = -\frac{M_j}{2h_j}(x - x_{j+1})^2 + \frac{M_{j+1}}{2h_j}(x - x_j)^2 + a_j$$

根据 $S(x_j) = y_j$, $S(x_{j+1}) = y_{j+1}$, 我们可以得到

$$-M_j h_j / 2 + a_j = y_j, \quad M_{j+1} h_j / 2 + a_j = y_{j+1}$$

下一步, 就是得到递推关系式

$$M_{j+1} = \frac{2(y_{j+1} - y_j)}{h_j} - M_j$$

再加上边界条件。我们就能确定所有 M_j 的值以及 a_j 的值, 于是 $S(x)$ 确定。

三次样条函数内插

有二次样条函数, 就有k次样条函数。当多项式的阶数比较小的时候, 插值函数不够光滑; 当多项式阶数很高时, 比方说n阶多项式, 事实上样条函数等同于多项式插值, 这个时候就没法避免Runge现象的发生。所以简单的分析告诉我们, 阶数并不是越大越好, 也不是越小越好。事实上, 当我们考虑某个区间[a,b]上的 $m > 0$ 阶以下的所有阶导数连续且其m阶导数平方可积的函数构成的函数空间, 将其记为 $K^m[a,b]$ 。对于任意的函数 $f \in K^2[a,b]$, 我们可以定义一个函数的模

$$\|f\| = \int_a^b dx |f''(x)|^2$$

我们知道一个函数f(x)的曲率

如果在一个区间上 $f'(x)$ 平方比起1来说要小很多话, 那么函数的曲率几乎就等于 $f''(x)$ 。

因此, 上面这个模从某种意义上是衡量了函数在一个区间上曲率模方的大小。

可以证明, 三次样条函数实际上是使得这个模最小的函数。换句话说, 它是“最光滑的”函数。

$$R(x) = f''(x)(1 + f'(x)^2)^{-3/2}.$$

$$\begin{aligned} \|S(t)\| &= \sqrt{x(t)^2 + y(t)^2} & T(t) &= \frac{S'(t)}{\|S'(t)\|} = \left[\frac{\frac{x'(t)}{\sqrt{x'(t)^2 + y'(t)^2}}}{\frac{y'(t)}{\sqrt{x'(t)^2 + y'(t)^2}}} \right] \\ \|S'(t)\| &= \sqrt{x'(t)^2 + y'(t)^2} \end{aligned}$$

$$\kappa = \frac{\|T'(t)\|}{\|S'(t)\|} = \frac{|x'y'' - x''y'|}{(x'^2 + y'^2)^{3/2}}$$

三次样条函数内插

我们不加证明地引用下列所谓**最小模定理**。

定理 3.1 实区间 $[a, b]$ 的任一 n 段分割 $\Delta: a = x_0 < x_1 \cdots x_{n-1} < x_n = b$ 上的三次样条函数 $S_\Delta(Y; x_i) = y_i, i = 0, 1, \cdots, n$ 。对任给的 $f \in \mathcal{K}^2[a, b]$ 且 $f(x_i) = y_i$ ，我们一定有 $\|f\| \geq \|S_\Delta(Y; \cdot)\|$ 。其中的函数模由式 (3.27) 所定义。事实上可以证明，

$$\|f - S_\Delta(Y; \cdot)\|^2 = \|f\|^2 - \|S_\Delta(Y; \cdot)\|^2 \geq 0,$$

只要函数 f 满足下列条件之一即可：

1. $S''_\Delta(Y; a) = S''_\Delta(Y; b) = 0$;
2. $f \in \mathcal{K}_p^2[a, b]$, $S_\Delta(Y; \cdot)$ 是周期的;
3. $f'(a) = S'_\Delta(Y; a), f'(b) = S'_\Delta(Y; b)$.

并且上面的任何一个条件都唯一地确定了样条函数 $S_\Delta(Y; \cdot)$ 。

三次样条函数内插

- $S(x)$ 在每个区间 $[x_{i-1}, x_i]$ 上是一个**三次多项式**;
- $S(x)$ 在所有节点处满足 $x_i (i=1, 2, \dots, n-1)$ 上具有一阶连续导数和二阶连续导数;
- $S(x)$ 在所有节点处满足 $S(x_i)=y_i (i=0, 1, \dots, n)$ 。

对于三次样条插值函数, 要确定 $S(x)$ 就要确定 **$4n$ 个待定参数**。由 $S(x_{i-0})=S(x_{i+0})=y_i (i=1, \dots, n-1)$, 得 $2(n-1)$ 个方程; 在两个端点处, $S(x_i)=y_i$, 有2个方程。然后根据一阶、二阶导数连续 $S'(x_{i-0})=S'(x_{i+0})$, $S''(x_{i-0})=S''(x_{i+0}) (i=1, \dots, n-1)$ 再得 $2(n-1)$ 个方程。**一共是 $4n-2$ 个方程**。所以**需要增加2个边界条件**来确定样条函数。常见的边界条件类型有:

- 同时给定初始端点和终端的一阶导数值: $S'(x_0)=y'_0, S'(x_n)=y'_n$
- 同时给定初始端点和终端的二阶导数值: $S''(x_{0,n})=y''_{0,n}$
- 若插值函数为周期函数时, 此时 $y_0=y_n$, 可给定两个周期性边界条件: $S'(x_0)=S'(x_n), S''(x_0)=S''(x_n)$
- 非结点条件: 假定 $[x_0, x_2]$ 区间用一个统一的三次多项式描述, 并且 $[x_{n-2}, x_n]$ 之间也可以用一个统一的三次多项式描述

三次样条函数内插

关于三次样条插值函数的构造, 我们可以采取类似的方式。假设三次样条函数的矩(moments)为 $S''(x_j)=M_j$ ($j=0, \dots, n$), 在各个小区间内, $S''(x)$ 为一次多项式, 我们可以写成:

$$S''(x) = M_j \left(\frac{x_{j+1} - x}{x_{j+1} - x_j} \right) + M_{j+1} \left(\frac{x - x_j}{x_{j+1} - x_j} \right), \quad x \in [x_j, x_{j+1}]$$

对这个式子做两次积分

$$S(x) = -\frac{M_j}{6h_j}(x - x_{j+1})^3 + \frac{M_{j+1}}{6h_j}(x - x_j)^3 + A_j(x - x_j) + B_j, \quad x \in [x_j, x_{j+1}]$$

根据 $S(x_j)=y_j$, $S(x_{j+1})=y_{j+1}$, 我们可以得到

$$A_j = \frac{y_{j+1} - y_j}{h_j} - \frac{h_j}{6}(M_{j+1} - M_j), \quad B_j = y_j - M_j \frac{h_j^2}{6}$$

我们还剩一个条件没有用, 就是 $S'(x)$ 连续。 $S'(x)$ 的表达式是

$$S'(x) = -\frac{M_j}{2h_j}(x - x_{j+1})^2 + \frac{M_{j+1}}{2h_j}(x - x_j)^2 + A_j, \quad x \in [x_j, x_{j+1}]$$

三次样条函数内插

先考虑节点 x_j 处左导数, 通过 $[x_{j-1}, x_j]$ 区间上的函数表达式得到:

$$S'(x_j - 0) = \frac{M_j}{2}h_{j-1} + A_{j-1} = \frac{h_{j-1}}{6}M_{j-1} + \frac{h_{j-1}}{3}M_j + \frac{y_j - y_{j-1}}{h_{j-1}}$$

同理, 利用 $[x_j, x_{j+1}]$ 区间的函数表达式计算节点 x_j 的右导数

$$S'(x_j + 0) = -\frac{M_j}{2}h_j + A_j = -\frac{h_j}{6}M_{j+1} - \frac{h_j}{3}M_j + \frac{y_{j+1} - y_j}{h_j}$$

要求一阶导数连续, 我们得到

$$\mu_j M_{j-1} + 2M_j + \lambda_j M_{j+1} = d_j, \quad j = 1, \dots, n-1$$

$$\mu_j = \frac{h_{j-1}}{h_{j-1} + h_j}, \quad \lambda_j = \frac{h_j}{h_{j-1} + h_j},$$

$$d_j =$$

$$6 \left[\frac{y_{j-1}}{h_{j-1}(h_{j-1} + h_j)} + \frac{y_{j+1}}{h_j(h_{j-1} + h_j)} - \frac{y_j}{h_{j-1}h_j} \right]$$

这是典型的**三对角矩阵**所对应的线性方程组的形式。

包含 $(n-1)$ 个方程。但是未知数 M_j 共有 $(n+1)$ 个。我们仍需两个边界条件才能完全确定所有的矩 M_j 进而确定三次样条函数。

函数的近似与计算:级数表达的函数的计算

很多函数是以级数的形式表达的(比如Bessel函数、Legendre函数等等)。很多函数实际上也是这样去计算的。需要指出的是, **尽管某些函数的级数展开在数学上是收敛的, 但是直接按照级数去计算往往并不是最合适的方法**。典型的例子如

$$\sin x = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

我们知道这个级数对于任意的 x 都是收敛的, 但是如果 $|x| \geq 1$, 那么直接这么计算需要求和的项数是比较多的。事实上, 一种更为有效的做法是首先将 x 化到第一或第四象限: $x \in [-\pi/2, \pi/2]$, 然后我们可以**利用公式: $\sin x = 3\sin(x/3) - 4\sin^3(x/3)$ 将其化为计算较小的角度 $(x/3)$ 的正弦问题**。当然, 这个过程可以进一步迭代直到 $x/3$ 足够地小以至于一两项级数就足以满足精度要求为止。当然**更为现代一些的算法是利用我们后面提及的Chebyshev近似来进行计算**。事实上, 多数的超越函数都可以利用Chebyshev近似的方法进行计算。(超越函数(Transcendental Functions), 指的是变量之间的关系不能用有限次加、减、乘、除、乘方、开方运算表示的函数。)

函数的Chebyshev近似及其计算

首先我们来讨论一下**连续函数的最佳平方逼近**。

假设对函数 $f(x)$, $x \in [a, b]$ 进行函数逼近,

$$S(x) = \sum_{n=0}^{N-1} c_n T_n(x)$$

其中 $c_n(n=0, 1, \dots, N-1)$ 是待定参数。我们用来逼近的函数 $T_n(x)$ 应该是形式简单的函数(比方说 $T_n(x)=x^n$, 就是多项式逼近;当然这里的简单, 并不是指的狭义上的数学形式简单, 更多是指可以通过递推关系的构造, 在计算机算法上实现起来简单)。

连续函数的最佳平方逼近问题就是求 $S(x)$, 使得 $\|S(x)-f(x)\|_2$ 达到最小值。也就是说, 我们要将下式最小化:

$$F = \|S(x) - f(x)\|_2^2 = \int_a^b dx \left[\sum_{n=0}^{N-1} c_n T_n(x) - f(x) \right]^2$$

根据 $p=2$ -范数和内积的关系, 我们可以把 F 写成

$$F = \|S - f\|_2^2 = \left\langle \sum_{n=0}^{N-1} c_n T_n(x) - f, \sum_{n=0}^{N-1} c_n T_n(x) - f \right\rangle$$

由多元函数取极值的条件知道, 系数 c_n 应满足方程

$$\frac{\partial F}{\partial c_n} = 0, \quad n = 0, \dots, N-1$$

函数的Chebyshev近似及其计算

$$\sum_{n=0}^{N-1} c_n \langle T_n, T_m \rangle = \langle f, T_m \rangle, \quad m = 0, 1, \dots, N-1$$

我们只需解这个方程组即可。特别地，**如果我们构造的函数有正交关系** $\langle T_n, T_m \rangle = A_n \delta_{mn}$ ，那么这个方程组的解是很简单的

$$c_n = \langle f, T_n \rangle / A_n$$

有时候我们会引入**加权正交**的概念，设函数 $f(x), g(x)$ 是变量取值范围为 $[a, b]$ 的函数， $\rho(x)$ 为权函数，如果

$$\langle f(x), g(x) \rangle = \int_a^b \rho(x) f(x) g(x) dx = 0$$

则称 $f(x)$ 与 $g(x)$ 带权正交。若函数族 $\{T_0(x), \dots, T_n(x), \dots\}$ 满足

$$\langle T_n(x), T_m(x) \rangle = \int_a^b dx \rho(x) T_n(x) T_m(x) = A_n \delta_{mn}$$

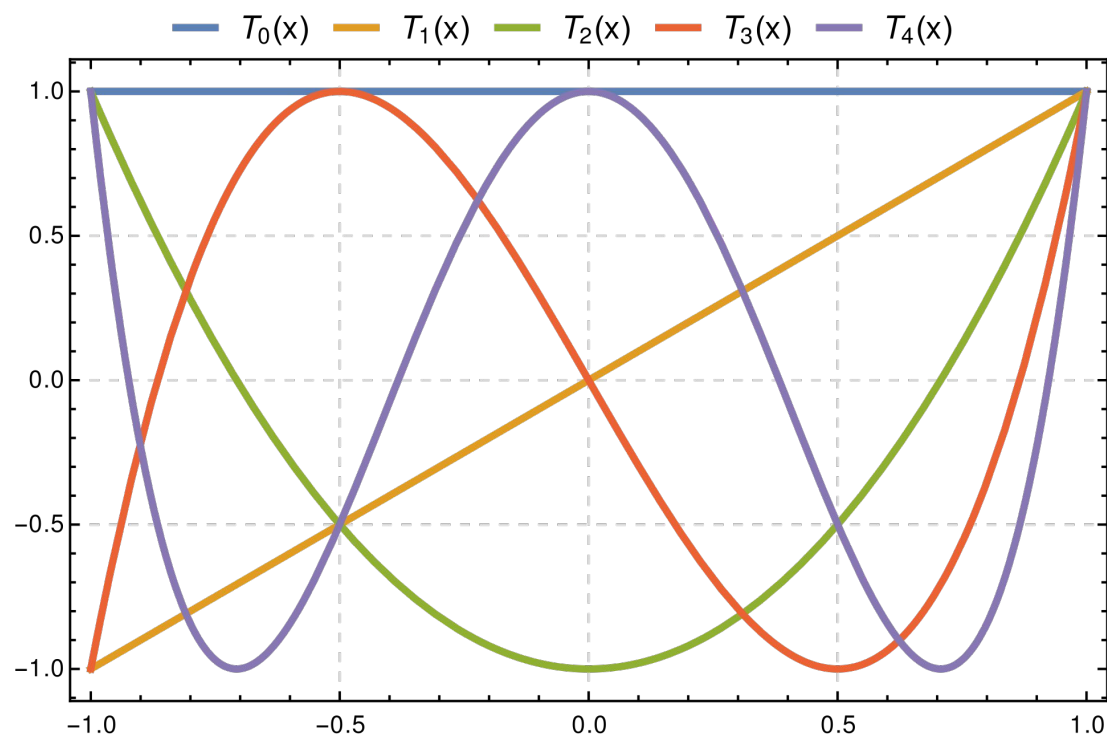
则称 $\{T_n(x)\}$ 是 $[a, b]$ 区间上的带权正交函数族。这里说明一下，就好像矩阵的范数的定义不是唯一的一样，**连续函数的最佳平方逼近也不是唯一的，我们可以在里面加入一个权重因子，一般来讲，只要这个因子满足正定和非奇异性就好。**

函数的Chebyshev近似及其计算

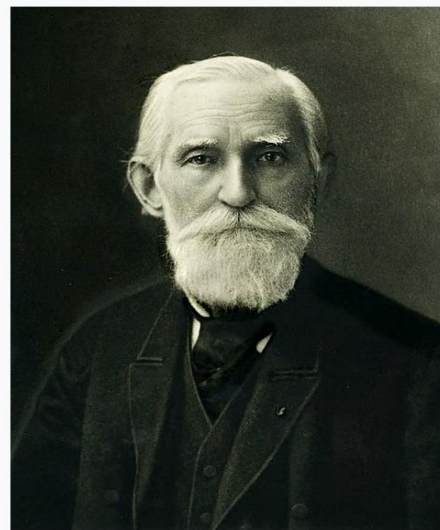
对于变量取值范围为 $x \in [a, b]$ 的函数, 我们不妨做变量替换 $x \rightarrow x' = 2(x-a)/(b-a) - 1$, 使得 $x' \in [-1, 1]$ 。下面我们对于任意的 $x \in [-1, 1]$, 引入n阶的第一类 Chebyshev多项式 $T_n(x)$:

$$T_n(x) = \cos(n \arccos x). \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_2(x) = 2x^2 - 1, \quad T_3(x) = 4x^3 - 3x$$



Pafnuty Chebyshev



Pafnuty Lvovich Chebyshev

Born 16 May 1821^[1]
Akatovo, Kaluga
Governorate, Russian
Empire^[1]

Died 8 December 1894

函数的Chebyshev近似及其计算

Chebyshev多项式满足一系列重要的性质。其中一个比较重要的性质是它满足**加权正交关系**

$$\int_{-1}^1 dx \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}} = \begin{cases} 0 & i \neq j \\ \pi/2 & i = j \neq 0 \\ \pi & i = j = 0 \end{cases}$$

另一个比较重要的性质是其零点以及极值点的位置。 **$T_n(x)$ 在 $[-1,1]$ 之中恰好有 n 个零点**，它们的位置由下式给出：

$$x_{n,k} = \cos \left(\frac{\pi(k + 1/2)}{n} \right), \quad k = 0, 1, \dots, n-1$$

而它的极值点共有 $(n+1)$ 个：

$$\hat{x}_{n,k} = \cos \left(\frac{\pi k}{n} \right), \quad k = 0, 1, \dots, n$$

在这些极值点处，Chebyshev多项式恰等于+1(极大值)或-1(极小值)。也就是说，Chebyshev多项式的绝对值总是在1以下。正是这个特性使得它成为近似任意函数的有力工具，因为可以很好地控制误差。

函数的Chebyshev近似及其计算

最后一个重要的性质是**分立版本的正交关系**。令 $x_{N,k}$ 是 $T_N(x)$ 的 N 个零点，那么对于 $i, j < N$ ，我们有

$$\sum_{k=0}^{N-1} T_i(x_{N,k}) T_j(x_{N,k}) = \begin{cases} 0 & i \neq j \\ N/2 & i = j \neq 0 \\ N & i = j = 0 \end{cases}$$

下面我们要**用Chebyshev多项式来对函数 $f(x)$ 进行近似展开**。我们首先来构造Chebyshev多项式，它的系数为 c_m , $m=0, 1, \dots, N-1$

$$c_m = \frac{1}{A_m} \int_{-1}^1 dx \frac{f(x) T_m(x)}{\sqrt{1-x^2}}, \quad \int_{-1}^1 dx \frac{T_n(x) T_m(x)}{\sqrt{1-x^2}} = A_m \delta_{mn}$$

$$c_m \approx c_{N,m} = \frac{2 - \delta_{0m}}{N} \sum_{k=0}^{N-1} T_m(x_{N,k}) f(x_{N,k}), \quad \text{例如: } f=1 \text{ 时} \dots$$

函数的Chebyshev近似及其计算

$$x_{N,k} = \cos \left(\frac{\pi(k+1/2)}{N} \right)$$

是 T_N 的零点。根据Chebyshev多项式的定义, 我们有

$$c_m \approx c_{N,m} = \frac{2 - \delta_{0m}}{N} \sum_{k=0}^{N-1} T_m(x_{N,k}) f(x_{N,k}),$$

$$T_m(x_{N,k}) = \cos(m \arccos x_{N,k}) = \cos \left(m \frac{\pi(k+1/2)}{N} \right)$$

$$c_{N,m} = \frac{2 - \delta_{0m}}{N} \sum_{k=0}^{N-1} \cos \left(\frac{m\pi(k+1/2)}{N} \right) f \left[\cos \left(\frac{\pi(k+1/2)}{N} \right) \right],$$

得到 $c_{N,m}$ 后, 可以构造

$$S(x) = \sum_{m=0}^{N-1} c_{N,m} T_m(x)$$

这个表达式也给出了函数 $f(x)$ 的一个非常好的近似。它和最佳平方逼近的差别在于 $c_{N,m} \approx c_m$ 。但它有一个很好的特性在于, 在零点 $x_{N,k}$ 处, 函数值严格等于 $f(x_{N,k})$ 。

函数的Chebyshev近似及其计算

$$S(x) = \sum_{m=0}^{N-1} c_{N,m} T_m(x)$$

这个表达式也给出了函数 $f(x)$ 的一个非常好的近似。它和最佳平方逼近的差别在于 $c_{N,m} \approx c_m$ 。但它有一个很好的特性在于，在零点 $x_{N,k}$ 处，函数值严格等于 $f(x_{N,k})$ 。

$$S(x_{N,k}) = \sum_{m=0}^{N-1} c_{N,m} T_m(x_{N,k}) = \sum_{m=0}^{N-1} \frac{2 - \delta_{0m}}{N} \sum_{k'=0}^{N-1} T_m(x_{N,k'}) f(x_{N,k'}) T_m(x_{N,k})$$

可以证明

$$\frac{T_0(x_{N,k'}) T_0(x_{N,k})}{2} + \sum_{m=1}^{N-1} T_m(x_{N,k'}) T_m(x_{N,k}) = \frac{N}{2} \delta_{kk'}$$

马上可以得到

$$S(x_{N,k}) = f(x_{N,k})$$

函数的Chebyshev近似及其计算

由于Chebyshev具有模永远小于1的特性, 因此它可以很好地控制误差。它非常接近于相应的最佳平方逼近, 而且其计算也方便很多。

由于截断到N阶的近似式的下一阶是 $c_N T_N(x)$ 而 $|T_N(x)| \leq 1$, 因此**误差主要由 $|c_N|$ 的大小来控制**。

一般来说随N的增加 c_N 衰减得非常快(如果函数足够光滑的话, 它们随着N的增加是指数衰减的), 所以我们往往仅仅需要几阶就够了。这正是Chebyshev强大的地方。**我们运用Chebyshev最多的时候并不是利用很大的N的时候, 而是往往利用不那么大的时候。**

最后我们指出, 虽然获得了Chebyshev近似之后我们可以把函数表达成标准的多项式的形式—也就是说的形式

$$f(x) = \sum_{i=0}^N a_i x^i$$

—但是我们一般不建议这样做, 除非你知道如

何处理舍入误差。换句话说, 从数值计算的角度而言, 以Chebyshev多项式为基远比以通常的x的幂次为基要好。例如我们刚刚提到的, **如果函数足够光滑, 它的Chebyshev展开系数一般是指指数衰减的。但是如果改用x的幂次来展开一般不是。**

Clenshaw算法

函数的Chebyshev展开如何计算呢？我们可利用所谓的Clenshaw提出的一个方法。这个方法专门用于计算

$$f(x) \approx S_N(x) = \sum_{k=0}^N c_k F_k(x) ,$$

类型的展开，只要基函数 $F_k(x)$ 同时满足一定的递推关系。Chebyshev 多项式显然属于这一类

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

Clenshaw 算法假设展开的基函数满足一定的递推关系

$$F_{k+1}(x) = \alpha_k(x)F_k(x) + \beta_k(x)F_{k-1}(x) .$$

这涵盖了一大类的展开：Legendre展开，Chebyshev展开等等。假定我们已经知道了系数 c_k , $k=0, \dots, N$ ，我们的目的是计算函数的 N 阶近似式 $S_N(x)$ 。**我们重新构造一系列新的函数 $b_k(x)$ ：**

$$b_{N+1}(x) = b_{N+2}(x) = 0 . \quad b_k(x) = c_k + \alpha_k(x)b_{k+1}(x) + \beta_{k+1}(x)b_{k+2}(x) .$$

Clenshaw算法

持续迭代直到我们获得 $b_1(x)$ 和 $b_2(x)$ 。最后我们发现,

$$S_N(x) = c_0 F_0(x) + b_1(x) F_1(x) + \beta_1(x) F_0(x) b_2(x) .$$

多数情形下这个迭代通常情况下是稳定的。

对于Chebyshev的特殊情形我们有 $\alpha_k(x)=2x$, $\beta_k(x)=-1$ 。因此Clenshaw迭代的公式为,

$$b_k(x) = c_k + 2xb_{k+1}(x) - b_{k+2}(x) ,$$

最终

$$S_N(x) = \frac{1}{2}c_0 + xb_1(x) - b_2(x) .$$

Chebyshev 过滤器

$$T_n(x) = \cos(n \arccos x)$$

定义域为 $[-1,1]$ 。但如果我们采用递推关系的定义方式

$$T_0(x) = 1, \quad T_1(x) = x,$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

那么, 在任何实数域都可以得到Chebyshev多项式, 或者说我们可以把Chebyshev多项式解析延拓到整个实空间。**只是在 $[-1,1]$ 区间, $T_n(x)$ 值的绝对值小于1, 而当 $|x|>1$ 时, 随着 n 变大, $T_n(x)$ 呈 $2^n x^n$ 阶的上升趋势。**利用这个性质, 我们可以构造Chebyshev过滤器(Chebyshev filter)。我们首先定义一个函数

$$q(x; \alpha, \beta) = \frac{2x^2 - (\alpha^2 + \beta^2)}{\alpha^2 - \beta^2} \quad \alpha > \beta$$

这个函数在 $x=\beta$ 和 $x=\alpha$ 之间从-1变到+1。然后我们把 $q(x;\alpha,\beta)$ 作为变量放到Chebyshev多项式中, 比方说

$$T_n(q(x; \alpha, \beta)), \quad n = 10$$

因为这个多项式在 $x>\alpha$ 和 $x<\beta$ 的值要远大于 $x \in [\beta, \alpha]$ 区间的值, 等效于这个多项式把 $[\beta, \alpha]$ 的信息过滤掉了。

Chebyshev 过滤器

Chebyshev过滤器的一个应用是求一个庞大矩阵的若干个小本征值和本征矢量。我们之前曾经讨论过,如果能得到一个矩阵A的小于 λ_0 的本征值和本征矢量,那么利用它们我们可以把线性方程组求解的条件数从 $\text{cond}=\lambda_{\max}/\lambda_{\min}$ 减小到 λ_{\max}/λ_0 。**但要求矩阵A的小本征值和本征矢量并不是那么容易,反而最大本征值和本征矢量是比较好求的。**比方说我们可以构造矩阵多项式 $p(A)$,这个多项式作用在任意矢量 $v=\sum_i a_i v_i$ 上面,得到

$$p(A)v = \sum_i a_i p(\lambda_i) v_i$$

当多项式阶数比较高时,与大本征值对应的矢量被放大,与小本征值对应的矢量被压低。所以我们相对容易得到大本征值和与之相关的本征矢量。要求小本征值系统,我们可采用Chebyshev过滤器,对于

$$T_n(q(A; \alpha, \beta))$$

所有在 $[\beta, \alpha]$ 区间的本征矢量都会被迅速压低,我们只需要设定 $\alpha > \lambda_{\max}$, $\beta \approx \lambda_0$, 我们就能得到小于 λ_0 的本征矢量构成的子空间的信息。

函数的 Pade 近似及其计算

一个函数 $f(x)$ 如果我们知道了它的Taylor展开的部分系数, 我们还可利用一个分式来近似地表示这个函数。如果我们令

$$R(x) = \frac{\sum_{k=0}^M a_k x^k}{1 + \sum_{k=1}^N b_k x^k}$$

我们期待它可以近似描述 $f(x)$ 的Taylor展开表达式

$$f(x) = \sum_{k=0}^{\infty} c_k x^k$$

我们要求

$$R^{(k)}(x)|_{x=0} = f^{(k)}(x)|_{x=0}, \quad k = 0, 1, \dots, M + N$$

并称 $R(x)$ 是函数 $f(x)$ 的一个 (M, N) 阶的Pade近似式。

假定所有的Taylor展开系数 c_k 已知, 那么最为简单的计算Pade近似式的各个系数 a_k 和 b_k 的方法是令 $f(x)=R(x)$ 并将分母乘到等式的左边, 然后比较两边 x 同幂次的系数。在假定 $M=N$ 的情形下, 这给出如下的方程

函数的 Pade 近似及其计算

在假定 $M=N$ 的情形下, 这给出如下的方程

$$\sum_{m=1}^N b_m c_{N-m+k} = -c_{N+k}, \quad k = 1, \dots, N$$

x^{N+k} term

$$\sum_{m=0}^k b_m c_{k-m} = a_k, \quad k = 1, \dots, N$$

令 $b_0=1$

上式中的前一个方程可以视为 (b_1, \dots, b_N) 的一个线性方程, 求解这个方程我们就可以获得系数 (b_1, \dots, b_N) , 再带入第二个方程就可以获得系数 (a_1, \dots, a_N) 。要能够确定解出所有的系数, **我们需要首先知道 $c_k, k=0, 1, \dots, 2N$ 这 $(2N+1)$ 个展开系数。**

Henri Padé



Born

17 December 1863
Abbeville

Died

9 July 1953 (aged 89)
Aix-en-Provence

函数的 Pade 近似及其计算

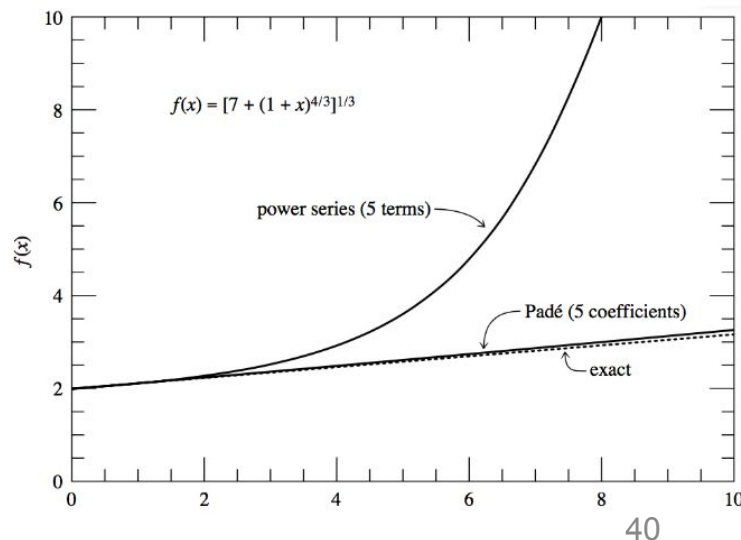
我们下面来看Pade近似是如何工作的。假定你获得了一个重要的物理量 $f(x)$ 作为一个小的耦合参数 x 的微扰展开式：

$$f(x) \approx 2 + \frac{1}{9}x + \frac{1}{81}x^2 - \frac{49}{8748}x^3 + \frac{175}{78732}x^4 + \dots$$

它实际上是你感兴趣的物理量 $f(x)$ 在 $x=0$ 处的Taylor展开的前5项。在 x 比较小时—比如说 $x < 1$ 的时候—这个表达式被大量的实验证明是相当精确的。**但是假定我们希望研究 $x > 1$ 时的 $f(x)$ 的情况。**当然，我们可以试图去计算下一阶的贡献，但这也许很困难。事实上，上面的那个展开式恰好是下列函数的Taylor展开：

$$f(x) = [7 + (1 + x)^{4/3}]^{1/3}$$

假如 x 的绝对值不是很大时， $|x| < 1$ ，上面展开式与严格函数本身差别很小。事实上，如果你把这两个函数都画出来的话，你会发现大约从 $x > 2$ 起两者差别才明显。但是如果并不知道函数的严格表达式，同时又没法去计算下一阶贡献，这时不妨试试Pade近似。**仅仅使用已知的5个系数来构建相应的Pade近似式并与函数进行比较后我们会发现，即使到大约 $x \approx 10$ ，Pade近似式与严格函数的差别都是可忽略的。**



数值微分的计算

数值上计算函数的导数实际上主要从导数定义出发 $f'(x) \approx \frac{f(x+h) - f(x)}{h}$

这个我们之前课上讲过, h 不能太大, 也不能太小; 太大了, 截断误差会大, 太小了, 舍入误差会大。有一种更好的方法是利用函数本身的 Chebyshev 近似。正如我们上节所说, 如果我们有,

$$f(x) \approx \sum_{n=0}^N c_n T_n(x)$$

那么我们同样可以获得 $f'(x)$ 的 Chebyshev 展开式,

$$f'(x) \approx \sum_{n=0}^N c'_n T_n(x)$$

其中系数 c'_n 与原先系数之间关系为

$$c'_N = c'_{N-1} = 0, \quad (n = N-1, N-2, \dots, 1)$$
$$c'_{n-1} = c'_{n+1} + 2nc_n,$$

那么我们就可通过 c_n 把与一阶导数相关的系数 c'_n 求出来。事实上, **有了函数的 Chebyshev 近似式, 除了可以计算它的导数 (微分) 之外, 还可以计算其积分。这就获得了一个近似的积分表达式。**

NUMERICAL DIFFERENTIATION OF APPROXIMATED FUNCTIONS WITH LIMITED ORDER-OF-ACCURACY DETERIORATION*

O. BRUNO[†] AND D. HOCH[†]

$$(5) \quad P_N(t) = \sum_{j=0}^{N-1}{}' \bar{c}_j T_j(t) \quad \text{satisfying} \quad P_N(t_k) = \bar{f}_k \quad \text{for} \quad k = 1, \dots, N,$$

where $T_j(t)$ is the Chebyshev polynomial of degree j ; see, e.g., [3, 17, 18].

Remark 1. Here and throughout the paper we use the usual convention according to which the prime in the summation symbol indicates that the contribution of the $j = 0$ term to the sum is $1/2$ of its value: $\sum' a_j = \frac{1}{2}a_0 + \sum_{j \neq 0} a_j$.

数值微分的计算

Remark 3. Our algorithm evaluates the needed derivatives of Chebyshev expansions (5) by means of the expression [18]

$$(11) \quad P'_N(t) \equiv \sum_{k=0}^{N-1} \bar{c}'_k T_k(t),$$

where the coefficients \bar{c}'_k satisfy the descending recurrence relation

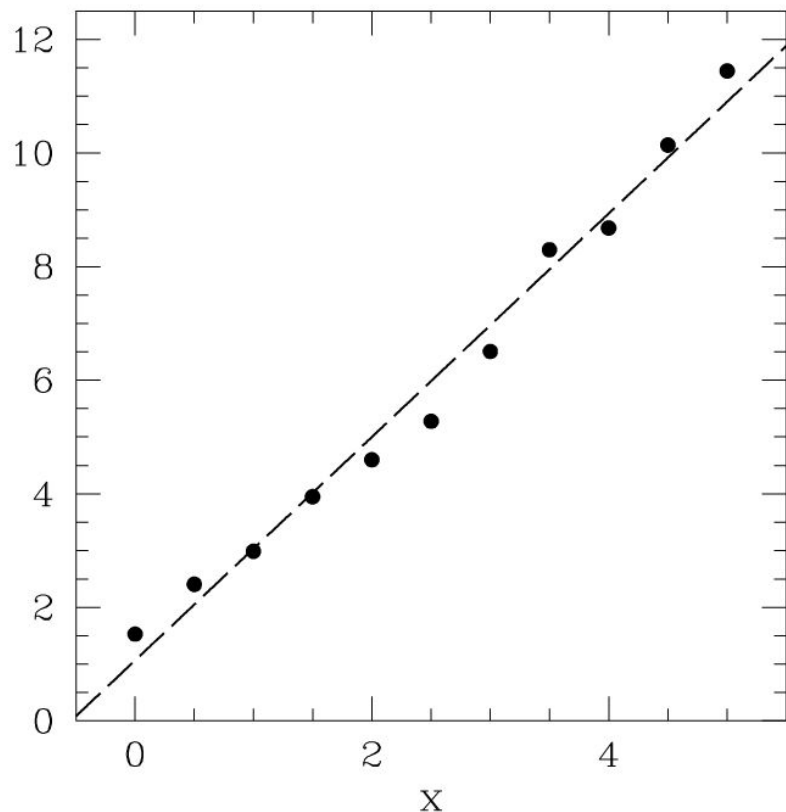
$$(12) \quad \begin{aligned} \bar{c}'_{N-1} &= 0, \\ \bar{c}'_{N-2} &= 2(N-1)\bar{c}_{N-1}, \\ \bar{c}'_{k-1} &= 2k\bar{c}_k + \bar{c}'_{k+1}, \quad k = N-2, N-3, \dots, 1. \end{aligned}$$

Needed derivatives of order m ($m = 2, 3, \dots$), in turn, are obtained by iterated application of the $m = 1$ procedure (11)–(12).

最小二乘法

数值上我们经常遇到拟合的问题，与前面讨论的插值问题不同的是：

- 拟合的自由参数(自由度)可能小于节点数；
- 在每个节点处，我们并不要求拟合曲线准确穿过节点；
- 函数值带有误差
- ...



$(x_i, y_i), i = 1, 2, \dots, N$

$y = f(x)$ 含有一些可调整的参数

“least squares” 拟合：

最小化

$$\sum_{i=1}^N [y_i - f(x_i)]^2 .$$

最小二乘法: 线性及非线性模型

$$y = a_0 + a_1 x$$

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m = \sum_{\alpha=0}^m a_{\alpha} x^{\alpha}$$

a_{α} 是可调节的自由参数。注意,“线性”是指 y 是 a_{α} 的线性函数,并不是说 y 是 x 的线性函数。

$$y = a_0 x^{a_1} + a_2$$

以上则是非线性模型的一个例子。
对于least-squares拟合,我们将会看到,线性模型的自由参数将通过线性方程得到求解。对于非线性模型的自由参数,通常涉及非线性方程,其求解不是那么容易。

$$\ln y = \ln a_0 + a_1 \ln x,$$

有时非线性模型可以变换为线性模型,如log-log变换。然而,在考虑函数值误差时,需要注意误差分布可以发生变化,例如高斯分布不一定会被保持。

Levenberg–Marquardt
algorithm (Levenberg–
Marquardt-Algorithmus)



In mathematics and computing, the Levenberg–Marquardt algorithm, also known as the damped least-squares method, is used to solve non-linear least squares problems. These minimization problems arise especially in least squares curve fitting.

最小二乘法: 直线拟合

$$F(a_0, a_1) = \sum_{i=1}^N (y_i - a_0 - a_1 x_i)^2,$$

让F对自由参数 a_0 和 a_1 微分, 并要求微分为0:

$$\begin{aligned}\sum_{i=1}^N (a_0 + a_1 x_i) &= \sum_{i=1}^N y_i, \\ \sum_{i=1}^N x_i (a_0 + a_1 x_i) &= \sum_{i=1}^N x_i y_i.\end{aligned}$$

可以写成线性方程组形式

$$U_{00} a_0 + U_{01} a_1 = v_0,$$

$$U_{10} a_0 + U_{11} a_1 = v_1,$$

$$U_{\alpha\beta} = \sum_{i=1}^N x_i^{\alpha+\beta},$$

$$v_{\alpha} = \sum_{i=1}^N y_i x_i^{\alpha}.$$

$U_{01}=U_{10}$
U是对称矩阵

最小二乘法: 直线拟合

解线性方程后, 可得:


$$a_{\alpha} = \sum_{\beta=0}^m (U^{-1})_{\alpha\beta} v_{\beta},$$

$$U^{-1} = \frac{1}{\Delta} \begin{pmatrix} U_{11} & -U_{01} \\ -U_{01} & U_{00} \end{pmatrix}$$

$$\Delta = U_{00}U_{11} - U_{01}^2,$$

$$a_0 = \frac{U_{11} v_0 - U_{01} v_1}{\Delta},$$

$$a_1 = \frac{-U_{01} v_0 + U_{00} v_1}{\Delta}.$$


$$y = a_0 + a_1 x$$

最小二乘法：多项式拟合

$$y = a_0 + a_1x + a_2x^2 + \cdots + a_mx^m = \sum_{\alpha=0}^m a_{\alpha}x^{\alpha}$$

最小化：

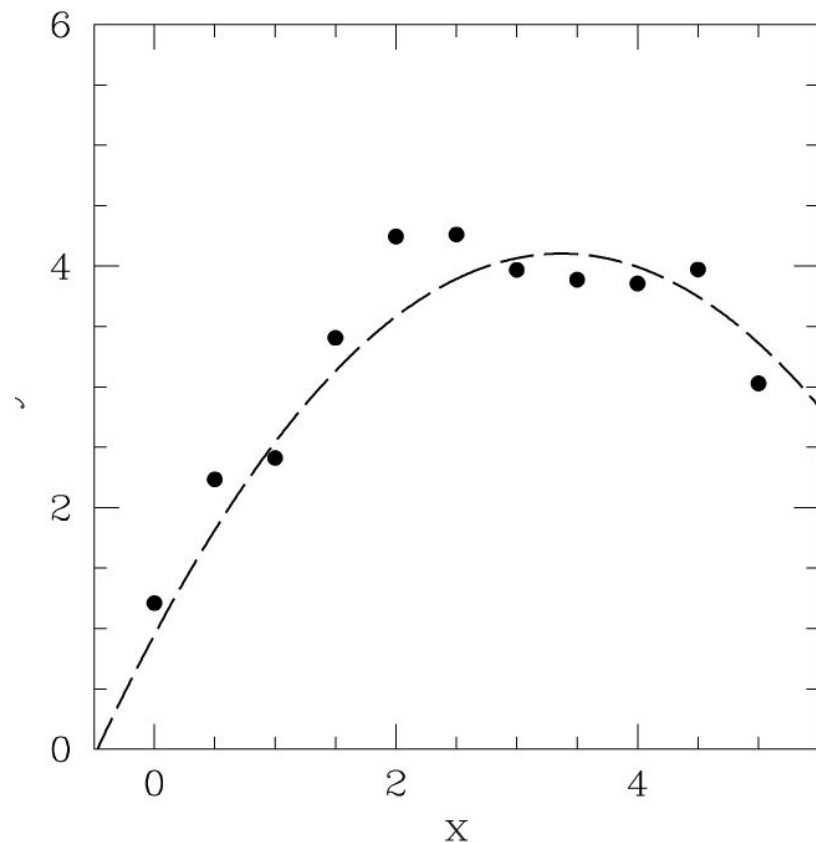
$$F(a_0, a_1, \cdots, a_m) = \sum_{i=1}^N \left(y_i - \sum_{\alpha=0}^m a_{\alpha} x_i^{\alpha} \right)^2$$

F对 a_i 微分为0：

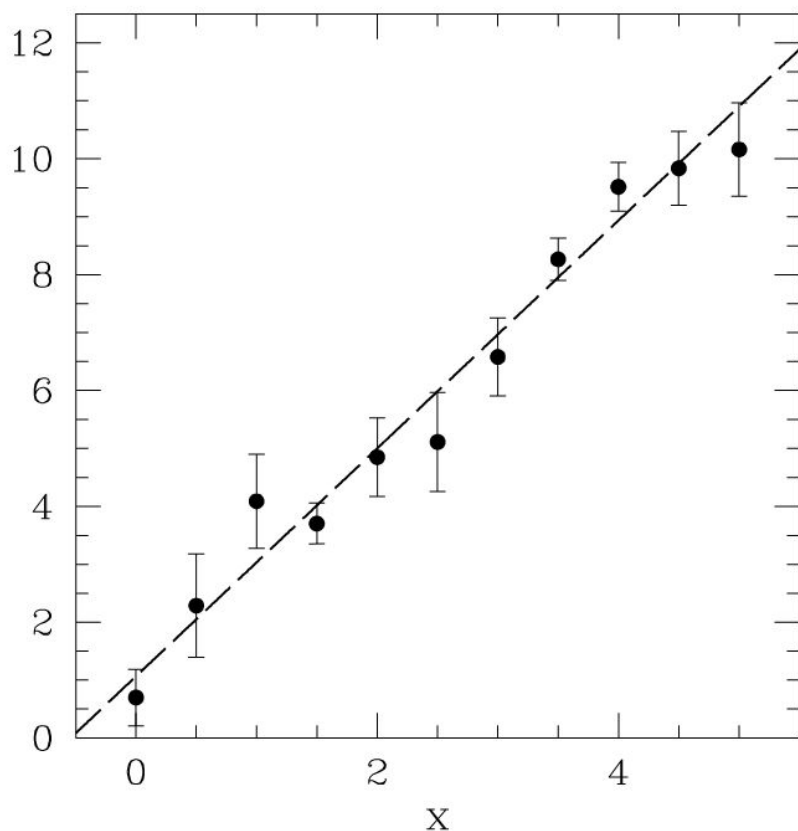
$$\sum_{i=1}^N x_i^{\alpha} \left(y_i - \sum_{\beta=0}^m a_{\beta} x_i^{\beta} \right) = 0,$$



$$\sum_{\beta=0}^m U_{\alpha\beta} a_{\beta} = v_{\alpha},$$



最小二乘法:函数值有误差(error-bar)情形



对于误差较小的数据点, 我们更倾向于让拟合函数靠近它们, 因而, 一个更加合适的最小化目标函数为:

$$\chi^2 = \sum_{i=1}^N \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2,$$

与之前类似, 如果我们采用多项式函数模型去拟合, 可以得到:

$$\sum_{\beta=0}^m U_{\alpha\beta} a_{\beta} = v_{\alpha},$$

其中

$$U_{\alpha\beta} = \sum_{i=1}^N \frac{x_i^{\alpha+\beta}}{\sigma_i^2},$$

$$v_{\alpha} = \sum_{i=1}^N \frac{y_i x_i^{\alpha}}{\sigma_i^2}.$$

最小二乘法:函数值有误差(error-bar)情形

y_i 在 σ_i 误差范围内变动, 将使得拟合参数值 a_α 发生改变:

$$\sigma_\alpha^2 = \langle \delta a_\alpha^2 \rangle \quad \delta a_\alpha = a_\alpha - \langle a_\alpha \rangle$$

其中 $\langle \cdots \rangle$ 表示对不同的 y_i 得到的结果进行加权平均。

可证:

$$\sigma_\alpha^2 = (U^{-1})_{\alpha\alpha} ,$$

covariance matrix

$$\text{Cov}(\alpha, \beta) \equiv \langle \delta a_\alpha \delta a_\beta \rangle = (U^{-1})_{\alpha\beta} .$$

$$r_{\alpha\beta} = \frac{\text{Cov}(\alpha, \beta)}{\sigma_\alpha \sigma_\beta} .$$

correlation coefficient

最小二乘法:卡方值

在后面我们还将详细介绍统计概率分布。卡方分布、高斯分布是科研工作中经常遇到的分布函数。

在我们的问题中, 拟合的好坏可以通过卡方值及卡方分布定量描述。

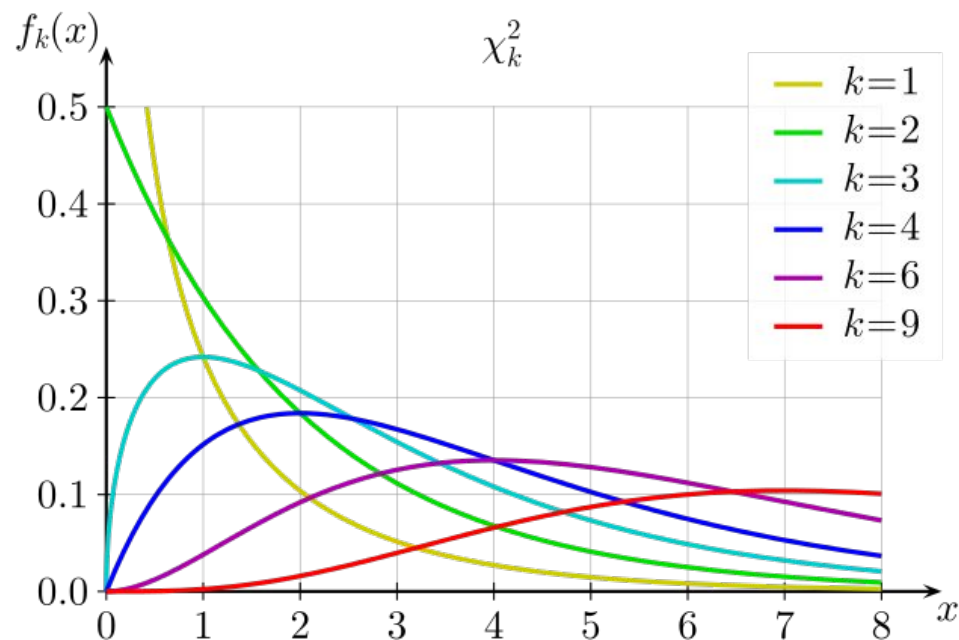
$$\chi^2 = \sum_{i=1}^N \left(\frac{y_i - f(x_i)}{\sigma_i} \right)^2,$$

to get the *best* fit. It the number of fit parameters, N_{fit} is equal to the number of data points, then we can clearly get the fit to go *exactly* through the data, so χ^2 would be zero. Hence the relevant quantity is not N but rather, it turns out, the difference $N_{\text{DOF}} = N - N_{\text{fit}}$, which is called the **number of degrees of freedom** (DOF). (When fitting to an m -th order polynomial, $N_{\text{fit}} = m + 1$ so $N_{\text{DOF}} = N - m - 1$.)

For a good fit, we expect that χ^2 should be about N_{DOF} . To be precise, the mean value of χ^2 (averaged over many sets of data from the same distribution as the one set of data we actually have) is equal to N_{DOF} .

最小二乘法:卡方分布

在后面我们还将详细介绍统计概率分布。卡方分布、高斯分布是科研工作中经常遇到的分布函数。



To summarize, a fitting program should provide the following information (assuming that error bars are given on the points):

1. The values of the fitting parameters.
2. Error bars on those parameters.
3. A measure of the goodness of fit.