由于突发恶疾住院若干天，故本篇记录时隔一整月.

本篇继续讨论函数插值的问题

函数插值法

在前述的两种插值法中，我们都构造了一个函数，使得至少在各个节点处与潜在的函数$f(x)$同值. 但如果我们想要通过插值函数$H(x)$来刻画$f(x)$更加深刻的性质，如一阶导数，那么前述的两种插值方法将难以胜任.

Hermite插值

我们试图找到某插值函数$H(x)$使得在$(n+1)$个点的点集上，有：

$$\forall i \in [0,n] : H(x_i) = y_i, H'(x_i) = y_i' \tag{5.1}$$

直观地，在$H(x)$中蕴含了更多的信息，这样一来就可能需要更多的自由度来承载这些信息，当然我们只有$(2n+2)$个约束条件，因此天然地，Hermite插值不可以也不会超过$(2n+1)$次.

基函数构造法

与在Lagrange插值法中作的工作相似，我们同样希望找到一组基，使得它满足这一问题中的要求：我们将上述的问题分开处理：

首先找到某一类函数$h_i(x)$，使得对于样本点集中的点(x_j, y_j)有：

$$h_i(x_j) = 0, j \neq i$$

$$h_i(x_j) = 1, j = i$$

同时满足：它的一阶导数在样本点集上恒为0：

$$h_i'(x_j) \equiv 0$$

这样，我们即可使用$h(x)$来控制插值函数的函数值而不影响其导数.

同样地，我们试图找到$H_i(x)$，使得对于样本点集中的点(x_j, y_j)有：

$$H_i'(x_j) = 0, j \neq i$$

$$H_i'(x_j) = 1, j = i$$

同时，它的原值应当在样本点集上恒为0：

$$H_i(x_j) \equiv 0$$

这样，我们即可使用$h(x)$来控制插值函数的一阶导数而不影响其原值.

幸运的是，数学家们创造性地发现了这组基的形式：

$$\begin{cases} h_i(x) = [a + b(x - x_i)]l_i^2(x) \\ a = 1 \\ b = -2l_i'(x_i) \end{cases}$$

$$\begin{cases} H_i(x) = cl_i^2(x) \\ c = 1 \end{cases}$$

其中，$l_i(x)$是关于x_i的Lagrange基：

$$l_i(x) = \frac{\prod_{j=0, j\neq i}^{n}(x - x_j)}{\prod_{j=0, j\neq i}^{n}(x_i - x_j)}$$

因此，在该方法中，每一个点-导数三元组(x_i, y_i, y_i')生成一项

$$y_i[1 - 2l_i'(x_i)(x - x_i)]l_i^2(x) + y_i'(x - x_i)l_i^2(x)$$

由此可给出Hermite插值表达式：

$$H(x) = \sum_{i=0}^{n}\{y_i[1 - 2l_i'(x_i)(x - x_i)]l_i^2(x) + y_i'(x - x_i)l_i^2(x)\}$$

Hermite插值法拥有误差表达式

$$R_n(x) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\omega_{n+1}^2(x)$$

这样的“函数值-导数”分别控制的方法有着更广阔的应用空间，如给出点集中部分拥有导函数值而其余没有，而这种形式的问题需要构造额外的基函数(如使用原基函数则待定参数数量过多)，而构造这一基函数的过程需要较强的分析要求，故不建议使用基函数构造法. (课后作业习题9涉及到这一方法，可以试着做一下)

对于上述这种“不完全”的情况，误差表达式会有所变化，但总得来说，如果给出了n组原值条件，其中有m组限制了导数值(不妨设它们就是第$0 \to (m-1)$个)，则误差可以表示为

$$R_n(x) = \frac{f^{(n+m+2)}(\xi)}{(n+m+2)!}\omega_{n+1}(x)\prod_{i=0}^{(m-1)}(x - x_i)$$

待定系数法

这种方法直接将$H(x)$显式地假设出来，再代入点集解方程组(5.1). 严格地来讲这种算法不能称之为方法，但在数据量较小时很好用.

降阶法

而面对稍多的数据时，待定系数法就不那么好用了. 因此我们试图通过某种方式来减少需要处理的自由度数：

我们先通过前述的插值法求出仅在原函数值上提供保证的插值函数：

$$N(x), s.t. \forall i \in [0, n], N(x_i) = y_i$$

明显地，Hermite插值函数与$N(x)$存在如下的关系：

$$H(x) - N(x) = P(x) \prod_{i=0}^{n} (x - x_i)$$

而此时，由于$H(x)$是$2n+1$阶多项式，而$N(x)$是n阶多项式，故$H(x) - N(x)$是$2n+1$阶多项式，故$P(n)$是n阶多项式.

此时再利用$H'(x_i) = y_i$对$P(n)$待定系数直球求解即可.

样条插值法

函数逼近

函数逼近问题给出一目标函数$f(x)$，希望提出一(容易处理的)近似函数$\phi(x)$来代替$f(x)$，以近似刻画并简化对$f(x)$的诸多运算.

而插值问题是给出一系列点$\{(x_i, y_i)\}$，希望提出一拟合函数$\phi(x)$以近似刻画这些点所隐含的函数关系$f(x)$.

函数逼近法的总体过程

1. 提出近似函数$\phi(x)$的形式并预留一些待定系数$\{a_0, a_1, a_2, ...\}$
2. 使用某种误差估计手段写出误差E关于上述待定系数$\{a_0, a_1, a_2, ...\}$的关系
3. 通过最小化误差E求出$\{a_0, a_1, a_2, ...\}$

由于其有着便于计算的性质，我们通常使用多项式函数$\phi(x) = \sum_{i=0}^{n} a_i x^i$来进行拟合，同时使用最小二乘法来描述误差.

在某些情况下，可能会将自变量x映射为其它的函数，如设置原型$y = a_0 + a_1 e^x + a_2 e^{2x}$逼近，则可以对逼近点列进行变换如$(e^{x_0}, y_0)$.

最小二乘法

“最小二乘”是一种描述误差的尺度，即使用均方误差(假使在点集P上考察)，其"误差泛函"表示如下：（注：仅仅是形式不同的事物不需要讨论多次，故此处的求和符号可以为离散求和，也可以为连续积分）

$$E[\phi_A(x)] = \sum_{(x,y)\in P} (f(x) - \phi(x))^2$$

我们找出某种手段来最小化这个E即可. 幸运的是，我们在针对“形式固定-系数待定”的多项式的讨论中不需要面对上述令人生畏的泛函，而可以将其转写为：

$$E(a_0, a_1, a_2, ..., a_n) = \sum_{(x,y)\in P} [y - (\sum_{i=0}^{n} a_i x^i)]^2$$

特别地，这种最小二乘意义下的误差是一串平方和，这意味着它必定有一个最小值，以下简单**证明**（这证明是我自己写的，虽然还挺漂亮的，但是可能有逻辑断裂可以跳过不看）

假定有一$(n+1)$个相互独立的自变量之函数：$f(a_0, a_1, a_2, ..., a_n) = \sum_{i=0}^{n} a_i^2$

易知$\forall i \in [0, n], i = 0$时函数$f$取得极值，而此时其Hessian矩阵为$diag(2, 2, 2, 2, ..., 2)$，明显是正定的，因此它在此处取最小值.

我们设置$A = (a_0, a_1, a_2, ..., a_n)$，那么对于任一和$A$无关的$(n+1)$个向量$x_i$与标量$y_i (i \in [0, n])$，我们产生一个新的$(n+1)$维向量$(y_0 + Ax_0, y_1 + Ax_1, y_2 + Ax_2, ..., y_n + Ax_n)$，易知这个向量仅仅是将向量$A$在其空间内平移、放缩而未旋转，未实际改变该空间的拓扑性质，因此f在新的自变量列$(y_0 + Ax_0, y_1 + Ax_1, y_2 + Ax_2, ..., y_n + Ax_n)$下仍拥有一唯一最小值点.

而此时，f恰可以被写为$f(a_0, a_1, a_2, ..., a_n) = \sum_{(x,y)\in P} [y - (\sum_{i=0}^{n} a_i x^i)]^2$

■

因此这提示了我们最小二乘法的解决策略：对于误差函数$E(a_0, a_1, a_2, ..., a_n) = \sum_{(x,y)\in P} [y - (\sum_{i=0}^{n} a_i x^i)]^2$，我们求：

$$
\begin{cases}
\dfrac{\partial E}{\partial a_0} = -2 \sum\limits_{(x,y)\in P} (y - \sum\limits_{j=0}^{n} a_j x^j) \\
\dfrac{\partial E}{\partial a_1} = -2 \sum\limits_{(x,y)\in P} (y - \sum\limits_{j=0}^{n} a_j x^j) x \\
... \\
\dfrac{\partial E}{\partial a_k} = -2 \sum\limits_{(x,y)\in P} (y - \sum\limits_{j=0}^{n} a_j x^j) x^k \\
... \\
\dfrac{\partial E}{\partial a_n} = -2 \sum\limits_{(x,y)\in P} (y - \sum\limits_{j=0}^{n} a_j x^j) x^n
\end{cases}
$$

用更为通俗的方式展开之：

$$
\begin{cases}
\dfrac{\partial E}{\partial a_0} = -2 \sum\limits_{(x,y)\in P} [y - (a_0 + a_1 x + a_2 x^2 + ... + a_n x^n)] \\
\dfrac{\partial E}{\partial a_1} = -2 \sum\limits_{(x,y)\in P} [y - (a_0 + a_1 x + a_2 x^2 + ... + a_n x^n)] x \\
... \\
\dfrac{\partial E}{\partial a_k} = -2 \sum\limits_{(x,y)\in P} [y - (a_0 + a_1 x + a_2 x^2 + ... + a_n x^n)] x^k \\
... \\
\dfrac{\partial E}{\partial a_n} = -2 \sum\limits_{(x,y)\in P} [y - (a_0 + a_1 x + a_2 x^2 + ... + a_n x^n)] x^n
\end{cases}
$$

令其全部为0以找到最小值，整理得到方程组

$$
\begin{cases}
\sum\limits_{(x,y)\in P} y = \sum\limits_{(x,y)\in P} a_0 + \sum\limits_{(x,y)\in P} x a_1 + \sum\limits_{(x,y)\in P} x^2 a_2 + ... + \sum\limits_{(x,y)\in P} x^n a_n \\
\sum\limits_{(x,y)\in P} xy = \sum\limits_{(x,y)\in P} x a_0 + \sum\limits_{(x,y)\in P} x^2 a_1 + \sum\limits_{(x,y)\in P} x^3 a_2 + ... + \sum\limits_{(x,y)\in P} x^{n+1} a_n \\
... \\
\sum\limits_{(x,y)\in P} x^k y = \sum\limits_{(x,y)\in P} x^k a_0 + \sum\limits_{(x,y)\in P} x^{k+1} a_1 + \sum\limits_{(x,y)\in P} x^{k+2} a_2 + ... + \sum\limits_{(x,y)\in P} x^{k+n} a_n \\
... \\
\sum\limits_{(x,y)\in P} x^n y = \sum\limits_{(x,y)\in P} x^n a_0 + \sum\limits_{(x,y)\in P} x^{n+1} a_1 + \sum\limits_{(x,y)\in P} x^{n+2} a_2 + ... + \sum\limits_{(x,y)\in P} x^{2n} a_n
\end{cases}
$$

写为矩阵形式则为：

$$MA = b$$

$$M = \begin{pmatrix} \sum_{(x,y)\in P} 1 & \sum_{(x,y)\in P} x & \sum_{(x,y)\in P} x^2 & \cdots & \sum_{(x,y)\in P} x^n \\ \sum_{(x,y)\in P} x & \sum_{(x,y)\in P} x^2 & \sum_{(x,y)\in P} x^3 & \cdots & \sum_{(x,y)\in P} x^{n+1} \\ \vdots & \vdots & \vdots & & \vdots \\ \sum_{(x,y)\in P} x^k & \sum_{(x,y)\in P} x^{k+1} & \sum_{(x,y)\in P} x^{k+2} & \cdots & \sum_{(x,y)\in P} x^{k+n} \\ \vdots & \vdots & \vdots & & \vdots \\ \sum_{(x,y)\in P} x^n & \sum_{(x,y)\in P} x^{n+1} & \sum_{(x,y)\in P} x^{n+2} & \cdots & \sum_{(x,y)\in P} x^{2n} \end{pmatrix}$$

$$b = \begin{pmatrix} \sum_{(x,y)\in P} y \\ \sum_{(x,y)\in P} xy \\ \cdots \\ \sum_{(x,y)\in P} x^k y \\ \cdots \\ \sum_{(x,y)\in P} x^n y \end{pmatrix}$$

此方程组被称为**正则方程组**. 它依据求和方式的不同，可以有两种表示形式，分别对应在离散点集上逼近和连续点集上逼近两种情况：

1. 离散点集逼近：

$$M = \begin{pmatrix} \sum_{i=0}^{m} 1 & \sum_{i=0}^{m} x & \sum_{i=0}^{m} x^2 & \cdots & \sum_{i=0}^{m} x^n \\ \sum_{i=0}^{m} x & \sum_{i=0}^{m} x^2 & \sum_{i=0}^{m} x^3 & \cdots & \sum_{i=0}^{m} x^{n+1} \\ \vdots & \vdots & \vdots & & \vdots \\ \sum_{i=0}^{m} x^k & \sum_{i=0}^{m} x^{k+1} & \sum_{i=0}^{m} x^{k+2} & \cdots & \sum_{i=0}^{m} x^{k+n} \\ \vdots & \vdots & \vdots & & \vdots \\ \sum_{i=0}^{m} x^n & \sum_{i=0}^{m} x^{n+1} & \sum_{i=0}^{m} x^{n+2} & \cdots & \sum_{i=0}^{m} x^{2n} \end{pmatrix}$$

$$b = \begin{pmatrix} \sum_{i=0}^{m} y \\ \sum_{i=0}^{m} xy \\ \cdots \\ \sum_{i=0}^{m} x^k y \\ \cdots \\ \sum_{i=0}^{m} x^n y \end{pmatrix}$$

2. 连续区间逼近

$$M=\begin{pmatrix}\int_a^b 1dx & \int_a^b xdx & \int_a^b x^2dx & ... & \int_a^b x^ndx\\ \int_a^b xdx & \int_a^b x^2dx & \int_a^b x^3dx & ... & \int_a^b x^{n+1}dx\\ \vdots & \vdots & \vdots & & \vdots\\ \int_a^b x^kdx & \int_a^b x^{k+1}dx & \int_a^b x^{k+2}dx & ... & \int_a^b x^{k+n}dx\\ \vdots & \vdots & \vdots & & \vdots\\ \int_a^b x^ndx & \int_a^b x^{n+1}dx & \int_a^b x^{n+2}dx & ... & \int_a^b x^{2n}dx\end{pmatrix}$$

$$b=\begin{pmatrix}\int_a^b f(x)dx\\ \int_a^b xf(x)dx\\ ...\\ \int_a^b x^kf(x)dx\\ ...\\ \int_a^b x^nf(x)dx\end{pmatrix}$$

非多项式逼近，一般函数族

在上述的讨论中，我们设置逼近函数为$\phi(x)=\sum_{i=0}^n a_ix^i$，即我们使用了函数族$\{1,x,x^2,...,x^n,...\}$的某一线性组合来逼近目标函数$f(x)$，且逼近点集上的各个点有着相同的"重要程度".

我们将这个过程推广开来：我们使用一般的函数族$\{\phi_0(x),\phi_1(x),\phi_2(x),...,\phi_n(x),...\}$来刻画目标函数$f(x)$，且不同的逼近锚点$(x,y)$拥有不同的非负非全零(实际上权函数有严格的约束，但我们可以理解为非负非全零(反正不会考))的"重要性"权函数$\omega(x)$，那么，我们设置逼近函数为$\phi(x)=\sum_{i=0}^n a_i\phi_i(x)$，在带权点集$P$下，其"最小二乘"尺度下的误差可以写作：

$$E(a_0,a_1,a_2,...,a_n)=\sum_{(x,y)\in P}\omega(x)[y-(\sum_{i=0}^n a_i\phi_i(x))]^2$$

（推导过程简单，略）为了能使矩阵变得好看一点，我们简记$(A,B)=\sum_{(x,y)\in P}\omega(x)A(x)B(x)$则正则方程组可以写作

$$MA=b$$

$$M=\begin{pmatrix}(\phi_0,\phi_0) & (\phi_0,\phi_1) & (\phi_0,\phi_2) & ... & (\phi_0,\phi_n)\\ (\phi_1,\phi_0) & (\phi_1,\phi_1) & (\phi_1,\phi_2) & ... & (\phi_1,\phi_n)\\ \vdots & \vdots & \vdots & & \vdots\\ (\phi_i,\phi_0) & (\phi_i,\phi_1) & (\phi_i,\phi_2) & ... & (\phi_i,\phi_n)\\ \vdots & \vdots & \vdots & & \vdots\\ (\phi_n,\phi_0) & (\phi_n,\phi_1) & (\phi_n,\phi_2) & ... & (\phi_n,\phi_n)\end{pmatrix}$$

$$b=\begin{pmatrix}(\phi_0,f)\\(\phi_1,f)\\\cdots\\(\phi_2,f)\\\cdots\\(\phi_n,f)\end{pmatrix}$$

为了使得上述方程恰定，矩阵M须满秩，这对函数族提出了要求(iff指“当且仅当”):

$$\sum_{i=0}^{n}a_i\phi_i(x)=0\ \ iff.\forall i\in[0,n],a_i=0$$

这一性质被称为函数族"线性无关".

正交函数族

解线性方程组是痛苦的，故我们对上述方程组可以作出更进一步的幻想：“如果矩阵M是一个对角矩阵就再好不过了”. 这样的幻想要求函数族ϕ_i和权函数$\omega(x)$具有这样的性质:

$$\forall i,j\in[0,n]:i\neq j\iff\sum_{(x,y)\in P}\omega(x)\phi_i(x)\phi_j(x)=0$$

$$\forall i\in[0,n]:\sum_{(x,y)\in P}\omega(x)\phi_i(x)\phi_i(x)=a_i>0$$

如有此性质，则称函数族$\{\phi_i\}$在域P上关于权函数$\omega(x)$正交. 易知一正交函数族所导出的正则方程组之系数矩阵必满秩，故正则方程组必有唯一解.

特别地，若对于任意的i，上述$a_i=1$，则称其为**正交-归一化函数**族. 正交-归一化函数族不仅在函数逼近中出现，在诸多领域中均有应用(如量子化学中的本征波函数)

既然正交函数族如此好用，那么是否可以将设置出的简单的非正交函数族如$\{x^i\}$转化为正交函数族呢?

构造正交函数族，Gram-Schmidt正交化方法

有一线性无关函数族$\{\phi_i(x)\}$，重构一新的函数族$\{\Phi_i(x)\}$:

$$\Phi_0(x)=\phi_0(x)$$

$$\Phi_i(x)=\begin{vmatrix}(\phi_0,\phi_0)&(\phi_0,\phi_1)&\cdots&(\phi_0,\phi_{i-1})&\phi_0(x)\\(\phi_1,\phi_0)&(\phi_1,\phi_1)&\cdots&(\phi_1,\phi_{i-1})&\phi_1(x)\\\vdots&\vdots&&\vdots&\vdots\\(\phi_i,\phi_0)&(\phi_i,\phi_1)&\cdots&(\phi_i,\phi_{i-1})&\phi_i(x)\end{vmatrix}$$

此时

$$(\Phi_i, \Phi_j) = \begin{cases} 0, i \neq j \\ \Delta_i, i = j \end{cases}$$

其中

$$\Delta_i = \begin{vmatrix} (\phi_0, \phi_0) & (\phi_0, \phi_1) & \cdots & (\phi_0, \phi_{i-1}) & (\phi_0, \phi_i) \\ (\phi_1, \phi_0) & (\phi_1, \phi_1) & \cdots & (\phi_1, \phi_{i-1}) & (\phi_1, \phi_i) \\ \vdots & \vdots & & \vdots & \vdots \\ (\phi_i, \phi_0) & (\phi_i, \phi_1) & \cdots & (\phi_i, \phi_{i-1}) & (\phi_i, \phi_i) \end{vmatrix}$$

课本151页给出了Gram-Schmidt正交化方法的一应用实例，在153页起给出了若干常用的正交函数族.