Chapter 3

代数问题的逼近方法

3.1 插值问题与函数逼近

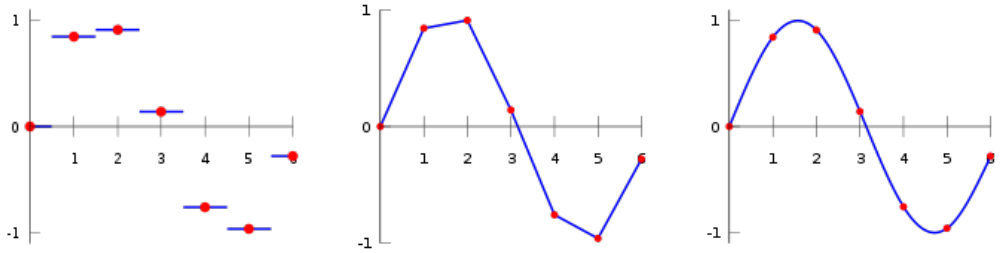
插值问题的来源于各类实际应用，如要求画出一条通过所有离散点的光滑曲线、利用列表函数求中间值、求列表函数的导数或积分、使用简单函数代替复杂函数，从而可以快速方便的求出需要求得的值等等。当然，这些问题又是来自于各类科学与工程计算问题的实际需求中抽象出来的。让用数学的语言进行定义插值问题：

定义3.1.1 (插值问题). 已知数据$\{t_i, y_i\}_{i=0}^n$,寻找函数$f : \mathbb{R} \to \mathbb{R}$,满足：

$$f(t_i) = y_i, \qquad \forall i = 0, 1, ..., n$$

其中f称为插值函数，简称为插值。称$\{t_i\}_{i=0}^n$为插值节点。

如下几个例子描述了如何找到一些“简单”函数来“穿过”给定的数据点： 插值函数的选择



主要考虑因素是多方面的。首先，函数的简单程度是考虑的重要因素，因为越简单的函数越有利于计算机进行求值等各种运算，如多项式是计算机最擅长的数学工具。其次，也要考虑与拟合数据在性态方面的接近程度（如：光滑性，单调性，周期性等），比如分段多项式、三角函数、指数函数和有理函数等也是数值计算中常用的数学工具。

3.1.1 插值一般方法

对于给定的数据点集$(t_i, y_i), i = 1, 2, ..., n$,选择基函数$\phi_1(t), \phi_2(t), ...\phi_n(t)$,在这组基函数所张

成的函数空间中选择一个插值函数。将插值函数f写成这些基函数的线性组合：

$$f(t) = \sum_{j=1}^{n} x_j \phi_j(t),$$

其中x_j是待定的参数，则数据点(t_i, y_i)上的插值函数f应满足：

$$f(t_i) = \sum_{j=1}^{n} x_j \phi_j(t_i) = y_i,$$

将其写成$\mathbf{Ax} = \mathbf{y}$的矩阵形式，$\mathbf{A}$的元素为$a_{ij} = \phi_j(t_i)$。
利用上述形式，结合线性代数的知识我们有如下结论:

性质3.1.1 (存在唯一性)**.** 若基函数个数n与数据个数m相等，则得到的是一个方阵线性方程组。若矩阵$\mathbf{A}$非奇异，则一定有且仅有唯一解。而若矩阵$\mathbf{A}$奇异，则可以有许多参数的解，代表着数据点不能被精确拟合。

性质3.1.2 (病态性)**.** 基函数可以有许多选择的方式，相对应的会有许多矩阵$\mathbf{A}$的表达形式。$\mathbf{A}$若是单位阵，下三角矩阵，三对角矩阵等等特殊的矩阵，会大大提升求解参数的效率，降低求解的难度，在之后的具体例子里有所体现。

3.1.2 多项式插值

单项式基底

最简单直接的基底是单项式基底，即对于n个数据点，进行选取k=n-1，则有单项式基底：

$$\phi_j(t) = t^{j-1}, \quad j = 1, 2, ..., n,$$

任何n-1次多项式，皆可用这个基底的线性组合来表示。此时，插值多项式的参数可由下述方程组求解：

$$\mathbf{Ax} = \begin{bmatrix} 1 & t_1 & ... & t_1^{n-1} \\ 1 & t_2 & ... & t_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & ... & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{y}$$

这里，$\mathbf{A}$是一个范德蒙矩阵，当$x_0 < x_1 < ... < x_n$时非奇异。单项式基底具有存在及唯一性，因此解此方程组需要的工作量是$\mathbf{O}(n^3)$, 计算成本高！

拉格朗日插值

对给定的数据点$(t_i, y_i), i = 1, 2, ..., n$,n-1次拉格朗日基函数可以表示为：

$$l_j(t) = \frac{\prod_{k=1, k\neq j}^{n}(t - t_k)}{\prod_{k=1, k\neq j}^{n}(t_j - t_k)}, j = 1, 2, ..., n,$$

显然，我们有结论：

$$l_j(t) = \begin{cases} 1, & i = j, \\ 0, & i \neq j \end{cases} i, j = 1, 2, \ldots, n,$$

说明对于拉格朗日插值来说，$\mathbf{A}$ 是单位矩阵$\mathbf{I}$，因此参数$x_1, x_2, \ldots, x_n$可以直接由y_i得到。拉格朗日插值求参数是很容易的，但是同单项式基底表达式相比，它基函数形式更加复杂，并且积分与微分的操作会困难许多。

牛顿插值

对给定的数据点$(t_i, y_i), i = 1, 2, ..., n$, n-1次牛顿基函数为：

$$\pi_j(t) = \prod_{k=1}^{j-1}(t - t_k), \qquad \forall j = 1, 2, ..., n,$$

注意到k始终比j小1，也就是说，多项式可以表示为：

$$Q_n(x) = x_1 + x_2(t - t_1) + x_3(t - t_1)(t - t_2) + \cdots + x_n(t - t_1)(t - t_2)\cdots(t - t_{n-1}). \quad (3.1)$$

这里，$\mathbf{A}$是下三角矩阵，解$\mathbf{Ax} = \mathbf{y}$复杂度$O(n^2)$ 同前两种方法相比，牛顿插值既可以节省计算量，又可以节约计算给定点的值的成本。牛顿法的另一大优势在于，它可以逐步进行计算。如果我们需要增加插值点的个数的时候，可以直接计算增加的点即可，这是其他两种方法做不到的。让我们考虑加入点(t_{n+1}, y_{n+1}),利用牛顿法可得新的插值多项式:

$$Q_{n+1}(t) = Q_n(t) + x_{n+1}\pi_{n+1}(t)$$

其中π的定义与之前的基函数相同，且

$$x_{n+1} = \frac{y_{n+1} - Q_n(t_{n+1})}{\pi_{n+1}(t_{n+1})}.$$

而之前两种插值方式只能将所有点再重新计算，再继续求解参数。

这里我们给出一例以体现三种不同插值公式的区别：

例3.1.1 (三种插值公式的基函数)**.** 分别绘制*Langrange,monominal,*以及*Newton*插值公式的基函数图像，并指出它们的区别。若给定数据点$(-1, 1), (0, 0), (1, 1)$，试计算相应的二次插值多项式。

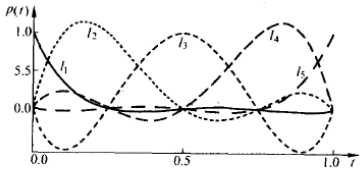


图 7.2 拉格朗日基函数

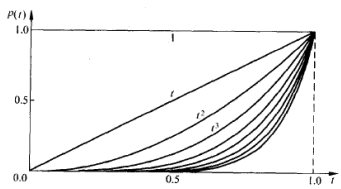


图 7.1 单项式基函数

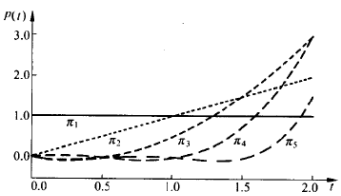


图 7.3 牛顿基函数

差商

差商的定义：$f[x_0,\cdots,x_n]=\dfrac{f[x_1,\cdots,x_n]-f[x_0,\cdots,x_{n-1}]}{x_n-x_0}$

差商的性质:

- $f[x_0,\cdots,x_n]=\dfrac{f^{(n)}(\xi)}{n!}$，其中$\min\{x_i\}<\xi<\max\{x_i\}$;
- 交换不变
- $\frac{\partial}{\partial x}f[x_0,\cdots,x_n,x]=f[x_0,x_1,\cdots,x_n,x,x]$

则可以得到插值多项式：

$$f(x)=P_n(x)+\omega_{n+1}(x)f[x_0,x_1,\cdots,x_n,x]$$

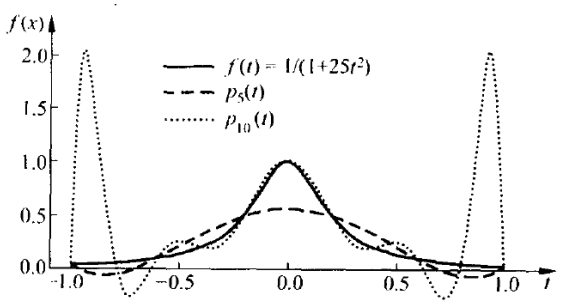
并且此时的误差为：

$$E_{n+1}=f(x)-P_n(x)=\omega_{n+1}(x)f[x_0,x_1,\cdots,x_n,x].$$

其中可以计算出二阶的误差为：$E_2=-\frac{h^2}{8}f''(\xi)$

3.1.3 分段插值

基函数和数据点的合适选择可以减轻高次多项式插值的困难，但有时候用一个多项式来拟合大量数据点，会出现令人不悦的振荡现象。分段多项式主要思想是: 对给定的数据点



集$(t_i,y_i),i=1,\cdots,n,\quad t_1<t_2<\cdots<t_n$,作分段多项式插值时，在每个子区间$[t_i,t_{i+1}]$上用不同的多项式。典型的分段插值方法包括三次样条插值和三次Hermit插值，如下图所示：

采用B样条作为插值基函数有很多优点。

关于回归、插值、逼近、拟合的区别：插值：插值曲线要经过型值点;逼近：为复杂函数寻找近似函数，其误差在某种度量意义下最小;拟合：在插值问题中考虑**给定数据点的误差**，其误差在某种度量意义下最小;回归：最小二乘解,在$\|\cdot\|_2$度量意义下最小。

例3.1.2 (多变量逼近问题：平面网格三角剖分). 平面区域可被一系列直边三角形覆盖.当三角形的外接圆直径趋向于*0*时,空洞区域的面积趋向于*0.*

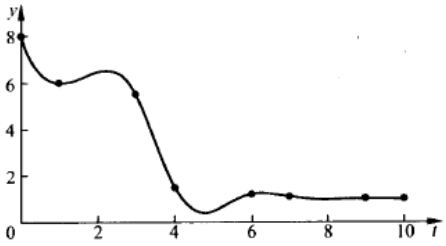


图 7.10 单调数据的三次样条插值

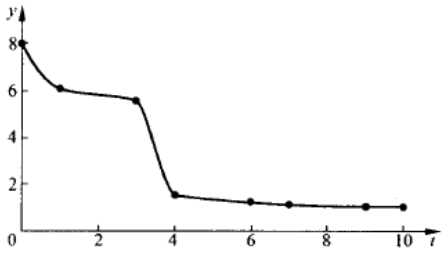


图 7.9 单调数据的三次埃尔米特插值

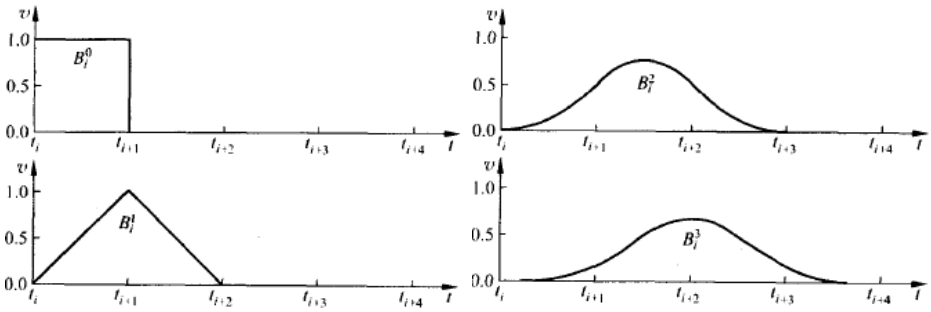
作为递推初始值，定义 0 次 B-样条

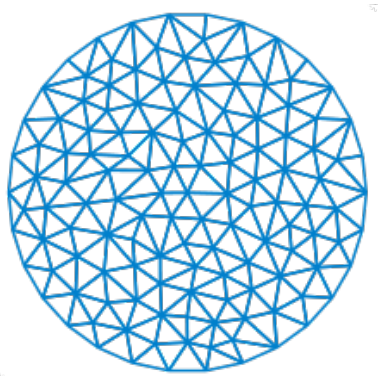
$$B_i^0(t) = \begin{cases} 1, & t_i \leqslant t < t_{i+1}, \\ 0, & \text{其他.} \end{cases}$$

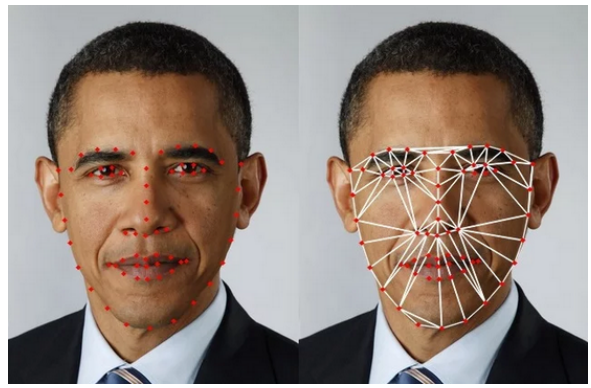
对 $k>0$，定义 k 次 B-样条为

$$B_i^k(t) = v_i^k(t)B_i^{k-1}(t) + (1 - v_{i+1}^k(t))B_{i+1}^{k-1}(t).$$

由于 B_i^0 是分段常数，v_i^k 是线性的，从定义可知，B_i^1 是分段线性的. 类似地，B_i^2 是分段二次







3.2 迭代求根法

给定非线性方程组$\mathbf{f}(\mathbf{x}) = \mathbf{y}$,非线性方程组的问题即为：

定义3.2.1 (求根问题). $\mathbf{x}$为何值时，非线性函数$\mathbf{f}$取值$\mathbf{y}$?

通常，我们会两边同时减去$\mathbf{y}$，使其变为$\mathbf{f}(\mathbf{x}) = \mathbf{0}$的形式。如： 对于一般的$f(x) = 0$，

例 5.1　非线性方程　一维非线性方程

$$f(x) = x^2 - 4\sin x = 0$$

的一个近似解为 $x = 1.93375$. 二维非线性方程组

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1^2 - x_2 + 0.25 \\ -x_1 + x_2^2 + 0.25 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

的解向量为 $\boldsymbol{x} = [0.5, 0.5]^{\mathrm{T}}$.

我们无法给出该非线性方程组解的通用表达式。线性方程组只有$0/1/\infty$个解等情况，非线性方程组解的个数可能是任意个。一个经典的做法是以一维情形为突破线索：

例3.2.1 (几个典型的一维问题). $e^x + 1 = 0$,无解
$e^x - x = 0$,有一个解
$x^2 - 4\sin x = 0$,有两个解
$\sin x = 0$,有无穷多个解

3.2.1 存在唯一性

回顾数学分析中的方法，几个存在性定理是研究方程解存在性的几类典型方法：

引理3.2.1 (介值定理). 若f是闭区间上$[a,b]$的连续函数，c介于$f(a)$和$f(b)$，之间，则必存在一个$x^1 \in [a,b]$，满足$f(x^1) = c$，取c为0即可证明f在$[a,b]$上一定有根。

其次，反函数定理$\mathbf{x} = f^{-1}(\mathbf{0})$。更进一步地，还有压缩映射理论(也是迭代法收敛性的基本定理)也可以获得解的存在性证明.

定义3.2.2 (不动点). f:$\mathbb{R}^n \to \mathbb{R}^n$是集合$S \in \mathbb{R}^n$上的压缩映射，即存在$\gamma \in [0,1]$,对任意集合$S$内的两个点$x$，$z$满足$||f(x) - f(z)|| \le \gamma||x - z||$。满足$g(x) = x$的$x$称为$g$的不动点。

定理3.2.1 (压缩映射(不动点)定理). 若g在闭集S上是压缩映射，且$g(S) \in S$,则g在S内存在唯一的不动点。此时如果非线性方程有形如f=x-g，则我们可称，f=0在闭集S上是有唯一解的。

而关于根的进一步研究，需要借助topological degree of function f与重根理论。

3.2.2 单个方程求根方法

让我们通过讨论标量非线性方程的求根方法来回顾

二分法

假设我们有一个比较小的区间[a,b]，而函数f在此区间上有符号变化，则连续函数f在这个区间内一定有零点。设初始的函数为f，满足f(a)与f(b)的符号不相同，最终区间长度的误差上线为tol。我们可以得到如下二分法的算法。一个Matlab实现：

```
function x = bisect(f,a,b,tol)
  xlow = a;  plow  = f(xlow);
  xhigh = b; phigh = f(xhigh);
  while xhigh - xlow > 2*tol,
    xmid = (xlow + xhigh)/2; pmid = f(xmid);
    if pmid*plow < 0,
      xhigh = xmid;   phigh = pmid;
    elseif pmid*phigh < 0,
      xlow = xmid;     plow = pmid;
    else
      xlow = xmid;    xhigh = xmid;
    end
  end
  x = [xlow, xhigh];
end

```

例 5.7　二分法迭代　用二分法求方程 $f(x)=x^3-4\sin x=0$ 的根. 取初始区间[a, b]为 $a=1$, $b=3$, 最重要的是使函数值在区间端点反号. 计算中点 $m=a+(b-a)/2=2$ 处的函数值知, $f(m)$ 与 $f(a)$ 的符号相反, 所以保留初始区间的左半部, 即取 $b=m$. 重复这个过程, 直到有根区间将方程的根隔离到所需的精确度. 迭代序列如下:

a	$f(a)$	b	$f(b)$
1.000000	−2.365884	3.000000	8.435520
1.000000	−2.365884	2.000000	0.362810
1.500000	−1.739980	2.000000	0.362810
1.750000	−0.873444	2.000000	0.362810
1.875000	−0.300718	2.000000	0.362810
1.875000	−0.300718	1.937500	0.019849
1.906250	−0.143255	1.937500	0.019849
1.921875	−0.062406	1.937500	0.019849
1.929688	−0.021454	1.937500	0.019849
1.933594	−0.000846	1.937500	0.019849
1.933594	−0.000846	1.935547	0.009491
1.933594	−0.000846	1.934570	0.004320
1.933594	−0.000846	1.934082	0.001736
1.933594	−0.000846	1.933838	0.000445
1.933716	−0.000201	1.933838	0.000445
1.933716	−0.000201	1.933777	0.000122
1.933746	−0.000039	1.933777	0.000122
1.933746	−0.000039	1.933762	0.000041
1.933746	−0.000039	1.933754	0.000001
1.933750	−0.000019	1.933754	0.000001
1.933752	−0.000009	1.933754	0.000001
1.933753	−0.000004	1.933754	0.000001

不动点迭代

不动点迭代的思想为，将原问题变形为：$x = g(x)$的形式，再进行求解，我们可以构造格式$x_{k+1} = gx_k$来对非线性方程进行求解。值得一提的是，这样的构造方式并不唯一，相对应的，不同的构造方式也有不同的稳定性以及收敛速度。

例3.2.2. 非线性方程$f(x) = x^2 - x - 2 = 0$的根为$x^* = 2$和$x^* = -1$,等价的不动点问题可以包括：

1.$g(x) = x^2 - 2$;
2.$g(x) = \sqrt{x+2}$;
3.$g(x) = 1 + 2/x$;
4.$g(x) = (x^2+2)/(2x-1)$;

根据不同的迭代方程，都可求出相应的零点.

牛顿迭代

牛顿法的思想可以理解为：由于零点是其切线与x的交点，非零点则不是，于是在点x_k附近，使用$f(x_k)$处的切线来近似，使用这个切线的零点作为新的近似值，以此来靠近真正的零点。其算法流程可表示为：

x_0 =初始值
for k = 0,1,2,...
$x_{k+1} = x_k - f(x_k)/f'(x_k)$
end

例 5.10　牛顿法　用牛顿法求方程

$$f(x) = x^2 - 4\sin x = 0$$

的根.

函数 $f(x)$ 的导数为

$$f'(x) = 2x - 4\cos x,$$

因此迭代格式为

$$x_{k+1} = x_k - \frac{x_k^2 - 4\sin x_k}{2x_k - 4\cos x_k}.$$

取初始值 $x_0 = 3$，得到下面的迭代序列，其中 $h_k = -f(x_k)/f'(x_k)$ 表示迭代中 x_k 的变化量. 当 $|h_k| < |x_k|$ 或 $|f(x_k)|$，或者这两者都小于给定误差时，中止迭代.

k	x_k	$f(x_k)$	$f'(x_k)$	h_k
0	3.000000	8.435520	9.959970	−0.846942
1	2.153058	1.294772	6.505771	−0.199019
2	1.954039	0.108438	5.403795	−0.020067
3	1.933972	0.001152	5.288919	−0.000218
4	1.933754	0.000000	5.287670	0.000000

割线法

牛顿法有一个计算上的缺陷，即在每次迭代时都要计算函数及其导数的值，导数在计算中往往不方便或者计算量很大，因此在步长较小的情况下，我们可以用有限差分来近似代替导数，即用相邻两次迭代的函数值来代替其导数。这种方法叫割线法。

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

其算法流程可以表示为：

x_0 =初始值
for k = 0,1,2,...
$x_{k+1} = x_k - f(x_k)(x_k - x_{k-1}/[f(x_k) - f(x_{k-1})]$
end

如何计算多项式的所有零点？这是一个被人们研究了很久的问题。对于某些特殊情形的n阶多项式p(x)，有时要求求出它的所有n个零点，具体可以使用以下的思想来进行求值：用之前的方法，如牛顿法求出一个根x_1，考虑低一阶的收缩多项式$p(x)/(x -$

x_1)，重复此过程，直到求出全部的根。形成给定多项式的友阵，利用之前讲过的计算特征值的方法计算特征值，即计算出多项式的根。一些专门计算多项式零点的方法，如Laguerre法、Traub法等。

3.2.3 非线性方程组求解

非线性方程组的求解要比单个的非线性方程求解困难很多：

- 由于这类问题所涉及的范围更加广泛，所以对解的存在性和个数的分析也更加复杂一些。
- 利用传统的数值方法一般无法既绝对安全又收敛保证地产生有解区间。
- 随着问题维数的增加，计算量也将显著增加。

解决非线性方程组的方法也有很多，如不动点迭代、牛顿法、割线修正法、稳健性牛顿法等，这里主要介绍前三种方法。

不动点迭代

给定函数g:$\mathbb{R}^n \to \mathbb{R}^n$，则不动点问题为：寻找$\mathbf{x} \in \mathbb{R}^n$，使

$$\mathbf{x} = g(\mathbf{x}).$$

在一维情形，不动点迭代的收敛与否与收敛速度由g导数的绝对值来决定。而在高维情形下，我们有类似的谱半径条件：

$$\rho(\mathbf{G}(\mathbf{x}^*)) < 1,$$

其中$\mathbf{G}(\mathbf{x})$表示g的雅可比矩阵在x点的值，即

$$\{\mathbf{G}(\mathbf{x})_{ij} = \frac{\partial g_i(\mathbf{x})}{\partial x_j}\}$$

若满足上述条件，当初始向量充分接近解时，不动点迭代收敛，谱半径越小，收敛速度越快。

牛顿法

对可微函数$\mathbf{f}$:$\mathbb{R}^n \to \mathbb{R}^n$，利用泰勒展开可得

$$\mathbf{f}(\mathbf{x}+\mathbf{s}) \approx \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{s},$$

其中$\mathbf{J}(\mathbf{x})\mathbf{s}$是雅可比矩阵，$\{\mathbf{J}(\mathbf{x})\}_{ij} = \partial f_i(\mathbf{x})/\partial x_j$,即可通过这个迭代过程来求出非线性方程组的零点，其算法为：
$\mathbf{x}_0 =$ 初始值
for k=0,1,2,...
解$\mathbf{J}(\mathbf{x}_k)\mathbf{s}_k = -\mathbf{x}_k$求$\mathbf{s}_k$
$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$
end
每一步牛顿法都会解一个线性方程组，因此其运算量是很大的。

割线修正法

与割线法类似，它是将牛顿法中的雅可比矩阵中的偏导数用每个坐标方向上的有限差分近似代替，从而可以节省不少计算成本，其具体算法可表示为：
$\mathbf{x}_0$ = 初始值
$\mathbf{B}_0$ = 初始雅可比近似
for k=0,1,2,...
解$\mathbf{B}_k\mathbf{s}_k = -\mathbf{x}_k$求$\mathbf{s}_k$
$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$
$\mathbf{y}_k = \mathbf{f}(\mathbf{x}_{k+1}) - \mathbf{f}(\mathbf{x}_k)$
$\mathbf{B}_{k+1} = \mathbf{B}_k + [(\mathbf{y}_k - \mathbf{B}_k\mathbf{s}_k)\mathbf{S}_k^T]/(\mathbf{s}_k^T\mathbf{s}_k)$
end
这个方法即可加快迭代进程，提升计算的效率。

3.3 数值最优化

3.3.1 优化问题

优化问题来自科学和工程的各个领域，在商业和工业的中的应用也非常普遍。任何设计问题一般都要优化某些性能指标，如成本或效率。在数学上，优化问题可以表示称确定某些参数，使给定函数在某个定义域上取得极值，也就是说:

定义3.3.1 (优化问题)**.** 给定一个函数f:$\mathbb{R}^n \to \mathbb{R}$和一个几何$S \in \mathbb{R}^n$,使$f$在$x^*$处的值是其在$S$上的最小值。

由于f最大值可以考虑-f的最小值，因此我们可以在这里只讨论最小值问题。如果不对目标函数f与集合S做某些假定，那么我们很难建立优化问题最优解的存在性和唯一性。因此，我们接下来会给出一些限制，在这些限制下，再来讨论具体的数值优化方法。

在所有假设中，最重要的假设就是凸性

定义3.3.2 (凸性)**.** 称集合S是凸集，即对所有x,$y$$\in S$，有

$$\{ax + (1-a)y : a \in [0,1]\} \in S$$

进一步，如果函数沿S中任意线段的图像都不超过连接线段端点函数值的弦，则称函数f在凸集S上是凸的。

3.3.2 一维优化

我们从一维优化方法开始讨论，这个问题不仅重要，并且是高维优化问题中许多算法的子问题。首先我们需要一个确定最小值点所在区间范围的方法，若在区间[a,b]上函数f只有一个峰值，则称函数是单峰的。

黄金分割搜索

假定f在[a,b]上是单峰的，$x_1, x_2 \in [a,b]$，且$x_1 < x_2$，比较两点的函数值，利用单峰的性质可以去掉$(x_2, b], [a, x_1)$中的一个，使最小值点位于剩下的子区间内。如果$f(x_1) < f(x_2)$,则最小值点肯定不会在$[a, x_1)$内，反之同理，这样就将最小值的区间缩小了一部分。我们为使包含最小值区间长度的降低规则变化，每个新区间两个端点的相对位置应该与上一段相同。由此我们得到黄金分割搜索的方法。

例 6.8　黄金分割搜索　用黄金分割搜索求函数

$$f(x) = 0.5 - x\mathrm{e}^{-x^2}$$

的最小值.

从初始区间[0,2]开始,计算点 $x_1 = 0.764$ 和 $x_2 = 1.236$ 处的函数值,得 $f(x_1) = 0.074, f(x_2) = 0.232$. 由于 $f(x_1) < f(x_2)$,所以最小值点一定落在区间$[a, x_2]$内,用 x_2 代替 b 并重复这个过程. 图 6.6 描述了第一次迭代,下表给出的是整个迭代序列:

x_1	$f(x_1)$	x_2	$f(x_2)$
0.763932	0.073809	1.236068	0.231775
0.472136	0.122204	0.763932	0.073809
0.763932	0.073809	0.944272	0.112868
0.652476	0.073740	0.763932	0.073809
0.583592	0.084857	0.652476	0.073740
0.652476	0.073740	0.695048	0.071243
0.695048	0.071243	0.721360	0.071291
0.678787	0.071815	0.695048	0.071243
0.695048	0.071243	0.705098	0.071122
0.705098	0.071122	0.711310	0.071133
0.701260	0.071147	0.705098	0.071122
0.705098	0.071122	0.707471	0.071118
0.707471	0.071118	0.708937	0.071121
0.706565	0.071118	0.707471	0.071118

逐次抛物插值

优化问题的黄金分割搜索除了对函数值进行比较外不再使用这些值，而逐次抛物插值的思想是，通过利用更多的这些函数值，加快方法的收敛速度。其具体流程是，首先用最小化函数在三个点上的值做二次多项式拟合，假定所做抛物线的最小值只有一个，以它作为函数最小值的新的近似，将前面三个点去掉一个，重复此过程直至收敛。计算与上一个方法相同的例子，我们有

计算三个点 $x_0=0, x_1=1.2, x_2=0.6$ 上的函数值，得 $f(x_0)=0.5, f(x_1)=0.216$，$f(x_2)=0.081$. 用这三个点做抛物拟合，并取其最小值点 $x_3=0.754$ 作为解的下一个近似. 去掉 x_0，用剩余的三个点重复这个过程. 图 6.8 描述了第一次迭代. 迭代序列如下：

k	x_k	$f(x_k)$
0	0.000000	0.500000
1	1.200000	0.215687
2	0.600000	0.081394
3	0.754267	0.072981
4	0.720797	0.071278
5	0.708374	0.071119
6	0.706920	0.071118
7	0.707103	0.071118

牛顿法

局部抛物近似的另一种方法是使用截断的泰勒级数展开，即：

$$f(x+h) \approx f(x) + f'(x)h + \frac{1}{2}f''(x)h^2.$$

则我们可以得到最小值为 $h=-f'(x)/f''(x)$，其算法可以写作：

x_0 =初始值
for $k=0,1,2,dots$
$x_{k+1}=x_k - f'(x_k)/f''(x_k)$
end

例 6.10 牛顿法 用牛顿法求例 6.8 中函数

$$f(x)=0.5-x\mathrm{e}^{-x^2}$$

的最小值.

f 的一阶和二阶导数

$$f'(x)=(2x^2-1)\mathrm{e}^{-x^2}, \qquad f''(x)=2x(3-2x^2)\mathrm{e}^{-x^2}$$

所以得求 f 最小值的牛顿迭代公式

$$x_{k+1}=x_k-(2x_k^2-1)/(2x_k(2-2x_k^2)).$$

取初值 $x_0=1$，得到下列迭代序列：

k	x_k	$f(x_k)$
0	1.000000	0.132121
1	0.500000	0.110600
2	0.700000	0.071162
3	0.707072	0.071118
4	0.707107	0.071118

3.3.3 无约束优化

接下来考虑多维无约束优化，它的许多特点与一维优化和求解n维非线性方程组是相同的。

最速下降法

基于一维优化的黄金分割搜索算法提出的。但它不能像黄金分割搜索那样保证收敛。

最速下降法思想是，任意选取一初始点，以其负梯度方向作为其下降方向，走一段距离到下一个点，再重复这样的步骤，最终得到收敛的结果。其算法可以表示为：
$\mathbf{x}_0$ =初始值
for $k = 0, 1, 2, \ldots$
$s_k = -\nabla f(\mathbf{x}_k)$
选择α_k使$f(\mathbf{x}_k + \alpha_k s_k)$最小化
$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k s_k$
end
最速下降法是十分可靠的方法，只要负梯度不为0即可继续进行。但这种方法很难看出函数的性态。

牛顿法

同样用局部抛物近似作为目标函数的近似，由截断泰勒级数展开，有：

$$f(x+s) \approx f(x) + \nabla f(x)^T s + \frac{1}{2} s^T H_f(x) s.$$

其中$H_f(x)$是f的二阶偏导数的黑塞矩阵，即$\{H_f(x)\}_{ij} = \partial^2 f(x)/\partial x_i \partial x_j$,由此我们可以得到无约束优化的牛顿法的算法为：
$\mathbf{x}_0$ =初始值
for $k = 0, 1, 2, \ldots$
解$H_f(\mathbf{x}_k)s_k = -\nabla f(\mathbf{x}_k) B s_k$
$\mathbf{x}_{k+1} = \mathbf{x}_k + s_k$
end 与一维情形类似，拟牛顿法和割线法对于无法计算二阶导数(Hessian矩阵)的情形

共轭梯度法

共轭梯度法师牛顿法的一个改进方案，这种方法不需要二阶导数，也不需要存储黑塞矩阵，对于大型问题来说是非常合适的，其算法可以表示为：
$\mathbf{x}_0$ =初始值
$\mathbf{g}_0 = \nabla f(\mathbf{x}_0)$
$\mathbf{s}_0 = -\mathbf{g}_0$
for $k = 0, 1, 2, \ldots$
选择α_k使$f(\mathbf{x}_k + \alpha_k s_k)$最小化
$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k s_k$
$\mathbf{g}_{k+1} = \nabla f(x_{k+1})$
$\beta_{k+1} = (\mathbf{g}_{k+1}^T \mathbf{g}_{k+1} / (\mathbf{g}_k^T \mathbf{g}_k) \quad \mathbf{s}_{k+1} = -\mathbf{g}_{k+1} + \beta_{k+1} \mathbf{s}_k$
end

3.3.4 约束优化

接下来考虑有关约束优化的方法，首先考虑形如

$$\min_x f(\mathbf{x})$$

$$s.t.\ \mathbf{g}(\mathbf{x}) = \mathbf{0}$$

的等式优化问题。

罚函数与障碍函数法

这类方法的思想是将约束优化问题转化为无约束优化问题，再用之前的方法解决问题。罚函数法师通过目标函数和约束条件加权组合的最小化同步求得目标函数的最小值，例如对于之前提到的等式优化问题，可以等价提出一种无约束问题：

$$\min_x \phi_p(\mathbf{x}) = f(\mathbf{x}) + \frac{1}{2}\rho \mathbf{g}(\mathbf{x})^T \mathbf{g}(\mathbf{x})$$

的解，在适当的条件下可以证明此无约束问题的最小值是收敛到约束优化问题的。

例 6.16　罚函数法　用罚函数法求解例 6.7 的约束优化问题，并将约束变成等式.

对这个问题，有

$$\phi_\rho(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{2}\rho \boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{g}(\boldsymbol{x}) = 0.5x_1^2 + 2.5x_2^2 + 0.5\rho(x_2 - x_1 + 1)^2,$$

以 $\boldsymbol{x}_0 = [1, 1]^{\mathrm{T}}$ 为初始点，取 $\rho = 1$，用无约束优化程序求出 ϕ_ρ 的最小值 $\boldsymbol{x}_\rho^*$，然后按每次 10 倍逐步增加 ρ，并将前面的解作为初始值，计算新的无约束最小值，下表列出了得到的无约束最小值序列. 当 $\rho = 1000$ 时，解在表中所显示的这些位上是精确的.

ρ	$\boldsymbol{x}_\rho^{*\mathrm{T}}$	
1	0.454	−0.091
10	0.769	−0.154
100	0.826	−0.165
1000	0.833	−0.167

线性规划

线性规划是一种最重要、最普遍的约束优化，在这类问题中目标函数和约束都是线性的，一种标准形式可以定义为：

$$\min_x f(\mathbf{x}) = \mathbf{c}^T \mathbf{x}$$

$$s.t.\ \mathbf{A}\mathbf{x} = \mathbf{b}; \mathbf{x} \geq 0$$

这类问题的可行域是$\mathbb{R}^n$中的凸多面体，其全局最小值一定在凸多面体的某个顶点出现。下面以图解法来示例线性规划的流程。假设求解问题：

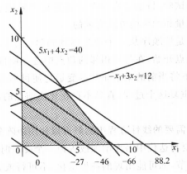
$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = -8x_1 - 11x_2$$

$$\text{s. t.} \quad 5x_1 + 4x_2 \leqslant 40,$$

$$-x_1 + 3x_2 \leqslant 12,$$

$$x_1 \geqslant 0, \quad x_2 \geqslant 0.$$

这个问题的可行域是坐标轴和另外两条直线所围成的区域，如图 6.11 中的阴影部分所示. 图中还画出了目标函数的等值线，并在图的下方标出了相应的目标函数值. 最小值点一定是可行域的顶点，这里 $\boldsymbol{x}^* = [3.79, 5.26]^{\mathrm{T}}$，图中用圆点表示，在这个点上目标函数值为 −88.2.



3.3.5 随机梯度下降法

软件包与参考材料

Exercise